REAL-TIME GESTURE IMITATION IN A SOFT-ARM CONTROL ROBOT

By

Sean Robert Thornton

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Electrical Engineering

May, 2009

Nashville, Tennessee

Approved:

Professor Richard Alan Peters II

Professor D. Mitchell Wilkes

“I can do all things through Christ who strengthens me.”

—Philippians 4:13

ACKNOWLEDGEMENTS

The author first and foremost thanks his Heavenly Father and Savior, through whom all things are possible, for His divine guidance and aid, without which this thesis could never have been written. The author also thanks his advisor, Dr. Peters, for all of the patient support and expertise provided over the course of this thesis. The author also expresses his deepest gratitude for the financial and academic support so kindly provided by the Vanderbilt University School of Engineering and the Graduate School and for the faculty and staff thereof.

Special thanks also go out to my parents, Leon and Jane for the encouragement, teaching, and love they have provided for my entire life. I would also like to thank my soon-to-be in-laws for the loving and encouraging environment which they have created for me in Tennessee these past two years. Last, but certainly not least, I thank my beloved fiancée, Alex, for her love, prayers, and abundant patience as she has waited for me these many months to finish this endeavor.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ISAC	Intelligent SoftArm Control
OpenCV	Open Computer Vision library
UDP	User Datagram Protocol
PID	Proportional-Differential-Integral controller
DAQ	Data Acquisition
CCD	Charge-Coupled Device
NTSC	National Television System Committee
FOV	Field of View
PTU	Pan-Tilt Unit
PC	Personal Computer
CRL	Cognitive Robotics Laboratory
DLL	Dynamic Link Library
GUI	Graphical User Interface
DOS	Disk Operating System
DOF	Degrees of Freedom
PCI	Peripheral Component Interconnect
MLL	Machine Learning Library
IPP	Integrated Performance Primitives
XML	Extensible Mark-up Language
ASL	American Sign Language

ANN	Associative Neural Network
TCP/IP	Transmission Control Protocol / Internet Protocol
ROI	Region of Interest
CPP	C-Plus-Plus source code file
H	C-Plus-Plus header code file

CHAPTER I

INTRODUCTION

Ever since the earliest days in the field of robotics—even since Čapek first popularized the notion of robots in science fiction literature—robots have been commonly conceptualized in mainstream culture to have a humanoid structure; that is, they are often visualized as having a torso, two arms, two legs, and a head [1, 2]. Although there a numerous examples of non-humanoid robots at work in industry, as in [3] and [4], much research has been directed towards the achievement of this robotic "ideal" [5, 6].

The humanoid structure is not without disadvantages, however. The very nature of this structure involves very many redundant degrees of freedom, resulting in high dimensionality in its configuration space and, consequently, difficult challenges in its modeling and control [5]. There have been a number of methods attempted to simplify the control of such complex manipulators, most of which involve some form of machine learning. There have been a wide range of controllers employing the techniques of fuzzy logic [7], neural networks [8, 9], genetic algorithms [10], and most notably, imitation learning [11].

I.1. Imitation Learning

As Schaal suggests, imitation learning may be the vehicle which will enable researchers to achieve this ideal of a humanoid robot [11]. According to Schaal, imitation learning is simply "a method to bias learning towards a particular solution, i.e., that of the teacher" which "has a profound impact on how quickly new skills can be acquired;" that is to say, imitation learning in robotics is the same method for learning that a human student uses—observe the instructor's solution, abstract the necessary procedural information, and copy that same procedure to solve other similar problems [12]. Billard further clarifies that, as in the case of a human student, "imitation is more that the mere ability to reproduce others' actions; it is the ability to replicate and, by so doing, learn 'new' skills (i.e., skills that are not part of the animal's [or robot's] usual repertoire) by the simple observation of those performed by others" [13]. In other words, it is not enough for an imitator to blindly mimic the behavior of another, but the imitator must be able to discern certain information about the goal of the behavior.

Although this aim of imitation learning—abstraction of the "why" of behavior—is beyond the scope of this work, the system developed in this document will provide a framework whereby further experiments directed towards this goal may be implemented. A summary of this framework follows and will be discussed at length in later chapters.

I.2. Summary of Procedures

This document details a system whereby ISAC, Vanderbilt University's soft-arm control humanoid robot, is able to detect hands and faces, follow their relative motions,

and duplicate these motions using his own architecture. As described in more detail in the following chapter, ISAC has two cameras for eyes and two arms which are actuated by McKibben artificial muscles. The goal of this application is to use these devices to implement the system just described; once this goal is complete, this system can be used as a foundation for other work in the field of imitation learning.

The system is divided into three subsystems: the vision subsystem, the imitation processing subsystem, and the actuation subsystem. The vision subsystem detects the face and a hand of the human to be imitated. The OpenCV libraries for Haar object detection are used in each of two stereo images to detect first the face and then the right hand of the human and its motion relative to the face. Using geometry and known properties of the cameras, the positions of these objects in the stereo images are then transformed into three-dimensional locations.

The imitation processing subsystem takes the vision data as input and outputs arm actuation data. It first takes the three-dimensional locations of the user's face and hand and maps them onto the workspace of ISAC, and then applies the inverse kinematics for the left arm of ISAC to determine the joint angles necessary to actuate that arm to the desired location. This subsystem also transmits the joint angles by means of a UDP datagram to the actuation subsystem, which resides on another machine.

The actuation subsystem was developed prior to this application by Ulutas *et al.*, and consists of a combination neural network/PID controller, rotary encoders, electro-pneumatic valves, and DAQ cards to interface the controller to the arm hardware of ISAC

[8]. These devices are all explained further in Chapter II, but it is important to note here that only the PID portion of the controller is used for this application.

These three subsystems run in a loop until the user chooses to terminate the imitation process. The loop must execute very quickly in order to achieve real-time or near real-time imitation. This speed of imitation is necessary to assure in future experiments that ISAC is correctly perceiving the motions of the human.

Due to the subjective nature of the quality of imitation, the experiments devised to test this system are likewise subjective. A tape measure was set up along the three spatial axes of the laboratory to measure the movements of the right hand of the human to be imitated. These movements consisted of vertical, horizontal, sagittal, and diagonal motions and were plotted on *x*-*y*, *x*-*z*, and *y*-*z*, graphs. The encoder data from the left arm of ISAC was also recorded. After applying forward kinematics to this data, the resultant (*x*, *y*, *z*) position of the end-effector was plotted alongside the motions of the human for comparison. These different motions were also repeated at "high," "medium," and "low" speeds to test the reaction time of the system.

The organization of the rest of this document is as follows: background material on previously developed hardware and software platforms is detailed in Chapter II, the details of the implementation of this system are explained in Chapter III, more in-depth explanations of the experiments and their results are presented in Chapter IV, and finally conclusions are drawn in Chapter V.

CHAPTER II

BACKGROUND INFORMATION

There are many pre-existing hardware and software platforms upon which this system is implemented. This chapter discusses in greater detail the origins and applications of these platforms.

II.1. ISAC

Intelligent SoftArm Control, more commonly known as ISAC, is an 18-dof humanoid robot in the Cognitive Robotics Laboratory at Vanderbilt University. ISAC has many anthropomorphic members which allow him to imitate human motions in a several ways. The hardware which implements his two most important members, eyes and arms, is described in the sections that follow.

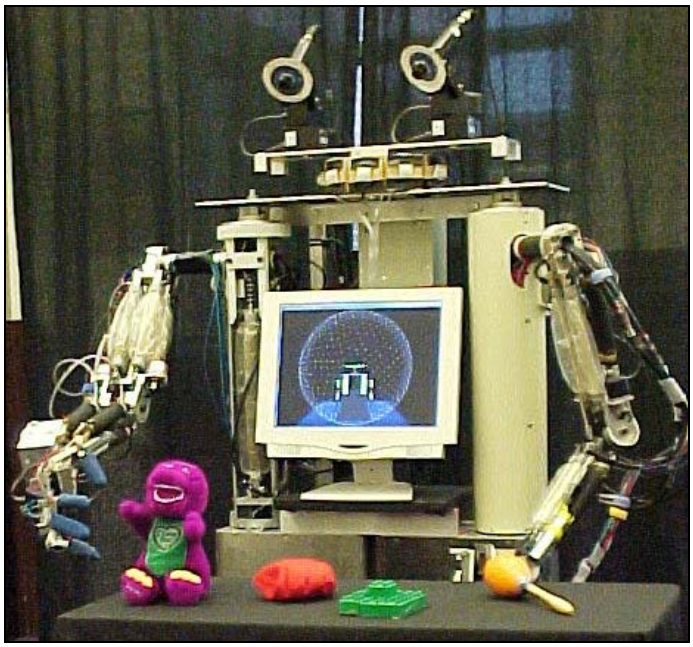


Figure 1: The humanoid robot ISAC [14]

II.1.1. Vision Hardware

A number of hardware components compose the vision subsystem for ISAC. These include charge-coupled device (CCD) cameras to detect images, pan-tilt units to direct the cameras at a desired target, and frame grabbers to capture images into computer software.

Sony® CCD Cameras

ISAC uses two Sony® XC-999 CCD cameras, shown below in Figure 2, which are situated approximately 11 inches apart. The effective NTSC resolution of these cameras is 768x494, although most applications on ISAC only operate them at 320x240, due to substantially decreased image processing time for smaller images [15]. The field

of view (FOV) of these cameras has been reported to be approximately 55.77° horizontal and 42.78° vertical [16].

Figure 2: Sony® XC-999 CCD Camera [15]

Directed Perception Pan-Tilt Units

The CCD cameras are each mounted on top of a Directed Perception PTU-46-17.5 pan-tilt unit, shown in Figure 3. These units, along with the pair of cameras, provide ISAC with stereo vision—that is, the ability to localize objects in three dimensions. Each PTU has an angular resolution of .013°, can rotate at velocities up to 300°/s, and can support a payload up to 6 lbs—more than enough to support the weight of the CCD cameras used [17]. In this application, however, once each PTU has been directed at the target, their positions are fixed, so no precise high-velocity movements are required.

Figure 3: Directed Perception PTU-46 series pan-tilt unit with controller [18]

CyberOptics Imagenation Frame Grabbers

The video feed of each CCD camera is routed to its own CyberOptics Imagenation PXC-200 Frame Grabber, shown in Figure 4, mounted in the PC "Sally" in the CRL. These frame grabbers and their software drivers interface with C/C++ and Visual Basic programs by means of certain DLL files (explained further in sections II.3.1. and III.1.) to allow for control of the frame grabber, manipulation of frame data, and creation of a simple GUI under the DOS operating system [19].

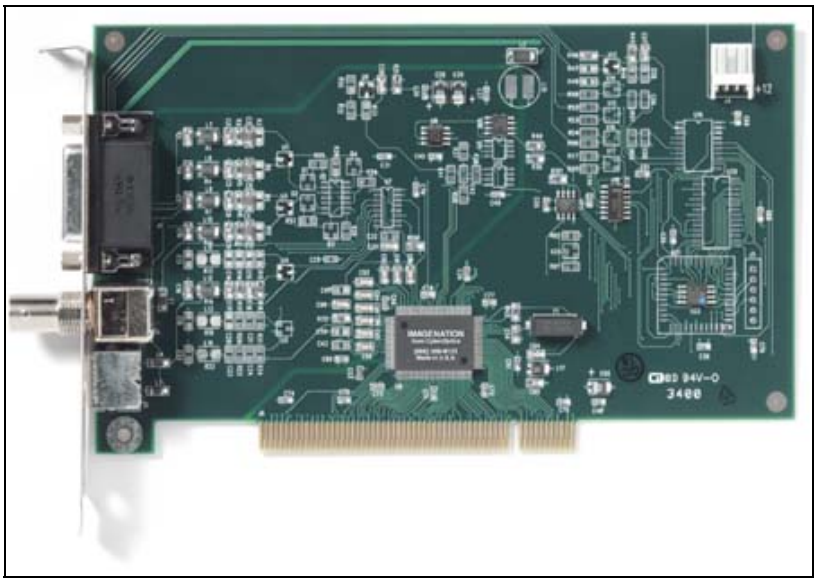


Figure 4: CyberOptics Imagenation PXC-200 series PCI frame grabber [20]

II.1.2. Actuation Hardware

Just as many components are necessary for ISAC to have vision, so also are several components required for the movement of the two humanoid arms, the most important of which are McKibben artificial muscles, air pressure valves, rotary encoders, and control boards for the valves and encoders. The mechanical layout of these components is somewhat complex.

McKibben Artificial Muscles

A McKibben artificial muscle is a muscle-like device which consists of an inflatable air bladder surrounded by a double-helical mesh, as shown in Figure 5. The combination of these two materials results in the property that, when filled by compressed air, the radial expansion of the bladder results in axial contraction of the mesh [21]. These artificial muscles are advantageous for a couple of reasons. First, they

have a very high strength-to-weight ratio, and second, they are compliant and flexible and thus not dangerous when operated by untrained personnel [22, 23].

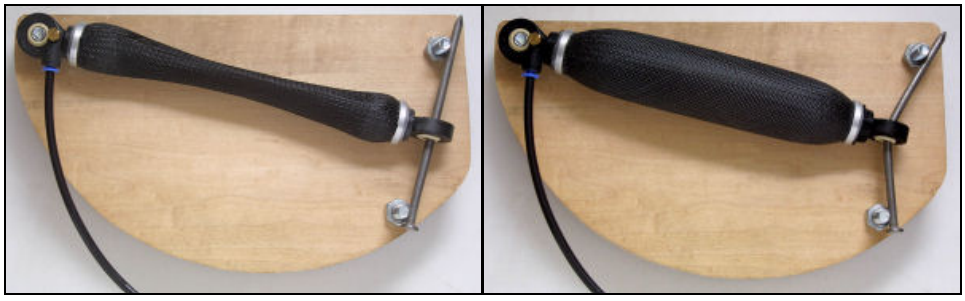


Figure 5: McKibben artificial muscle in relaxed and contracted states [24]

McKibben artificial muscles are not without disadvantages, however; many of their properties make their behavior somewhat difficult to predict, and thus difficult to control accurately. These disadvantages include the storage of energy in the elastic of the bladder, fatigue in the bladder and mesh material, friction between the bladder and the mesh, hysteresis, response non-linearity, and delay from the pressure controller [22, 23].

Each of the 6-dof arms of ISAC is actuated by 12 such artificial muscles. Since this type of artificial muscle can only generate a contractual force, they are laid out on ISAC in agonist pairs to enable each joint to move in both directions. This configuration also reduces a number of the disadvantages mentioned above [23]. As explained in detail in [16], the actuated angle of each joint is assigned a number 0 through 5, with angle zero corresponding to the rotation of the "trunk" of each arm, angle 1 to the rotation of the shoulder, angles 2 and 3 to the flexion and rotation of the forearm, and angles 4 and 5 to the pitch and roll of the end-effector. An overview of the left arm is shown in Figure 6.

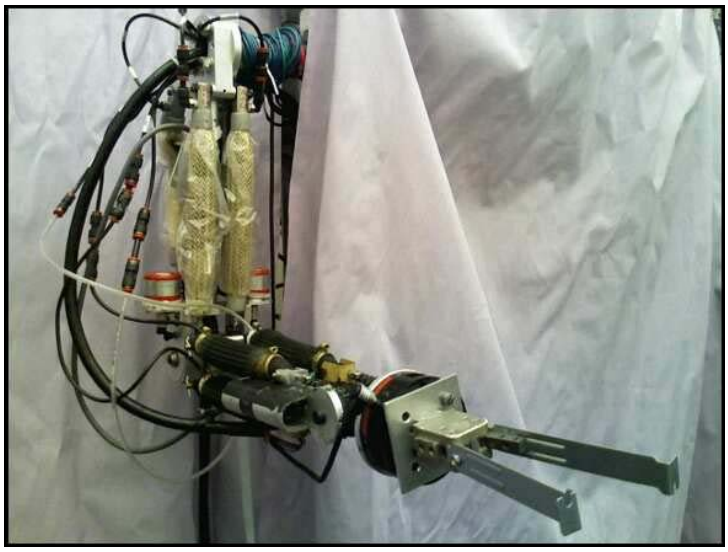


Figure 6: The right arm of ISAC [16]

The complexity in the design arises from the elbow joint, about which the forearm is both flexed and rotated, and the wrist joint, about which the end-effector is both flexed and rotated. Both of these joints must be controlled by two agonist pairs of muscles, one for each degree of freedom. As is also explained in [16], these pairs are laid out and numbered according to Figure 7 and actuate a universal joint as diagrammed in Figure 8.

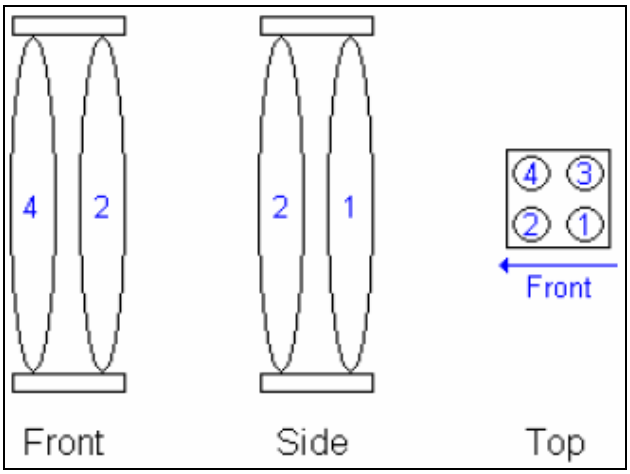


Figure 7: Configuration of a double-pair of opposing artificial muscles [16]

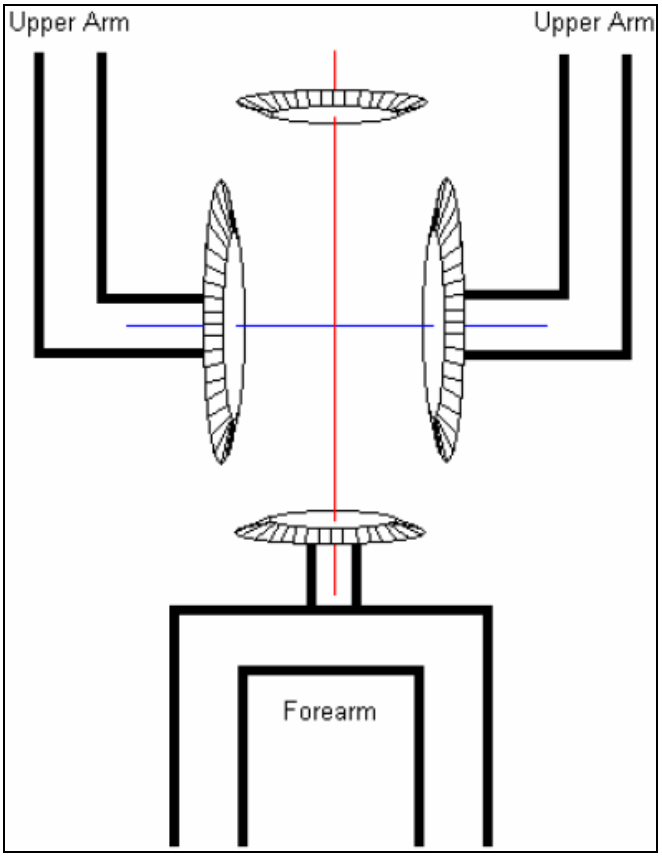


Figure 8: Universal joint at elbow/wrist [16]

The result of this architecture is that ISAC can, for example, flex its forearm by contracting bicep muscles number 2 and 4 and relaxing tricep muscles number 1 and 3, or ISAC can rotate its forearm by contracting one bicep muscle and the diagonal tricep muscle (e.g. number 1 and 4) and relaxing the opposing two muscles [16].

SMC Electro-Pneumatic Regulator Valves

The McKibben artificial muscles are powered by compressed air. This air is first filtered and cooled to keep the temperature dependence of the muscle behavior to a minimum and then passed through an SMC ITV 2050-312CN4 model electro-pneumatic regulator valve (Figure 9) for each muscle—12 valves for each arm.

Figure 9: SMC ITV 2050 series electro-pneumatic regulator valve [25]

Given an input signal from 0 to 5 volts, these valves regulate the air pressure for each of the McKibben artificial muscles between 5 and 900 kPa, thus allowing the arm to take on a continuous variety of configurations [25]. When the input signal decreases, the air pressure is relieved through an exhaust port to allow the corresponding pressure decrease.

These valves are organized into two arrays—one on the chest of ISAC, and one on the back. The 12 valves on the chest regulate the pressures for the right arm, and the 12 on the back regulate the left.

Rotary Encoders

To properly control the movement of the arms, feedback about the current angles of the arms is necessary. This information is obtained from rotary encoders attached at each axis of rotation on each arm. Although product data on the SUMTAK and Epson-Seiko rotary encoders used on the arms is no longer available due to the discontinuation of these products, most rotary encoders operate on the same basic principles. A light source inside the encoder directs light onto a rotating code disk and through slits in that disk. The light that shines through the slits then passes onto a stationary phase plate, which also contains slits. There are two pairs of these slits, which are spaced such that light signals passing through will be 90° out of phase. These light signals are then detected and processed into digital signals which can be read by a controller—the number of cycles of light and dark correspond to how far the encoder has rotated within a certain resolution [26]. The resolution of the encoders used on ISAC has been reported to be .345° [16]. Figure 10 shows an overview of rotary encoder operation.

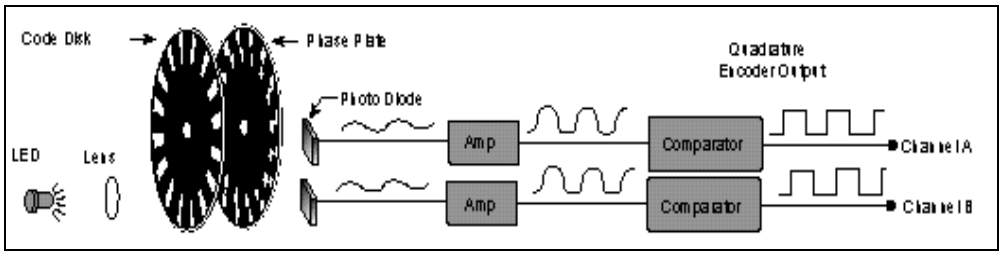


Figure 10: Basics of rotary encoder operation [26]

Vital Systems Controller Cards

Each of the electro-pneumatic regulators is controlled by a 0 to 5 volt input signal, and each encoder value must be identified. This is achieved by interfacing them with the PC "Octavia" in the CRL through three Vital Systems MOTENC-Lite PCI boards, shown in Figure 11.

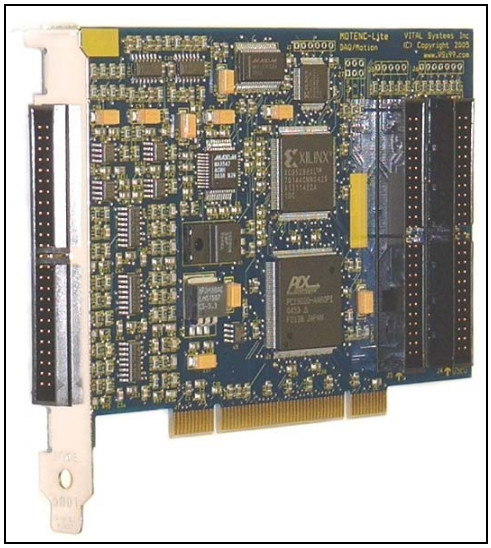


Figure 11: MOTENC-Lite PCI board [27]

Each of these boards has 8 analog outputs and 4 encoder inputs, so three boards are just enough to handle communication with the 24 valves and 12 encoders on the two arms of ISAC [27]. These boards are connected via ribbon cables to termination boards on ISAC, which serve as a simple connector between the PCI boards and the valves and encoders. These termination boards are shown in Figure 12.

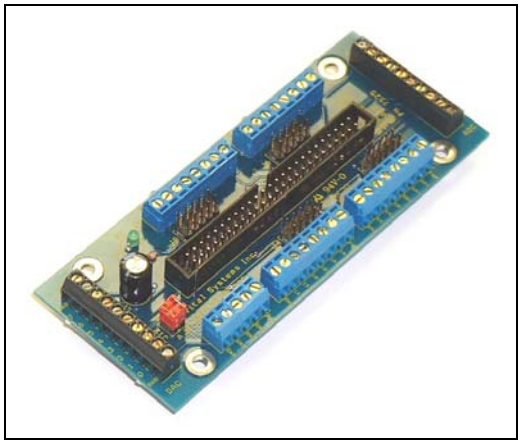


Figure 12: DAC/ADC/Encoder Termination Board for MOTENC-Lite [28]

Each of these boards supports the 8 analog outputs and 4 encoder inputs for each MOTENC-Lite PCI board. According to [29], black terminal blocks J11 and J10, on the lower left and upper right, respectively, handle analog output and input, respectively, and the 8-connection blue terminal blocks J3, J5, J7, and J9 handle inputs from encoders. (The header pins on the board also support the same four encoder channels, but these are not used.)

II.2. OpenCV

For ISAC to imitate the hand gestures of a human being, it needs to see the gesture. Thus this project requires a substantial amount of computer vision source code, almost all of which was developed using OpenCV, an open-source computer vision library [30]. Originally developed by Intel in an effort to make core functions for computer vision available to anyone working in the field, OpenCV has been used in a wide variety of applications since its alpha release in January of 1999 [31]. Some of these applications include object detection, medical imaging, and security systems.

The OpenCV library includes functions for capturing images from still or video cameras into an object called `IplImage` and processing those images in a variety of ways [32]. The `IplImage` structure, shown in Figure 13, also contains a variety of parameters useful for image processing [33]. For brevity, these parameters will not be discussed here, although some will be addressed as necessary in later sections.

```
typedef struct _IplImage
{
    int  nSize;         /* sizeof(IplImage) */
    int  ID;            /* version (=0)*/
    int  nChannels;     /* Most of OpenCV functions support 1,2,3 or
                           4 channels */
    int  alphaChannel;  /* ignored by OpenCV */
    int  depth;         /* pixel depth in bits: IPL_DEPTH_8U,
                           IPL_DEPTH_8S, IPL_DEPTH_16S, IPL_DEPTH_32S,
                           IPL_DEPTH_32F and IPL_DEPTH_64F are
                           supported */
    char colorModel[4]; /* ignored by OpenCV */
    char channelSeq[4]; /* ditto */
    int  dataOrder;     /* 0 - interleaved color channels, 1 -
                           separate color channels. cvCreateImage can
                           only create interleaved images */
    int  origin;        /* 0 - top-left origin, 1 - bottom-left origin
                           (Windows bitmaps style) */
    int  align;         /* Alignment of image rows (4 or 8). OpenCV
                           ignores it and uses widthStep instead */
    int  width;         /* image width in pixels */
    int  height;        /* image height in pixels */
    struct _IplROI *roi;/* image ROI. if NULL, the whole image is
                           selected */
    struct _IplImage *maskROI; /* must be NULL */
    void  *imageId;     /* ditto */
    struct _IplTileInfo *tileInfo; /* ditto */
    int  imageSize;     /* image data size in bytes
                           (==image->height*image->widthStep
                           in case of interleaved data)*/
    char *imageData;  /* pointer to aligned image data */
    int  widthStep;   /* size of aligned image row in bytes */
    int  BorderMode[4]; /* ignored by OpenCV */
    int  BorderConst[4]; /* ditto */
    char *imageDataOrigin; /* pointer to very origin of image data
                              (not necessarily aligned) -
                              needed for correct deallocation */
}
IplImage;
```

Figure 13: Definition of IplImage

OpenCV also contains a library called HighGUI, for development of simple GUIs within a C/C++ program, and a library called MLL, which contains certain machine

learning methods which are often useful in computer vision applications, such as pattern recognition [34, 35].

OpenCV functions are optimized for computational efficiency, but can be enhanced further by the addition of Intel's Integrated Performance Primitives (IPP) library. This library installs additional optimized algorithms for multimedia and communications processing which can be used by OpenCV for greater processing speed; however, although OpenCV can be used on virtually any machine, IPP can only be used on Intel architectures [31, 36]. Due to the high cost of IPP, this application does not incorporate this library, although further discussion exists in section 2 of Chapter V.

II.3. Prior Work

Although the focus of this project is to provide a real-time imitation algorithm for later imitation learning experiments in the CRL, many other robotic disciplines are involved, such as computer vision and actuator control. Several previous projects in the CRL have already provided software for these, so for simplicity, much of this code was borrowed. Explanations of each borrowed portion are provided in the following sections.

II.3.1. PXC Samples

The vision subsystem for ISAC, described in further detail in section 1 of Chapter III, is built on two CyberOptics Imagenation PXC-200 Frame Grabber PCI cards shown in Figure 4. Most vision software for ISAC uses C code called CPXCk_FG.h and CPXCk_FG.c, developed by an unknown author for the CRL and modified in February of

2005 by Katherine Achim. In order to use this code in an OpenCV application, images captured by the frame grabbers must first be saved to the hard drive as bitmap images and then loaded back into the program as an `IplImage` structure. Unfortunately, this introduces significant overhead in processing time, so this code was rejected in favor of developing a faster routine.

The driver software for the frame grabbers comes packaged with sample programs, sample C source code, and programming libraries for developers. The functions in these libraries are explained in [19]. Portions of the source code for the sample program `pxcdraw1.exe`, which captures images into a memory buffer and displays them on the screen, among other things, were transported into C++ for use in this application. Thus, captured frames can be manipulated by OpenCV in this application without the requirement of being saved and reopened every time.

II.3.2. Camera Head

The Imagenation PXC-200 Frame Grabbers are each mounted on the Directed Perception PTU-46 series pan-tilt units shown in Figure 3. Source code for a C++ interface with these pan-tilt units is contained in `CameraHead.h` and `CameraHead.cpp`, developed by Palis Ratanaswasd for the CRL. This code allows pan and tilt angles to be directly set by a simple C++ function call. It was included in this application to allow the image planes of the cameras to be directed towards and focus on the human being whom ISAC is intended to imitate.

II.3.3. Haar Face Detection

As explained in [16, 31], a function for face detection from the OpenCV ML library, called cvHaarDetectObjects(), uses patterns of Haar-like features to detect objects. Haar-like features, which are described by relationships of light and dark regions as shown in Figure 14, are called such due to their similarity to Haar wavelets, a mathematical construct proposed in 1909 by Alfréd Haar [37]. This function is based on an algorithm developed by Paul Viola and Michael Jones in [38] for face detection, but has been enhanced by Rainer Lienhart and Jochen Maydt in [39] by adding the use of the diagonal features shown in Figure 14 below and by the inclusion of the library functions createsamples() and haartraining(), which allow the user to train and detect objects other than faces [31].

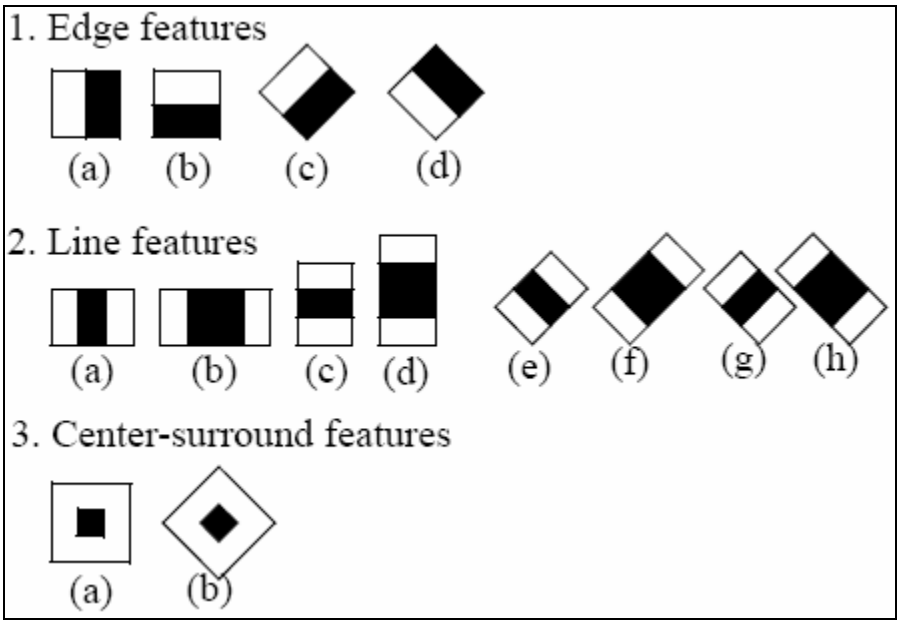


Figure 14: Haar-like features [39]

Objects are trained by providing the haartraining() function with a large number of positive and negative samples of the object. This function then builds a boosted rejection cascade from the data and saves the cascade in an XML file for later use by the cvHaarDetectObjects() function. Although the training of new objects was not explicitly employed in this application, more details on this procedure are found in [40].

A sample program for using cvHaarDetectObjects() to detect faces in [31] was modified by Sean Begley in [16] to detect only the most prominent face and was further modified by this author to detect the most prominent face and hand. The two XML cascade files used are haarcascade_frontalface_alt2.xml, provided by OpenCV, and aGest.xml, a cascade developed by Juan Wachs at Ben-Gurion University of the Negev for detecting a hand held in the ASL position for the letter "A," shown in Figure 15 below [41].

Figure 15: The ASL letter "A" [42]

II.3.4. NN/PID Controller

For ISAC to successfully imitate the movements of a human being, there must be a fast way to move its arms to the correct location. An arm controller for ISAC was recently developed by Ulutas *et al.* which combines a classic PID controller with a neural network. The first portion of this controller makes use of an ANN, which has learned the relationship between the input voltages to the electro-pneumatic valves and the resultant joint angles, to move the end-effector into a sphere of radius *r* around the goal position. Then the controller switches to a classical PID architecture to move the end-effector as close to the goal position as possible [8]. When incorporating that controller with this application, it was found that the neural network portion responded too slowly, while the PID portion performed well at any distance from the target location. The source code for this controller was therefore modified to omit the neural network portion. Other minor changes were made to simplify the code. It was also altered to include a UDP communication interface with the vision subsystem described later in section III.1. This communication interface will be described in more detail in section III.2.

II.3.5. PracticalSocket

Since the vision subsystem and actuation subsystem operate on two separate PCs, it was necessary to establish a fast method of communication between them. A straightforward way to implement TCP/IP and UDP sockets in C programming is presented in [43] along with a number of sample programs. A very simple C++ wrapper for these sockets, called `PracticalSocket`, is available for download from their website

[44]. For simplicity, this application uses UDP sockets, whereby a simple C string can be transmitted to a specific computer by calling the function `UDPSocket::sendTo(sendString, strlen(sendString), destAddress, destPort)`, where `destAddress` is the IP address of the recipient machine, and `destPort` is the port which the recipient machine has opened for communication. On the receiving end, a call to `UDPSocket::recvFrom()`, which takes corresponding arguments, will retrieve the string sent and store the IP address and port of the broadcasting machine.

CHAPTER III

REAL-TIME IMITATION

For ISAC to imitate a human being's hand movements in real-time, several smaller tasks must be completed in a loop which iterates over the entire duration of the imitation. These tasks are summarized in Table 1.

The key challenge faced in the implementation of each of these tasks is that ISAC must imitate in real-time. Each task in the loop must execute quickly enough for the overall loop to keep up with the hand motion. If the loop were too slow, the human being making the hand gestures could simply move slowly enough for the system to keep pace, but it is assumed that such behavior is undesirable—the system should be able to follow movements at normal human speeds.

A flowchart for this system is shown in Figure 16, and a summary of the tasks associated with each portion of the flowchart is shown in Table 1. This chapter explains the implementation details of each of the tasks listed in Table 1, with special attention given to methods designed to decrease overall processing time.

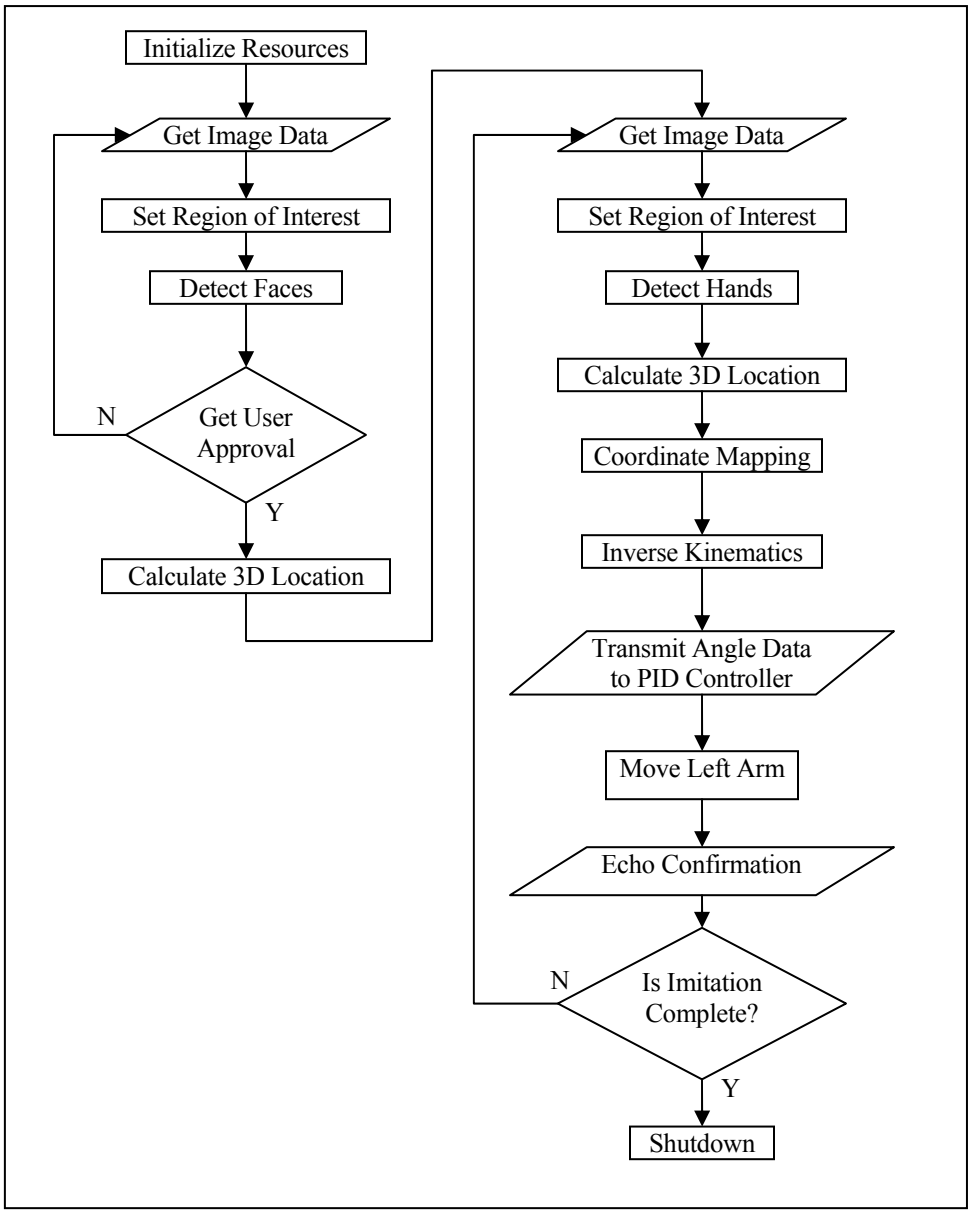


Figure 16: Main program flow

Table 1: Summary of steps to be completed during main program flow

1.	Initialize all resources (libraries, frame grabbers, frame buffers, pan-tilt units, etc)
2.	Grab two grayscale frames (left and right) into `IplImage` objects
3.	Set ROI (region of interest) in each image for a face
4.	Call `cvHaarDetectObjects()` and find the faces
5.	Repeat steps 2 through 4 until user input indicates approval of the detected faces
6.	Calculate the three-dimensional position of the face
7.	Repeat steps 2 through 4, but this time detect the hands
8.	Calculate the three-dimensional position of the hand
9.	Map those coordinates into the corresponding coordinates in the workspace of ISAC
10.	Apply the inverse kinematics of ISAC to calculate the necessary joint angles
11.	Transmit these angles to the PID controller
12.	Direct the PID controller to reach to the given angles
13.	Echo confirmation to vision subsystem to continue
14.	Repeat steps 7 through 13 until the motion to be imitated is complete

III.1. Vision Subsystem

Steps 2 through 8 in Table 1 above are the tasks pertaining to the vision subsystem of this application.

III.1.1. Image Capture

Prior vision systems for ISAC made use of the CPXCk_FG.cpp functions for capturing images, but this required saving grabbed images to the hard drive before they could be opened as IplImage objects. In this application, a simple function called IplCopy() was developed which makes use of the PXC200 Frame Grabber and Frame Libraries in the files pxc_95.dll and frame_32.dll to simplify and accelerate this process [19]. This function is shown in Figure 17.

```
/*************************************************************************
 * Name:         IplCopy                                                 *
 * Description:  Grabs image from frame grabber and stores it as an IplImage *
 *************************************************************************/
void IplCopy(long fg, FRAME __PX_FAR *fb)
{
    // grab the image
    pxc_library.Grab(fg, fb, (short)grab_type);

    // direct the IplImage structure to the grabbed data
    buffer_addr = frame_library.FrameBuffer(fb);
    image->imageData = (char *)buffer_addr;
}
```

Figure 17: IplCopy() implementation

This function calls the Grab() function in the frame grabber library to capture an image into a frame buffer, and then calls the FrameBuffer() function in the frame library to retrieve the starting address for the image data in that buffer. Then, simply assigning that address to the imageData pointer in an IplImage allows that data to be later processed by OpenCV.

III.1.2. Haar Object Detection

A sample program provided by [31] demonstrates how to use the cvHaarDetectObejcts() function to detect faces in an image. This function is not limited to faces, however, and can detect any well-defined object, provided that an appropriate XML cascade file has be trained and loaded. This application uses a modified version of that sample program, called haarDetect(), to detect the most prominent face or the most prominent hand. A condensed version of haarDetect() is shown in Figure 18.

This function first initializes some variables for detecting the largest object, and then sets the ROI of the image. Since the face of the human to be imitated is always within the same region of the image, setting the ROI reduces processing time for this step, since cvHaarDetectObjects() will not have to search the entire image.

Once all of the faces or hands have been detected, they are stored in a CvSeq object. A simple loop runs through the entire sequence of objects until the object occupying the largest area in the image is found. This object is accepted as the face or hand of the human. The ROI is reset so that later OpenCV functions which may need the entire image will have access to it, and finally the center of the detected object is returned to the calling function.

The function cvHaarDetectObjects() can detect any object with an appropriately trained XML cascade file. The cascades used in this application are haarcascade_fontalface_alt2.xml, which is provided with OpenCV, and aGest.xml, which was trained by Juan Wachs at Ben-Gurion University of the Negev to detect the ASL letter "A" when made by the user's right hand [41].

```
/*****************************************************************************
 * Name:         haarDetect                                                  *
 * Description:  Detects the specified object in an image (based on Sean     *
 *               Begley's / Learning OpenCV's code)                          *
 *****************************************************************************/
CvPoint haarDetect(IplImage *img, int objectType)
{
    int i;
    int area, maxArea = 0;
    CvSeq *haarObjects = 0;
    CvRect imgROI,
           *object,
           maxObject = cvRect(0,0,0,0);
    CvPoint objectCenter;
    // these steps are necessary for accurate detection
    cvEqualizeHist(img, img);
    cvClearMemStorage(storage);
    // select the type of object to be detected
    if (objectType == FACE)
    {
        // set region of interest in img to save processing time
        imgROI.x = img->width / 3;      // right 2/3 horizonally
        imgROI.width = imgROI.x * 2;
        imgROI.y = img->height / 3;     // middle 1/3 vertically
        imgROI.height = imgROI.y;
        cvSetImageROI(img, imgROI);
        // detect the faces in img -- see OpenCV documentation for details on
        // other parameters
        haarObjects = cvHaarDetectObjects(img, face_cascade, storage,
                                          1.1, 2, 0, cvSize(30, 30));
    }   // else statements for other objects (i.e. HAND)

    for(i = 0; i < (haarObjects ? haarObjects->total : 0); i++)
    {   // scan through the CvSeq of detected objects and find the largest
        object = (CvRect *)cvGetSeqElem(haarObjects, i);
        area = object->width * object->height;
        // save the current object if it is the largest thus far
        if (area > maxArea)
            maxObject = *object;
    }
    cvResetImageROI(img);   // reset ROI for proper image display
    // calculate and return the center of the detected object
    objectCenter.x = maxObject.x + maxObject.width / 2 + imgROI.x;
    objectCenter.y = maxObject.y + maxObject.height / 2 + imgROI.y;
    return objectCenter;
}
```

Figure 18: Condensed haarDetect() function

III.1.3. Three-Dimensional Localization

For ISAC to correctly interpret the user's motion from the two camera images, the relative three-dimensional positions of the hand and face need to be calculated. To

accomplish this, ISAC makes use of stereopsis to calculate three dimensional locations from known angles, dimensions of the cameras, and the two dimensional locations in each image. The equations for a stereopsis method known as Depth from Disparity are reported in [16] and for brevity are not reproduced here. This application makes use of a function called calculateXYZ(), shown condensed in Figure 19, based on a function of the same name in [16]. The primary difference is that the coordinate axes have been interchanged to match the axes defined by the kinematics of ISAC.

```
/*****************************************************************************
 * Name:         calculateXYZ                                                *
 * Description:  Calculates the 3-dimensional location of an object shown in *
 *               stereo images (based on Sean Begley's code)                 *
 *****************************************************************************/
void calculateXYZ(CvPoint3D32f *objloc, CvPoint left, CvPoint right)
{
    double f = 15262.5;         // focal length = 305.25mm * 50px/mm
    double b = 14000;           // base length = 280mm * 50px/mm
    double yri, yli, zri, zli;  // coordinates in each image
    double x = 0;               // distance from ISAC
    double yr, yl, zr, zl;      // actual vertical-horizontal coordinates
    double z = 0, y = 0;

    yri = (double)right.x;      // get pixel coordinates
    yli = (double)left.x;
    zri = (double)right.y;
    zli = (double)left.y;

    x = f * b / (yli - yri);    // calculate x in pixels

    yr = yri * x / f;           // calculate real y and z values in pixels
    yl = yli * x / f;
    zr = zri * x / f;
    zl = zli * x / f;

    y = (yr + yl) / 2;          // average real coordinates from each image
    z = (zl + zr) / 2;

    objloc->x = x;              // write the values into the objloc to be
    objloc->y = y;              // returned
    objloc->z = z;
}
```

Figure 19: Condensed calculateXYZ() function

III.2. Imitation Processing Subsystem

The next three steps in Table 1—steps 9 through 11—comprise the subsystem by which ISAC converts the movements of the human into movements of ISAC's arm.

III.2.1 Mapping to ISAC

Perhaps the most important function of this subsystem is implemented by `mapToISAC()`, which takes the three-dimensional locations of the face and hand of the human and transforms them onto the physical structure of ISAC. This function first subtracts the locations of the hand and face in order to find the relative position of the hand, and then uses experimentally determined constants to scale and shift this position to the left arm of ISAC. The resultant position is then checked against these same constants to ensure that the desired location does not fall beyond the workspace of ISAC. This function is shown condensed in Figure 20.

```
/*****************************************************************************
 * Name:         mapToISAC                                                   *
 * Description:  Map the locations of the face and hand onto ISAC's body     *
 *****************************************************************************/
void mapToISAC(CvPoint3D32f *hand, CvPoint3D32f *face)
{
    // find hand position relative to face and scale/shift to ISAC
    hand->x = -(hand->x - face->x) * ((ISAC_X_MAX-ISAC_X_MIN) /
                                      (STEREO_X_MAX-STEREO_X_MIN))
                                      + ISAC_X_MIN;
    hand->y = -(hand->y - face->y) * ((ISAC_Y_MAX-ISAC_Y_MIN) /
                                      (STEREO_Y_MAX-STEREO_Y_MIN))
                                      + ISAC_Y_MIN;
    hand->z = -(hand->y - face->z) * ((ISAC_Z_MAX-ISAC_Z_MIN) /
                                      (STEREO_Z_MAX-STEREO_Z_MIN))
                                      + ISAC_Z_MIN;
    // if the hand location exceeds ISAC's workspace, cap it off
    if (hand->x > ISAC_X_MAX) hand->x = ISAC_X_MAX;
    if (hand->x < ISAC_X_MIN) hand->x = ISAC_X_MIN;
    if (hand->y > ISAC_Y_MAX) hand->y = ISAC_Y_MAX;
    if (hand->y < ISAC_Y_MIN) hand->y = ISAC_Y_MIN;
    if (hand->z > ISAC_Z_MAX) hand->z = ISAC_Z_MAX;
    if (hand->z < ISAC_Z_MIN) hand->z = ISAC_Z_MIN;
}
```

Figure 20: Condensed mapToISAC() function

III.2.2 Inverse Kinematics

Once the desired location of the end-effector has been calculated by mapToISAC(), the inverse kinematics for the left arm need to be applied to determine the necessary joint angles. This is accomplished by the function simpleInverseKinematics(), also based on a function of the same name found in [16]. This function is too long to display here, but it is included along with all other functions in Appendix A. Since the current configuration of the arm has no end-effector, it makes sense to treat this arm as a simple 3-dof manipulator.

III.2.3 UDP Transmission

The final task of this subsystem is to transmit the angle data to the PC Octavia on which the actuation subsystem is implemented. Prior to this application, the method for accomplishing communication between these two computers was to copy files from one to the other using Windows Explorer. However, this process is too slow to be repeated at each step of the imitation, so this application makes use of UDP datagrams to transmit the information. Thus, this step must format and send the joint angle data as a UDP datagram to Octavia. The formation is accomplished by formatUDP(), shown in Figure 21, which places the first three joint angles returned by the inverse kinematics into a tab-delimited C-string. This string is then transmitted by the UDPSocket::sendTo() function (cf. section II.3.5.).

```
/*****************************************************************************
 * Name:         formatUDP                                                   *
 * Description:  Converts the given angle data into a string transmittable   *
 *               via the UDP socket                                          *
 *****************************************************************************/
string formatUDP(double *angles)
{
    string UDPstring;
    char numbuffer[33];

    UDPstring = "";
    sprintf(numbuffer,"%.0f",angles[0]);
    UDPstring.append(numbuffer);
    UDPstring.append("\t");
    sprintf(numbuffer,"%.0f",angles[1]);
    UDPstring.append(numbuffer);
    UDPstring.append("\t");
    sprintf(numbuffer,"%.0f",angles[2]);
    UDPstring.append(numbuffer);

    return UDPstring;
}
```

Figure 21: formatUDP() function

III.3. Actuation Subsystem

The final steps in Table 1 are the function of the actuation subsystem. It takes a given set of joint angles (received via UDP datagram) and commands the arm to that location by determining appropriate pressure values for each McKibben artificial muscle. The code for these functions, originally written by Ulutas *et al.* (and heavily modified by this author for clarity, to include UDP communication, *et cetera*) is too long to reproduce here [8]. It is, however, included along with all other functions in Appendix A.

After initializing the electro-pneumatic valves to what is called the "home" position—the arms of ISAC held straight down with the elbows bent forward at 90°—and initializing the rotary encoders to a value of zero, this subsystem enters a loop whereby it receives joint angle information from the imitation processing subsystem, activates a PID controller to actuate the left arm towards those desired joint angles, and then echoes the received UDP datagram to the vision subsystem as confirmation that the movement is complete. This effectively closes the loop outlined in Figure 16, returning the application to step 7 shown in Table 1.

The PID controller implemented by [8] is actually implemented as a simple P controller which directs the left arm to the desired position by adding a proportion of the error in the joint angles to the current joint angles. The proportionality constants used for θ_1, θ_2, and θ_3 are .0036, .0052, and .006, respectively. These values are small so as to minimize overshoot and to reduce the arm velocity to non-hazardous levels. The original implementation also iterated the P controller over 55 steps, pausing for 40 milliseconds at each step—this was found to be both excessively accurate and excessively sluggish, so

these numbers were reduced to a single step with no pause. This reduced accuracy is admissible for three reasons: First, it does not reduce the similarity between the motions of the arm of ISAC and those of the human; second, it speeds up this portion of the loop, thus enabling a real-time response; and third, since the single step brings the arm to a position relatively close to that desired, the human need merely to pause in the same position for a few iterations of the loop if reaching this exact position is necessary.

This loop will not be exited unless the desired elbow-joint angle received is 9999—an obvious physical impossibility for the arm. This angle is transmitted to the actuation subsystem only if the user chooses to terminate the imitation. Upon exiting the loop, the arms are returned to the "home" position and then slowly depressurized to equilibrium with the pressure in the lab.

CHAPTER IV

EXPERIMENT

Several experiments have been devised to demonstrate the successful operation of this system. Owing to the subjective nature of imitation, assigning a value of "success" to any particular experiment is also subjective; however, measures have been taken to reduce that subjectivity as much as possible.

IV.1. Goals

The goal of each experiment was to determine whether or not ISAC can successfully imitate motions in a particular direction, that is, in the coordinate frame of ISAC, the *x* (sagittal), *y* (horizontal), and *z* (vertical) directions, and combinations of the three (diagonal), in real-time or near real-time. Several motions were demonstrated to ISAC in each of four directions at varying speeds, and its response to each was measured. It is desirable for the motion of ISAC to be in the same direction as that of the human, to have the same trajectory as that of the human, and to occur in tandem with that of the human.

Unfortunately, the morphology of the arms of ISAC limits the workspace to a region so narrow and irregular that it is impossible to generate any substantial amount of motion in the sagittal or vertical directions without also incurring the other, so special

consideration was given to these cases. Provided that the majority of the motion occurred in the desired direction, interference in the other was tolerated.

Similarity in the directions and trajectories of the two motions was compared by visual inspection of the data plotted for the human arm and that of ISAC; laying *x*-*y*, *x*-*z*, and *y*-*z* plots of the two motions side-by-side demonstrates either obvious similarity or obvious dissimilarity. Lastly, allowing for a small delay between the motion of the human and that of ISAC, the imitation must occur simultaneously with the demonstration. This is also apparent on plots of the two motions side-by-side.

IV.2. Procedures

To determine the success of the system, each trial in the experiment was conducted under the following guidelines. First, a tape measure was oriented in the desired direction (say, vertically) to act as a guide for the hand. The hand moved back and forth in a single triangle wave pulse, or as close to such as could be approximated by a human. The coordinates of the hand as detected by the vision subsystem in each frame for the duration of the motion was written to a file for later comparison with the motion of the arm of ISAC. To eliminate visual anomalies—false detections of the hand of the human—a light colored curtain was draped behind the human to block out the background noise, and a light was positioned in front of the human to better illuminate the hand and face.

Next, as the arm of ISAC moved up and down along an imitated trajectory, the values of the rotary encoders at each point in the trajectory were also written to a file.

Once the imitation was complete, the encoder values were used to determine the joint angles and, by application of the forward kinematics of the arm, the end-effector position throughout the duration of the motion.

These two sets of data, the hand and end-effector coordinates, were then plotted side-by-side (not on the same axes since the scales are quite different) on the same sets of coordinate axes, namely *x*-*y*, *x*-*z*, and *y*-*z* axes, for comparison. The trajectories were compared by visual inspection, and the concurrence of the two was determined by comparing the peak times of their respective triangle wave pulses. If the motions occurred primarily in the same direction (with the allowances for sagittal and vertical mixing as described above), had a high correlation in trajectory, and had a small delay, the trial was deemed a success. Five trials each were recorded for motions in the vertical, horizontal, sagittal, and diagonal (vertical-horizontal) directions at each of three speeds—"high," "medium," and "low." The results of these trials are summarized in the following section and included in their entirety in Appendix B.

IV.3. Results

Every one of the trials for each experiment was at least moderately successful, and most of them were right on target. Summaries of these trials can be seen in the tables and figures which follow in this section.

IV.3.1. Vertical Motion

Tables 2 – 4 summarize the results of the vertical experiment. Grades of success, failure, and moderate—a trial with approximate similarity but errors too substantial to ignore—were assigned to each trial.

Table 2: Vertical motion at "low" speed

Trial	1	2	3	4	5
Trajectory	success	success	success	success	success
Delay	15%	4%	8%	3%	9%

Table 3: Vertical motion at "medium" speed

Trial	1	2	3	4	5
Trajectory	success	success	success	success	success
Delay	11%	7%	2%	15%	13%

Table 4: Vertical motion at "high" speed

Trial	1	2	3	4	5
Trajectory	success	success	success	success	success
Delay	7%	13%	15%	15%	12%

Samples of the trajectories of these motions are shown in Figure 22. Note that the scale of sagittal movement is reduced by a factor of 50. This is due to the fact that calculating the depth of objects in a stereo image introduces an additional factor of σ, which in this application is equal to 50px/mm. It can be seen on these plots that although the vertical motions match, and although there is very little movement in the horizontal

direction, there is some movement in the sagittal direction. This is expected, however; as stated before, the limitations of the workspace of ISAC prevent motion in the vertical or sagittal directions without interference appearing in the other. It can also be seen in the trajectories of the motion of the arm of ISAC that "tails" hang off of each of the curves—these points correspond to the movement of the arm from the "home" position to the initial point of the imitation motion.

A sample of vertical delay is also shown in Figure 23. Since each data set lasts for equal duration in time but contains a different number of data points, the delay was calculated as a ratio of the whole. In the case shown in Figure 23, it can be seen that the arm of ISAC takes 8% longer (with respect to the entire motion) than the arm of the human. For all of the experiments, a value of 10% or less, which is barely noticeable, was considered real-rime, 10-25% was considered near real-time, and anything larger than 25% was considered too much delay to be useful. All plots of the trajectories and delays for each trial at each speed are presented in their entirety in Appendix B.

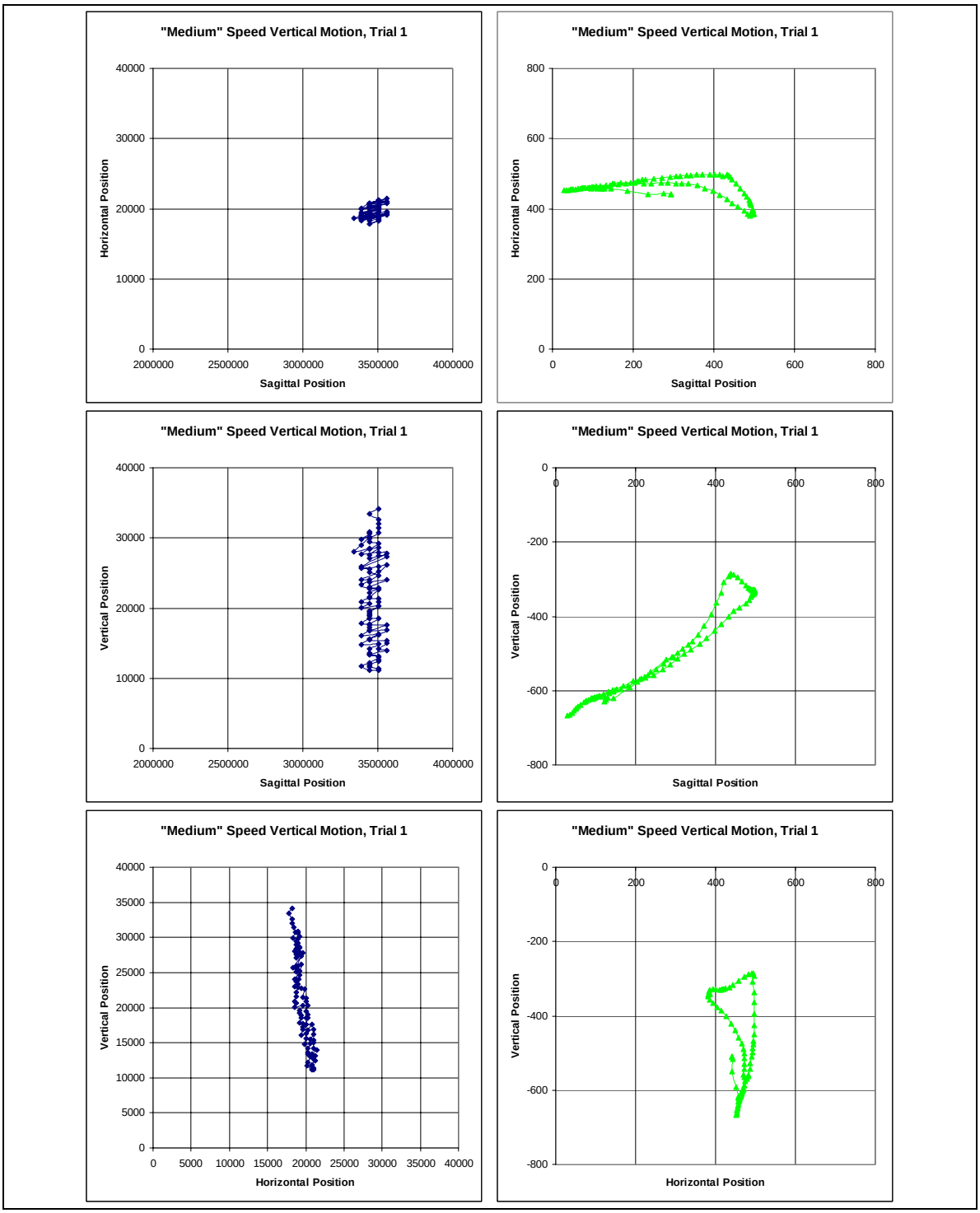


Figure 22: A trial of "medium" speed vertical motion; human motion is shown on the left, that of ISAC on the right

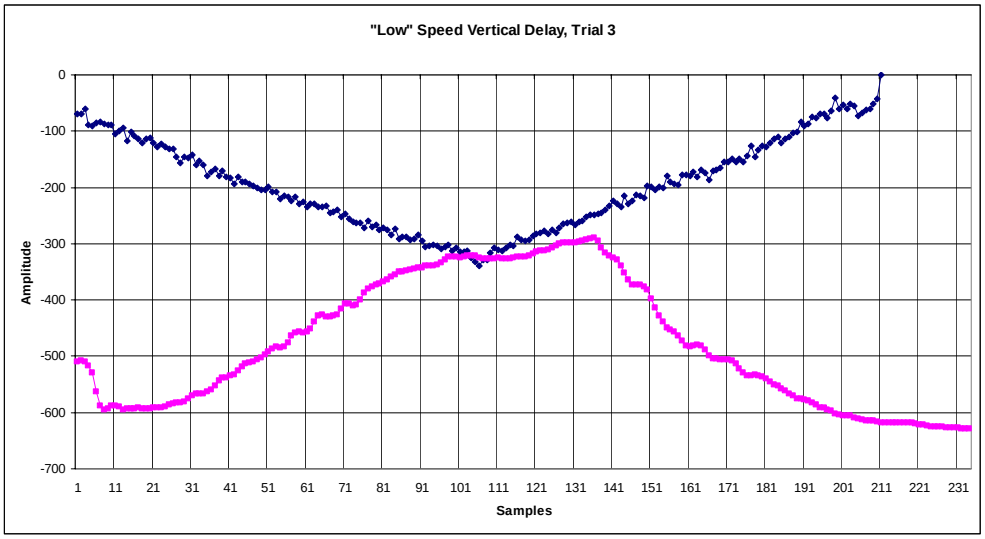


Figure 23: A trial of "low" speed vertical motion; human motion is on above and that of ISAC is below

IV.3.2. Horizontal Motion

Tables 5 – 7 summarize the results of the horizontal motion trials, and Figures 24 and 25 show samples of their trajectories and delays. As in the previous experiment, grades of success, failure, and moderate were given to each trial trajectory, delays of <10% were considered ideal (real-time), delays of 10-25% were considered acceptable, and longer delays were considered unacceptable.

Table 5: Horizontal motion at "low" speed

Trial	1	2	3	4	5
Trajectory	success	success	success	success	success
Delay	8%	7%	5%	6%	4%

Table 6: Horizontal motion at "medium" speed

Trial	1	2	3	4	5
Trajectory	success	success	success	success	success
Delay	5%	16%	13%	8%	6%

Table 7: Horizontal motion at "high" speed

Trial	1	2	3	4	5
Trajectory	success	moderate	success	success	success
Delay	11%	16%	12%	18%	9%

It can be seen from the trajectories in Figure 24 that horizontal hand motion is not a natural motion for a human to make, and despite the precautions taken in the experiment, it was still difficult to maneuver in that fashion. The limitations on the workspace of ISAC, however, actually helped in this case—since ISAC was unable to move sagittally as much as horizontally, these errors made by the human did not carry through to movements of the arm of ISAC. "Tails" can also be seen on these trajectories.

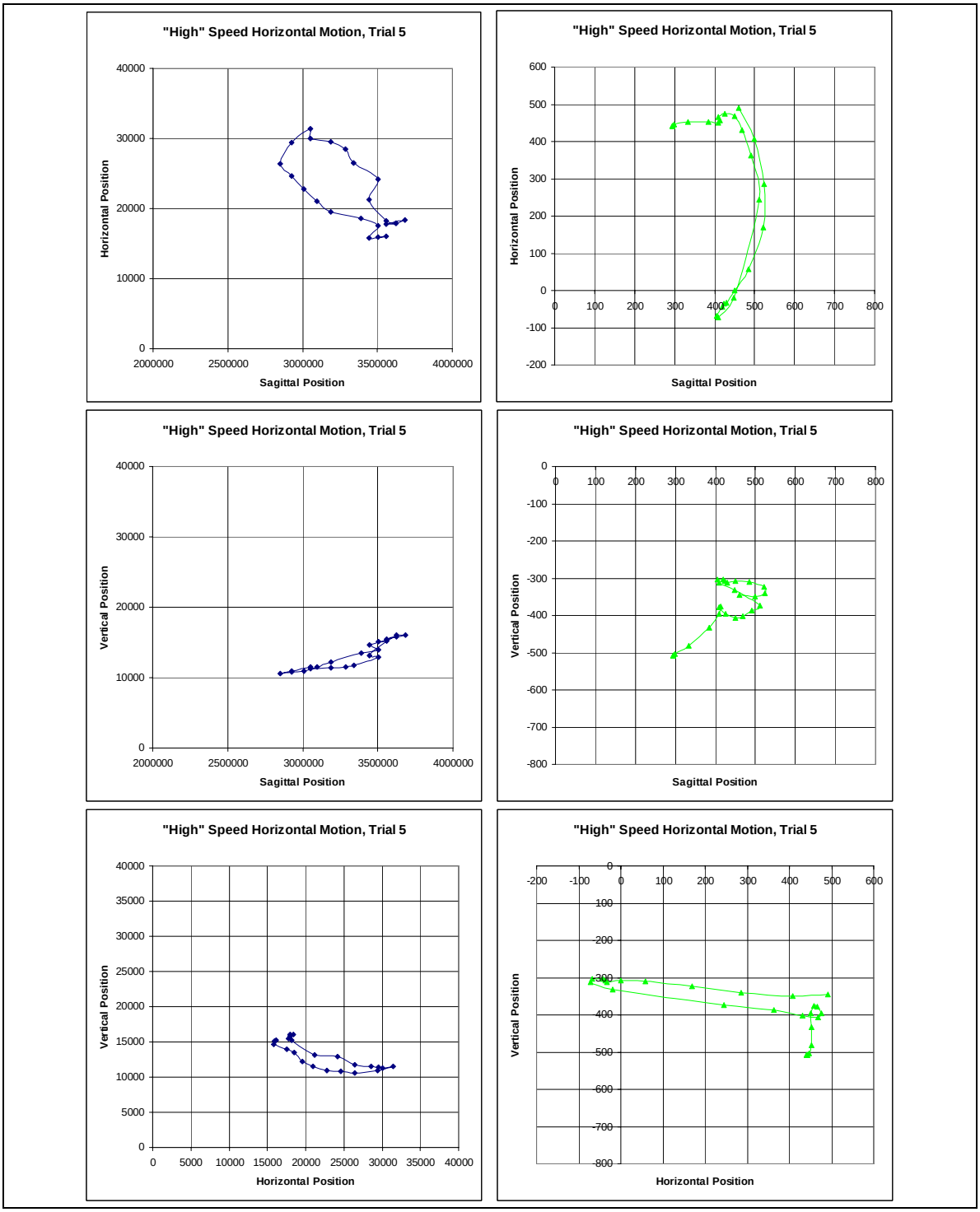


Figure 24: A trial of "high" speed horizontal motion; human motion is shown on the left, that of ISAC on the right

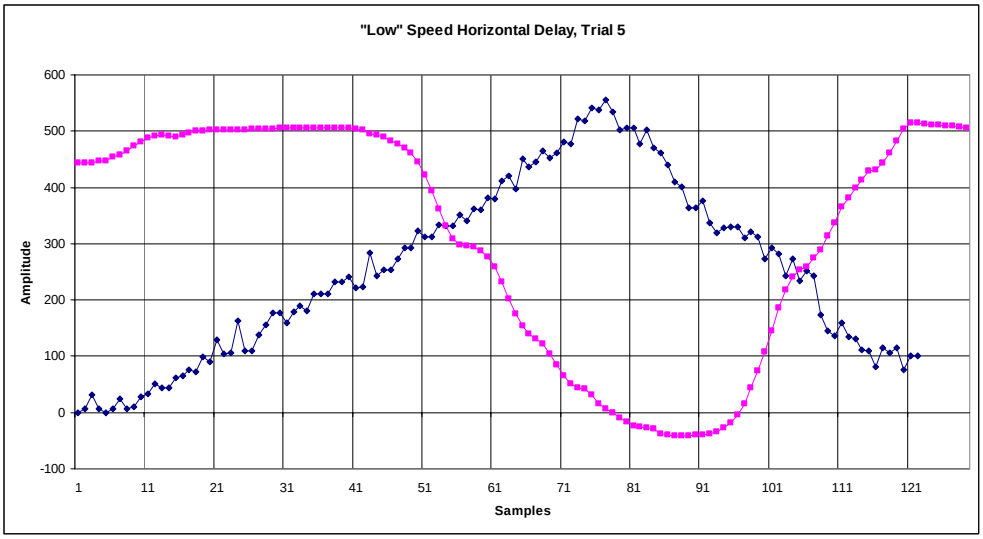


Figure 25: A trial of "low" speed horizontal motion; the motion of ISAC is the upper curve and that of the human is the lower

IV.3.3. Sagittal Motion

Tables 8 – 10 summarize the results of each trial in the sagittal motion experiment, and Figures 26 and 27 contain samples of their trajectories and delay.

Table 8: Sagittal motion at "low" speed

Trial	1	2	3	4	5
Trajectory	success	success	success	success	success
Delay	<1%	<1%	6%	<1%	7%

Table 9: Sagittal motion at "medium" speed

Trial	1	2	3	4	5
Trajectory	success	success	success	success	success
Delay	7%	15%	14%	15%	10%

Table 10: Sagittal motion at "high" speed

Trial	1	2	3	4	5
Trajectory	success	success	success	success	success
Delay	14%	8%	19%	4%	11%

It can be seen from these trajectories that ISAC responds to sagittal motions very well. As in the last trial, some human error introduced slight vertical motion into the test, and this was exaggerated by the limitations of the workspace of ISAC which cause coupling in vertical and sagittal motions. Despite this error, however, there was a very obvious correlation between the motion of the human and that of ISAC.

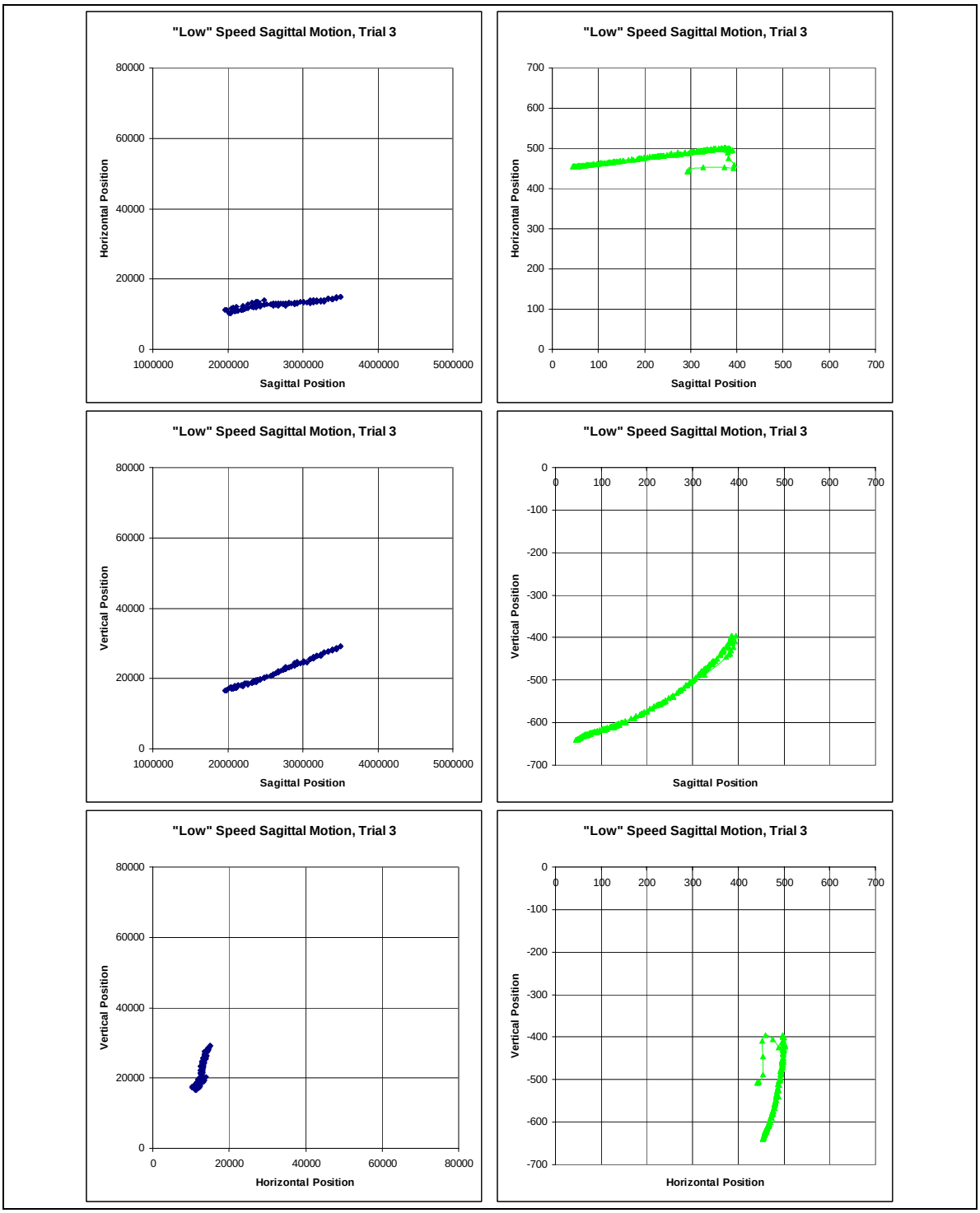


Figure 26: A trial of "low" speed sagittal motion; human motion is shown on the left, that of ISAC on the right

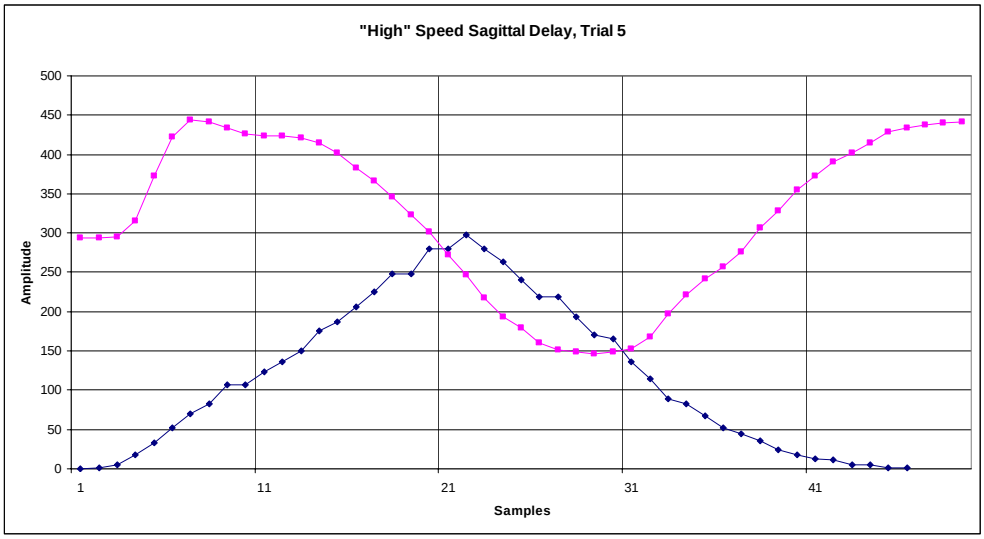


Figure 27: A trial of "high" speed sagittal delay; the motion of ISAC is above and that of the human below

IV.3.4. Diagonal Motion

Tables 11 – 13 summarize the results of the horizontal motion trials, and Figures 28 and 29 show samples of their trajectories and delays.

Table 11: Diagonal motion at "low" speed

Trial	1	2	3	4	5
Trajectory	success	success	success	moderate	success
Delay	~8%	~1%	~2%	~1%	~2%

Table 12: Diagonal motion at "medium" speed

Trial	1	2	3	4	5
Trajectory	success	success	moderate	moderate	success
Delay	~2%	~6%	~1%	6%	~1%

Table 13: Diagonal motion at "high" speed

Trial	1	2	3	4	5
Trajectory	success	success	success	success	success
Delay	6%	17%	8%	2%	~2%

The trajectories of these motions exhibit much of the irregularities found in previous experiments. Diagonal motion is, like horizontal, a very unnatural movement for a human, but the limitations on the workspace of ISAC did not propagate these errors. As with the vertical experiment, there was also interference in the sagittal direction. The majority of diagonal trajectories were also influenced by visual anomalies. This is further explained in the next section.

These irregularities were also manifested in the delays, as exemplified by Figure 29. The result was that the peak of the motion of ISAC was not well-defined (it appeared more as a plateau), so it was approximated by the midpoint of the resulting plateau. This explains the approximate delays in Tables 11 – 13.

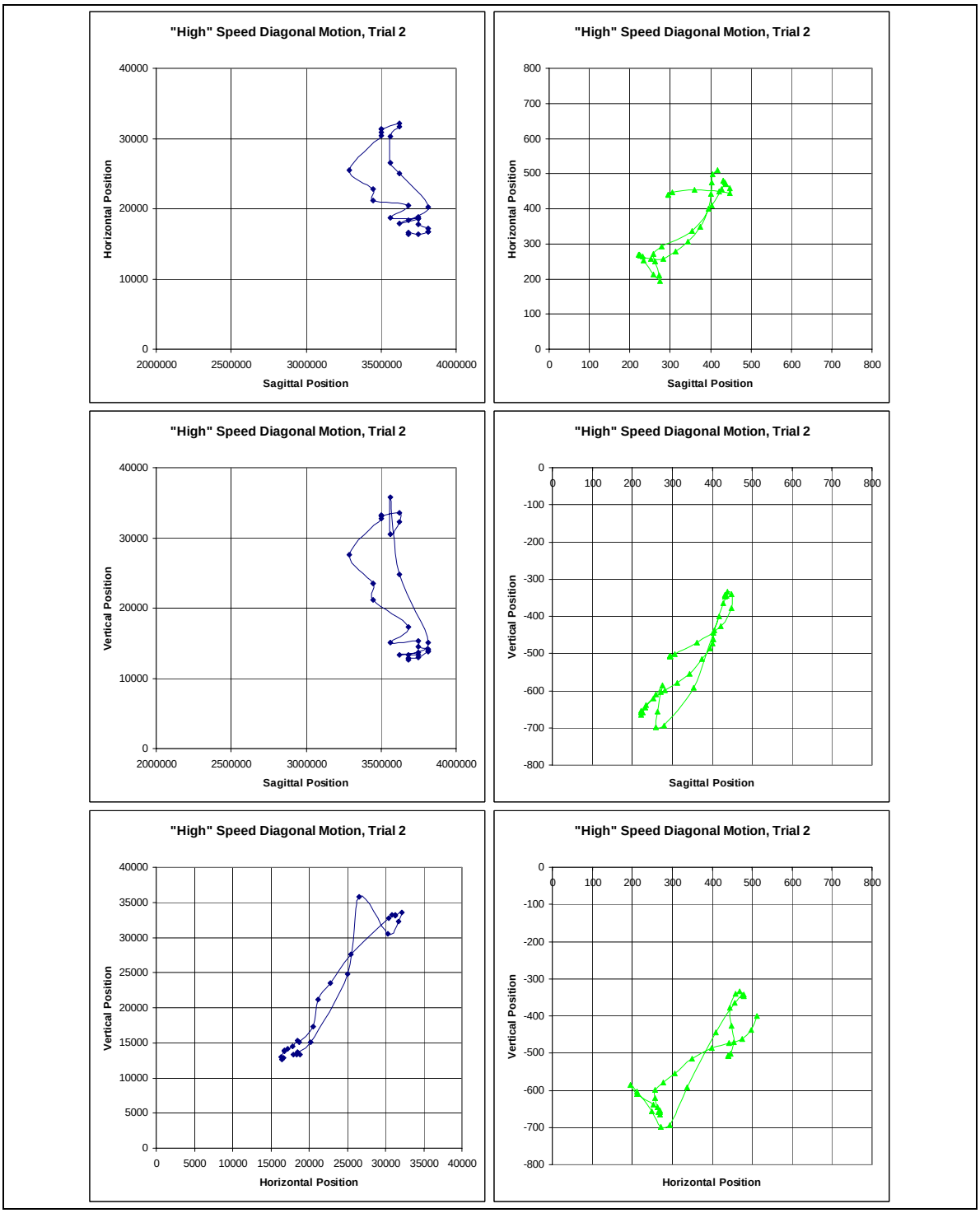


Figure 28: A trial of "low" speed diagonal motion; human motion is shown on the left, that of ISAC on the right

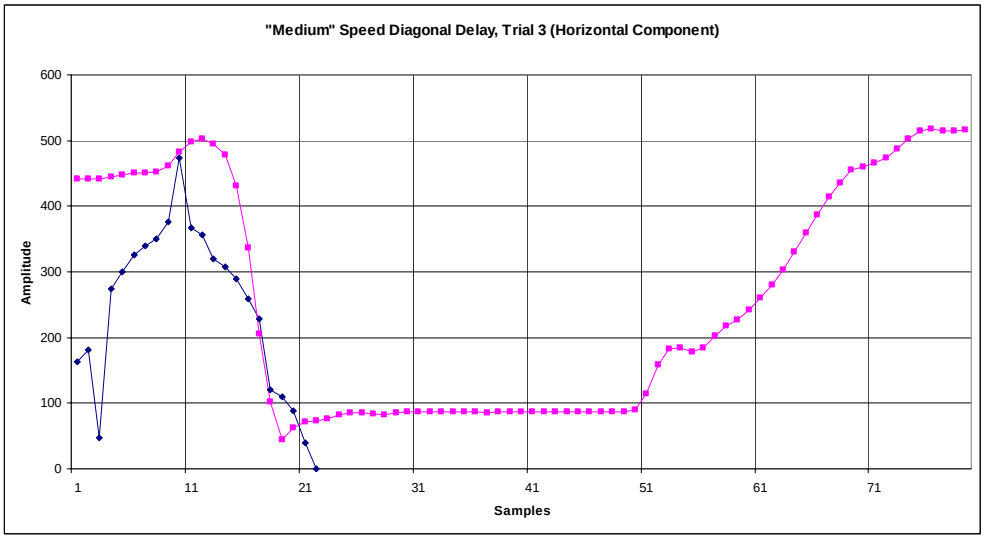


Figure 29: A trial of "medium" speed diagonal motion; the motion of ISAC is the upper curve and that of the human is the lower

IV.3.5. Problems Encountered

Most of the results were quite satisfactory, but there were a few problems encountered, however, which resulted in several minor discrepancies and a few substantial errors. The primary problem, which only had a pronounced effect on the diagonal experiment, was the existence of visual anomalies. Despite the precautions taken to ensure appropriate lighting and limited visual background noise, occasional false positive identifications of the hand of the human to be imitated resulted in miscalculations of the actual position of the hand. This caused the imitation and actuation subsystems to try to direct the hand of ISAC to an incorrect location. Although this is a problem visually, the PID controller of the actuation subsystem dampened out most physical effects of these anomalies; the only trials in which these effects were pronounced were those in which several of these anomalies occurred in sequence. This

happened mostly in the diagonal trials. An example of these anomalies is shown in Figure 30.

Other errors encountered were mentioned previously in this chapter. These included the limitations of the workspace of ISAC, which is so small and irregular that vertical and sagittal motion are almost always coupled, and human error, which occasionally surfaced in the trials of motions that are not natural for a human to make.

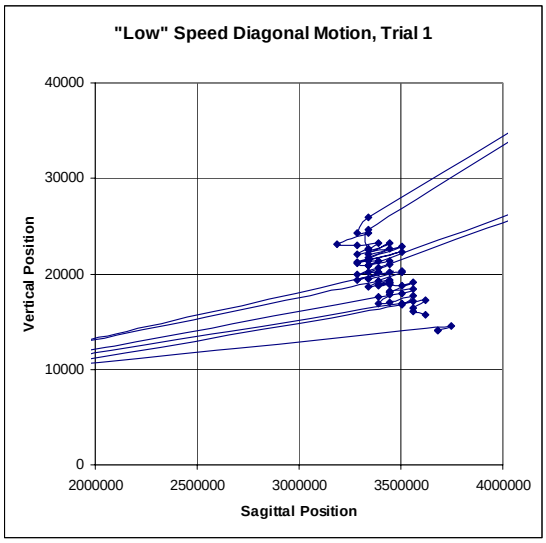


Figure 30: Trajectory of human hand motion showing visual anomalies

Although there were many instances of each of these errors, most did not impact the results in any significant way. Some discussion of dealing with these errors in future experiments is carried out in the following chapter.

CHAPTER V

CONCLUSION

All of the experiments were successful, although a few trials were only marginally so due to errors. The results and the errors are discussed further in section V.1. below, and potential solutions to the problems encountered and other improvements are discussed in section V.2.

V.1. Discussion of Results

The trajectories of the motions shown in Figures 22, 24, 26, and 28 showed that ISAC imitated the human very well, especially in the horizontal direction. The vertical and sagittal directions were also imitated fairly well, but the limitations of the workspace of ISAC caused some coupling between those two directions. The diagonal motion, although satisfactory, did not perform nearly as well as the others. The appearance of visual anomalies, in which the human hand was falsely identified in the wrong location in one or both of the images, substantially affected the results in the diagonal experiment. Although these anomalies also appeared in the other experiments, they did not have much effect because there were not very many; in the diagonal case, however, many of these anomalies appeared in a row, thus misleading ISAC into imitating an imaginary hand.

The delays shown in all of the tables in the previous chapter also showed that the system responded quickly to movements of the human hand. For “low” speeds, all but

one of these delays were well within the 10% limit for real-time imitation, and for "medium" and "high" speeds the delays were well within the 25% limit for near real-time. This limit was only encroached upon by a few of the "high" speed trials, so as long as the human does not wave his or her hand around wildly, ISAC is able to imitate the motion. Although the individual delays varied somewhat from trial to trial, it is clear that the average delay for each motion increased with the speed of the motion.

In view of the positive results shown, it is clear that each of the experiments was a success, and consequently that this system is successful in achieving its aim. ISAC is now able to successfully imitate human hand motions in real-time. This will now be useful in teleoperation for the various grasping or pointing tasks for which ISAC is frequently used, and more importantly, this system provides a foundation upon which imitation learning research and experiments can be built.

V.2. Future Work

Several modifications and future systems have already been hinted at earlier in this document, but more detail will be outlined here.

V.2.1. Modifications

Although this system was a success, there are a number of improvements which could be added to enhance its performance. Among these improvements are a more robust method for detecting hands, a faster Haar detection routine, a method for filtering

visual anomalies, more finely tuned PID parameters, and even a more robust physical platform.

Although the pre-trained Haar cascade provided by [41] was sufficient to this application, it was still quite particular about the lighting, orientation, and configuration of the hand. If the hand was not a perfect ASL "A," or if it was angled even slightly, the Haar detection routine was not able to locate it, resulting in a lost sample (no joint information was transmitted to the actuation subsystem). In order to ensure that no samples are lost, a more robust method for detecting the hand is necessary. Further training of the hand Haar cascade to include these variations would be one solution, or a completely different hand detection method, such as a particle filter or a flock of features approach, might be an even better solution to the problem.

It is also of note that the system, while it responded quickly, still had a delay between the motion of the human and the motion of ISAC. While some delay is inevitable in any physical system, and while the physical structure of ISAC introduces a substantial amount of delay, the responsiveness could still be improved in a number of ways. One of these ways is a faster routine for detecting faces and hands. Although the Haar detection routine was quick, its speed increased tremendously by restricting the image ROI to an area where the face or hand is likely to be detected. If there were a way to further shrink the images or the image ROIs, or if an entirely separate, faster algorithm were developed, this portion of the delay could be substantially reduced. Another way to decrease the delay due to image processing time would be to purchase and install the IPP library discussed in section II.2.

Another problem, perhaps the most serious of all, was the presence of visual anomalies and noise in the vision subsystem. Since ISAC directly imitated the information from the vision subsystem, these anomalies severely impacted the quality of his motions. If a filter were designed to take a moving average of the detected hand positions, or if a threshold function were designed to set the maximum distance between the currently detected hand and the previously detected hand, this problem could be alleviated. Another approach, which would also have the effect of decreasing the delay between the motion of the human and that of ISAC, would be to restrict the image ROI for hand detection to a certain region around the previously detected hand location. This would have the same result as setting a threshold distance from the current detection to the previous detection, with the added benefit of greatly diminishing the delay in the vision subsystem.

Another approach to decreasing the delay in the overall system would be to further tweak the parameters of the PID controller. It was previously mentioned that the controller designed by Ulutas, *et al.* is only a P controller, so adding D and I parameters may improve the performance of the controller. Also, the parameters chosen were somewhat overdamped, so increasing their values or may also result in a faster response.

Lastly, the hardware of ISAC, specifically the McKibben artificial muscles, introduced substantial delay in the overall system. It is quite probable that applying this system to a physical platform with more responsive and controllable actuators would demonstrate significantly lower delay and possibly allow the "high" speed motions performed in Chapter IV to occur in real-time.

V.2.2. Future Systems

The system developed in this document is of considerable importance because it is the first step towards a number of advanced systems which could be developed by the CRL, including teleoperation and imitation learning. Another important future development which would further aid these systems would be the inclusion of a method to imitate hand and head configuration.

Since ISAC is not currently outfitted with an anthropomorphic hand end-effector, this system only imitated the position of the hand. If a more advanced end-effector were added to the arms of ISAC, a system such as HandVu could be incorporated to also recognize the configuration of the hand, thus allowing the end-effector to grasp objects from a number of differing orientations [45]. If other Haar cascades could be trained to detect different facial orientations, such as look-left or look-right, the cameras of ISAC could also follow the same procedures outlined in this system to imitate those motions.

ISAC already has a cognitive architecture whereby it can solve simple problems, such as reaching to or grasping certain objects, and react to certain stimuli, but these motions and reactions must be preprogrammed either by hard-coding them, by physically moving its arms to the correct positions, or by the use of advanced closed-loop control algorithms [46]. By adding this system to that architecture, the arms could be moved in the correct motions without physical manipulation, and these motions could then be stored for later replay by the aforementioned reaching and grasping system.

More complex motions can also now be recorded. If the aforementioned HandVu system were also incorporated, ISAC would now also be able to accomplish assembly,

reconfiguration, or even problem-solving techniques, instead of only simple reaching and grasping tasks. These techniques will be very important to the development of imitation learning.

Lastly, systems can now be developed whereby ISAC can watch a human solve a task, such as retrieving an object from a box, organizing objects according to certain criteria, or solving puzzles, and imitate that behavior to learn the solution. If reasoning methods whereby ISAC could gain knowledge of the goal of the task were incorporated, this system would provide a very natural means for ISAC to attempt learned solutions towards that goal. This and many other possibilities in the realm of imitation learning are now available to ISAC by the application of this system.

APPENDIX A.

PROGRAM CODE

The following pages contain all of the source code for the two programs used in this experiment, imitation.exe and actuation.exe, including all supporting .cpp and .h files peculiar to the CRL or this application.

```
/*************************************************************************
 |imitation.cpp                                                Sean Thornton|
 |-----------------------------------------------------------------------|
 | This program uses ISAC's cameras to observe the face and hand motions |
 | of a human and maps their locations onto ISAC's body.  The necessary  |
 | joint angles are then calculated and transmitted via UDP to the program |
 | actuation.cpp on Octavia.                                             |
 |                                                                       |
 | Portions of this program are based on the pxcdraw1 sample provided by |
 | Imagenation and some code from Sean Begley's masters thesis.          |
 | Special thanks to Juan Wachs of Ben-Gurion University of the Negev for |
 | the trained aGest.xml haar cascade.                                   |
 *************************************************************************/

#include "stdafx.h"
#include <windows.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <commdlg.h>
#include <time.h>
#include "pxc.h"
#include "iframe.h"
#include "camerahead.h"
#include "cv.h"
#include "highgui.h"
#include "PracticalSocket.h"

/*-------------------------< Program Constants >--------------------------*/

// this program uses 8 bit grayscale frames.
#define PIXEL_TYPE PBITS_Y8

// define the pxc libraries used for 32-bit Windows
#define PXC_NAME  "pxc_95.dll"
#define FRAME_NAME  "frame_32.dll"
#define PXC_NT    "pxc_nt.dll"

PXC pxc_library;
FRAMELIB frame_library;

// application name as used in message boxes
const char szAppName[] = "Imitation Processor";

// these constants are used for grabbing images from the PXC devices
const int grab_type = 0;
const int ImageMaxX = 320,  // 768x494 is the cameras' native resolution
          ImageMaxY = 243;  // reduced from 640x486 to fit

// these constants are used in haar detection
const CvScalar BLACK = {0,0,0};
const int FACE = 1,
          HAND = 2;

// these constants define the rough limits of the workspace of ISAC
const double ISAC_X_MIN = 350;
const double ISAC_X_MAX = 450;
const double ISAC_Y_MIN = -800;
const double ISAC_Y_MAX = 500;
const double ISAC_Z_MIN = -400;
const double ISAC_Z_MAX = 300;
const double STEREO_X_MIN = 1300000;
const double STEREO_X_MAX = 4500000;
const double STEREO_Y_MIN = 15000;
const double STEREO_Y_MAX = 35000;
const double STEREO_Z_MIN = 10000;
const double STEREO_Z_MAX = 35000;

const double ANG_L_0_MIN = -58.372;
```

```
const double ANG_L_0_MAX = 14.995;
const double ANG_L_1_MIN = 35.760;
const double ANG_L_1_MAX = 120.25;
const double ANG_L_2_MIN = -210.378;
const double ANG_L_2_MAX = -144.212;
const double ANG_L_4_MIN = -40;
const double ANG_L_4_MAX = 40;

// these constants define certain D-H parameters for ISAC
const double CENTER_TO_ANGLE_0 = 246;
const double SHOULDER_OFFSET = 200;
const double UPPERARM = 325;
const double FOREARM = 290;
const double HAND_L = 250;

// these constants are needed for inverse kinematics
const double PI = 3.14159265359;
const double R2D = (180.0 / PI);


/*-----------------------------< Image Data >-------------------------------*/

//PXC200 frame grabbers
static long frame_grabber_L=0L, frame_grabber_R=0L;
static FRAME   PX_FAR *frame_buffer_L=NULL;
static FRAME   PX_FAR *frame_buffer_R=NULL;
static int initflags=0;

//OpenCV
IplImage *imageL = 0,
         *imageR = 0;   // the images for face/hand detection

//Haar detection
static CvMemStorage *storage = 0;
static CvHaarClassifierCascade *hand_cascade = 0, *face_cascade = 0;
const char *hand_cascade_name = "C:/Bob/ehci-0.4/samples/data/aGest.xml";
const char *face_cascade_name =
          "I:/Etc/OpenCV/data/haarcascades/haarcascade_frontalface_alt2.xml";

//Pan-tilt units
CameraHead hd;
CameraHead *hdp = &hd;
double headAngles[4] = {-2, 0, 2, 0};


/*--------------------------< UDP Communcation >----------------------------*/

UDPSocket sock;
const int MAXLENGTH = 4096;
const string ARMCONTROLADDRESS = "129.59.72.55";
const unsigned short ARMCONTROLPORT = 3999;
string angleinfo;


/*---------------------< Files for Experimental Data >---------------------*/

const char *filename = "C:/Bob/imitation/data/handdata.txt";
FILE *coordinateFile;


/*-------------------------< Program Functions >---------------------------*/

void IplCopy(IplImage *img, long fg, FRAME   PX_FAR *fb);
CvPoint haarDetect(IplImage *img, int objectType);
void calculateXYZ(CvPoint3D32f *objloc, CvPoint left, CvPoint right);
void mapToISAC(CvPoint3D32f *hand, CvPoint3D32f *face);
double* simpleInverseKinematics(CvPoint3D32f point);
string formatUDP(double *angles);
bool AppInit();
```

```
void AppExit();


/*****************************************************************************
 * Name:         main                                                        *
 * Description:  Main window program and message processing loop.            *
 *****************************************************************************/
int main(int argc, char *argv[])
{
    bool err = 0;
    char c = 0;
    char motionComplete[MAXLENGTH];
    string srcAddr;
    unsigned short srcPort;
    CvPoint leftPoint, rightPoint;  // the centers of the detected objects
    CvPoint3D32f faceLoc, handLoc;  // the 3D locations of the face/hand
    double *armAngles = new double[6];

    // call initialization procedure
    printf("Initializing system...\n");
    if(!AppInit())
    {
        err = 1;
    }
    else
    {
        // detect the face and wait for user approval
        printf("\nSearching for face.  Please press 'f' when located...");

        // while 'f' has not been pressed
        while(c != 102)
        {
            // acquire images
            IplCopy(imageL, frame_grabber_L, frame_buffer_L);
            IplCopy(imageR, frame_grabber_R, frame_buffer_R);

            // detect and locate the face in the images
            leftPoint = haarDetect(imageL, FACE);
            rightPoint = haarDetect(imageR, FACE);

            // draw a circle around the face in each image
            cvCircle(imageL,leftPoint,20,BLACK,3,8);
            cvCircle(imageR,rightPoint,20,BLACK,3,8);

            // display the image for the user to see
            cvShowImage("Left Eye", imageL);
            cvShowImage("Right Eye", imageR);

            // wait 2 seconds to give the user a chance to press 'f'
            // if the face is found to be acceptable
            c = cvWaitKey(1000);
        }

        // calculate the 3D location of the face
        calculateXYZ(&faceLoc, leftPoint, rightPoint);

        // transmit the home position angles to the arm controller
        armAngles[0] = 0;
        armAngles[1] = 90;
        armAngles[2] = -180;
        angleinfo = formatUDP(armAngles);
        sock.sendTo(angleinfo.c str(), strlen(angleinfo.c_str()),
                    ARMCONTROLADDRESS, ARMCONTROLPORT);

        // Run the main hand detection loop and wait for ESC key
        printf("\n\nDetecting hand... (ESC to quit)\n");

        for(;;)
        {
```

```
            // if the user is clicking the mouse, let him/her have priority
            if (GetKeyState(VK_LBUTTON) >= 0)
            {
                // acquire images
                IplCopy(imageL, frame_grabber_L, frame_buffer_L);
                IplCopy(imageR, frame_grabber_R, frame_buffer_R);

                // detect and locate the hand in the images
                leftPoint = haarDetect(imageL, HAND);
                rightPoint = haarDetect(imageR, HAND);

                // only imitate if a hand is detected in both images
                if(leftPoint.x && leftPoint.y && rightPoint.x &&rightPoint.y)
                {
                    calculateXYZ(&handLoc, leftPoint, rightPoint);
                    if(handLoc.x > 0 && handLoc.y > 0 && handLoc.z > 0)
                    {
                        // write hand position to file for comparing results
                        fprintf(coordinateFile,"%f\t%f\t%f\n",handLoc.x,
                                handLoc.y,handLoc.z);

                        // wait for a go-ahead signal from controller
                        sock.recvFrom(motionComplete, MAXLENGTH, srcAddr,
                                      srcPort);

                        // map the location of hand to joint angles for ISAC
                        mapToISAC(&handLoc, &faceLoc);
                        armAngles = simpleInverseKinematics(handLoc);

                        // transmit the angles to the arm controller
                        angleinfo = formatUDP(armAngles);
                        sock.sendTo(angleinfo.c_str(),
                                    strlen(angleinfo.c_str()),
                                    ARMCONTROLADDRESS, ARMCONTROLPORT);
                    }
                }
                // otherwise, just stay in the same position
                else
                {
                    sock.recvFrom(motionComplete, MAXLENGTH, srcAddr,
                                  srcPort);
                    sock.sendTo(angleinfo.c_str(), strlen(angleinfo.c_str()),
                                ARMCONTROLADDRESS, ARMCONTROLPORT);
                }

                // draw a circle around the hand
                cvCircle(imageL,leftPoint,20,BLACK,3,8);
                cvCircle(imageR,rightPoint,20,BLACK,3,8);

                // display the image
                cvShowImage("Left Eye", imageL);
                cvShowImage("Right Eye", imageR);
            }

            // if the [ESC] key is pressed, exit the imitation loop
            c = cvWaitKey(1);
            if (c == 27)
            {
                break;
            }
        }
    }

    // shutdown the application
    AppExit();
    return err;
}
```

```
/*****************************************************************************
 * Name:         IplCopy                                                     *
 * Description:  Grabs image from frame grabber and converts it to           *
 *               IplImage                                                    *
 *****************************************************************************/
void IplCopy(IplImage *img, long fg, FRAME __PX_FAR *fb)
{
    void *buffer_addr = 0;  // the address of the grabbed image data

    // grab the image
    pxc_library.Grab(fg, fb, (short)grab_type);

    // direct the IplImage to the image data in the buffer
    buffer_addr = frame_library.FrameBuffer(fb);
    img->imageData = (char *)buffer_addr;
}


/*****************************************************************************
 * Name:         haarDetect                                                  *
 * Description:  Detects the specified object in an image (based on Sean     *
 *               Begley's / Learning OpenCV's code)                          *
 *****************************************************************************/
CvPoint haarDetect(IplImage *img, int objectType)
{
    int i;
    int area, maxArea = 0;
    CvSeq *haarObjects = 0;
    CvRect imgROI,
           *object,
           maxObject = cvRect(0,0,0,0);
    CvPoint objectCenter;

    // these steps are necessary for accurate detection
    cvEqualizeHist(img, img);
    cvClearMemStorage(storage);

    // select the type of object to be detected
    if (objectType == FACE)
    {
        // set region of interest in img to save processing time
        imgROI.x = img->width / 3;      // right 2/3 horizonally
        imgROI.width = imgROI.x * 2;
        imgROI.y = img->height / 3;     // middle 1/3 vertically
        imgROI.height = imgROI.y;
        cvSetImageROI(img, imgROI);

        // detect the faces in img -- see OpenCV documentation for details on
        // other parameters
        haarObjects = cvHaarDetectObjects(img, face_cascade, storage,
                                          1.1, 2, 0, cvSize(30, 30));
    }
    else if (objectType == HAND)
    {
        imgROI.x = 0;                       // ROI is left 3/4 for hands
        imgROI.width = 3 * img->width / 4;
        imgROI.y = 0;
        imgROI.height = img->height;
        cvSetImageROI(img, imgROI);

        haarObjects = cvHaarDetectObjects(img, hand_cascade, storage,
                                          1.1, 2, 0, cvSize(20, 20));
    }
    else
    {
        // if an illegal object type is requested, return a garbage point
        objectCenter = cvPoint(0,0);
    }
```

```
    // scan through the CvSeq of detected objects and find the largest
    for(i = 0; i < (haarObjects ? haarObjects->total : 0); i++)
    {
        object = (CvRect *)cvGetSeqElem(haarObjects, i);

        area = object->width * object->height;

        // save the current object if it is the largest thus far
        if (area > maxArea)
        {
            maxObject = *object;
        }
    }

    // reset ROI for proper image display
    cvResetImageROI(img);

    // calculate and return the center of the detected object
    objectCenter.x = maxObject.x + maxObject.width / 2 + imgROI.x;
    objectCenter.y = maxObject.y + maxObject.height / 2 + imgROI.y;

    return objectCenter;
}


/*****************************************************************************
 * Name:         calculateXYZ                                                *
 * Description:  Calculates the 3-dimensional location of an object shown  *
 *               in stereo images (based on Sean Begley's code)              *
 *****************************************************************************/
void calculateXYZ(CvPoint3D32f *objloc, CvPoint left, CvPoint right)
{
    double f = 15262.5;         // focal length = 305.25mm * 50px/mm
    double b = 14000;           // base length = 280mm * 50px/mm
    double yri, yli, zri, zli;  // coordinates in each image
    double x = 0;               // distance from ISAC
    double yr, yl, zr, zl;      // actual vertical-horizontal coordinates
    double z = 0, y = 0;

    yri = (double)right.x;      // get pixel coordinates
    yli = (double)left.x;
    zri = (double)right.y;
    zli = (double)left.y;

    x = f * b / (yli - yri);    // calculate x in pixels

    yr = yri * x / f;           // calculate real y and z values in pixels
    yl = yli * x / f;
    zr = zri * x / f;
    zl = zli * x / f;

    y = (yr + yl) / 2;          // average real coordinates from each image
    z = (zl + zr) / 2;

    objloc->x = x;              // write the values into the objloc to be
    objloc->y = y;              // returned
    objloc->z = z;
}


/*****************************************************************************
 * Name:         mapToISAC                                                   *
 * Description:  Map the locations of the face and hand onto ISAC's body   *
 *               Note that in (hand-face)-(limit-face) face cancels out!   *
 *               ...looks like some effort could have been saved earlier   *
 *****************************************************************************/
void mapToISAC(CvPoint3D32f *hand, CvPoint3D32f *face)
{
    // find hand position relative to face and scale/shift to ISAC
```

```
    hand->x = -(hand->x-STEREO X MIN) * ((ISAC X MAX-ISAC X MIN) /
                                         (STEREO X MAX-STEREO_X_MIN))
                                         - ISAC X MIN;
    hand->y = -(hand->y-STEREO Y MIN) * ((ISAC Y MAX-ISAC Y MIN) /
                                         (STEREO Y MAX-STEREO_Y_MIN))
                                         - ISAC Y MIN;
    hand->z = -(hand->z-STEREO Z MIN) * ((ISAC Z MAX-ISAC Z MIN) /
                                         (STEREO Z MAX-STEREO_Z_MIN))
                                         - ISAC_Z_MIN;

    // if the hand location exceeds ISAC's workspace, cap it off
    if (hand->x > ISAC X MAX)
        hand->x = ISAC X MAX;
    if (hand->x < ISAC X MIN)
        hand->x = ISAC X MIN;
    if (hand->y > ISAC Y MAX)
        hand->y = ISAC Y MAX;
    if (hand->y < ISAC Y MIN)
        hand->y = ISAC Y MIN;
    if (hand->z > ISAC Z MAX)
        hand->z = ISAC Z MAX;
    if (hand->z < ISAC Z MIN)
        hand->z = ISAC_Z_MIN;
}


/*****************************************************************************
 * Name:         simpleInverseKinematics                                     *
 * Description:  This is a trimmed version of Sean Begley's function used  *
 *               to calculate the joint angles needed to reach a given     *
 *               point                                                       *
 *****************************************************************************/
double* simpleInverseKinematics(CvPoint3D32f point)
{
    //T = Theta = Angle
    //X, Y, & Z are relative to ISAC's workspace thus:
    //    X = front to back, Y = right to left, Z = top to bottom
    double T0, T1, T2, T4;              // angles 0, 1, 2, & 4
    double a, b, u, f, h;               // holders for constant values
    double Ta, Tb, v1, v2;              // intermediate angle calculations
    double n, o;                        // intermediate values for T0
    double c2;                          // intermediate values for T1 & T2
    double x, y, z;                     // absoulte values of incoming point
    double *ret = new double[6];        // return array for angles
    double alpha, beta;

    //set constant holders
    a = CENTER TO ANGLE 0;
    b = SHOULDER OFFSET;
    u = UPPERARM;
    f = FOREARM;
    h = HAND_L;

    //adjust x, y, z for math
    x = point.x;
    y = point.y;
    z = -point.z;

    //Calculate Theta 0 (shoulder X Y)
    if (y >= a)
    {
        Ta = atan(x/(y - a));
        Tb = acos(b*sin(Ta)/x);

        T0 = Tb - Ta;
    }
    else
    {
        n  = a - y;
```

```
        Tb = atan(x/n);
        o  = x/sin(Tb);
        Ta = acos(b/o);
        T0 = -(PI - Ta - Tb);
    }

    //Calculate Theta 1 (elbow) & Theta 2 (shoulder X Z)
    //**** current assumption that 0 on the Z axis is at shoulder level
    //geometric solution
    alpha = atan2(z,x);
    beta = acos((x*x+z*z+u*u-(f+h)*(f+h))/(2*u*sqrt(x*x+z*z)));

    c2 = (x*x + z*z - u*u - (f+h))/(2*u*(f+h));

    //angles according to the book
    v1 = alpha + beta;
    v2 = acos(c2);

    //conversion to ISAC's frame
    T1 = v1;
    T2 = -v2 - PI/2;

    //Calculate Theta 4 (wrist up/down)
    T4 = -(PI + T1 + T2);

    //Convert to Degrees
    T0=T0*R2D;
    T1=T1*R2D;
    T2=T2*R2D;
    T4=T4*R2D;

    //Ensure that final Theta values do not exceed ISAC's Range of Motion
    if(T0 > ANG_L_0_MAX) T0 = ANG_L_0_MAX;
    if(T0 < ANG_L_0_MIN) T0 = ANG_L_0_MIN;
    if(T1 > ANG_L_1_MAX) T1 = ANG_L_1_MAX;
    if(T1 < ANG_L_1_MIN) T1 = ANG_L_1_MIN;
    if(T2 > ANG_L_2_MAX) T2 = ANG_L_2_MAX;
    if(T2 < ANG_L_2_MIN) T2 = ANG_L_2_MIN;
    if(T4 > ANG_L_4_MAX) T4 = ANG_L_4_MAX;
    if(T4 < ANG_L_4_MIN) T4 = ANG_L_4_MIN;

    //populate return array in degrees
    ret[0]=T0;
    ret[1]=T1;
    ret[2]=T2;
    ret[3]=0;
    ret[4]=T4;
    ret[5]=0;

    return ret;
}


/*****************************************************************************
 * Name:         formatUDP                                                   *
 * Description:  Converts the given angle data into a string transmittable *
 *               via the UDP socket                                          *
 *****************************************************************************/
string formatUDP(double *angles)
{
    string UDPstring;
    char numbuffer[33];

    // initialize the string,
    UDPstring = "";
    // append string versions of each angle,
    sprintf(numbuffer,"%.0f",angles[0]);
    UDPstring.append(numbuffer);
    // and delimit with a tab
```

```
    UDPstring.append("\t");
    sprintf(numbuffer,"%.0f",angles[1]);
    UDPstring.append(numbuffer);
    UDPstring.append("\t");
    sprintf(numbuffer,"%.0f",angles[2]);
    UDPstring.append(numbuffer);

    return UDPstring;
}


/*************************************************************************
 * Name:         AppInit                                                 *
 * Description:  Initialize all libraries and allocate all data needed   *
 *************************************************************************/
bool AppInit()
{
    // initialize the libraries
    printf("\tlibraries...");
    if(!imagenation_OpenLibrary(PXC_NAME,&pxc_library,sizeof(pxc_library)))
    {
        if(!imagenation_OpenLibrary(PXC_NT,&pxc_library,sizeof(pxc_library)))
        {
            MessageBox(0, "Unable to load PXC dll files.  Please verify that"
                          PXC_NAME" and "PXC_NT" are correctly installed.",
                          szAppName, MB_OK);
            return false;
        }
    }
    initflags|=1;

    if(!imagenation_OpenLibrary(FRAME_NAME,&frame_library,
                                sizeof(frame_library)))
    {
        MessageBox(0, "Unable to load "FRAME_NAME
                      ".  Please verify that this file is correctly"
                      " installed.", szAppName, MB_OK);
        return false;
    }
    initflags|=2;
    printf("done!\n");

    // allocate frame grabbers
    printf("\tframe grabbers...");
    frame_grabber_L = pxc_library.AllocateFG(0);
    frame_grabber_R = pxc_library.AllocateFG(1);
    if(!frame_grabber_L || !frame_grabber_R)
    {
        MessageBox(0, "Unable to allocate frame grabbers. Please verify that"
                      " they are not in use by another process.",
                      szAppName, MB_OK);
        return false;
    }

    pxc_library.SetWidth(frame_grabber_L,(short)ImageMaxX);
    pxc_library.SetHeight(frame_grabber_L,(short)ImageMaxY);
    pxc_library.SetLeft(frame_grabber_L,0);
    pxc_library.SetTop(frame_grabber_L,0);
    pxc_library.SetXResolution(frame_grabber_L,(short)ImageMaxX);
    pxc_library.SetYResolution(frame_grabber_L,(short)ImageMaxY);

    pxc_library.SetWidth(frame_grabber_R,(short)ImageMaxX);
    pxc_library.SetHeight(frame_grabber_R,(short)ImageMaxY);
    pxc_library.SetLeft(frame_grabber_R,0);
    pxc_library.SetTop(frame_grabber_R,0);
    pxc_library.SetXResolution(frame_grabber_R,(short)ImageMaxX);
    pxc_library.SetYResolution(frame_grabber_R,(short)ImageMaxY);
    printf("done!\n");
```

```
    // allocate frame buffers
    printf("\tframe buffers...");
    frame_buffer_L = pxc_library.AllocateBuffer((short)ImageMaxX,
                                                (short)ImageMaxY,
                                                PIXEL_TYPE);
    frame_buffer_R = pxc_library.AllocateBuffer((short)ImageMaxX,
                                                (short)ImageMaxY,
                                                PIXEL_TYPE);
    if(!frame_buffer_L || !frame_buffer_R)
    {
        MessageBox(0, "Unable to allocate frame buffers.", szAppName, MB_OK);
        return false;
    }
    printf("done!\n");

    // create OpenCV image windows
    printf("\tOpenCV data...");
    cvNamedWindow("Left Eye", 1);
    cvMoveWindow("Left Eye", 300, 450);
    cvNamedWindow("Right Eye", 1);
    cvMoveWindow("Right Eye", 633, 450);

    // initialize the IplImages to the correct size for the grabbed frames
    imageL = cvCreateImage(cvSize(pxc_library.GetWidth(frame_grabber_L),
                                  pxc_library.GetHeight(frame_grabber_L)),
                                  IPL_DEPTH_8U, 1);
    imageR = cvCreateImage(cvSize(pxc_library.GetWidth(frame_grabber_R),
                                  pxc_library.GetHeight(frame_grabber_R)),
                                  IPL_DEPTH_8U, 1);

    // initialize the haar cascade and memory storage for detected faces
    // note that (cvErode is a dummy function call to cv.lib--a silly fix
    // to a glitch in OpenCV)
    cvErode(imageL, imageL, NULL, 1);
    hand_cascade = (CvHaarClassifierCascade *)cvLoad(hand_cascade_name,
                                                     0, 0, 0);
    face_cascade = (CvHaarClassifierCascade *)cvLoad(face_cascade_name,
                                                     0, 0, 0);
    storage = cvCreateMemStorage(0);
    if(!hand_cascade || !face_cascade)
    {
        MessageBox(0, "Unable to load Haar cascade files.", szAppName,MB_OK);
        return false;
    }
    printf("done!\n");

    // initialize the pan-tilt units
    printf("\tpan-tilt units...");
    if(!hd.Initialize())
    {
        MessageBox(0,"Unable to initialize pan-tilt units.",szAppName,MB_OK);
        return false;
    }

    hd.MoveHead(headAngles);
    printf("done!\n");

    // initialize the file for experimental result data
    printf("\texperimental data file...");
    if(!(coordinateFile = fopen(filename, "w")))
    {
        MessageBox(0, "Unable to open data file.", szAppName, MB_OK);
        return false;
    }
    printf("done!\n");

    return true;
}
```

```
/*****************************************************************************
 * Name:         AppExit                                                     *
 * Description:  Frees all allocated memory before exiting                   *
 *****************************************************************************/
void AppExit()
{
    // shut down arm controller
    angleinfo = "0 0 9999";
    sock.sendTo(angleinfo.c_str(), strlen(angleinfo.c_str()),
                ARMCONTROLADDRESS, ARMCONTROLPORT);

    // release all pxc memory and libraries
    if(frame_buffer_L)
        frame_library.FreeFrame(frame_buffer_L);
    if(frame_buffer_R)
        frame_library.FreeFrame(frame_buffer_R);
    if(frame_grabber_L)
        pxc_library.FreeFG(frame_grabber_L);
    if(frame_grabber_R)
        pxc_library.FreeFG(frame_grabber_R);
    if (initflags&1)
        imagenation_CloseLibrary(&pxc_library);
    if (initflags&2)
        imagenation_CloseLibrary(&frame_library);
    if (coordinateFile)
        fclose(coordinateFile);

    // reset the head location
    for (int i = 0; i < 4; i++)
        headAngles[i] = 0;
    hd.MoveHead(headAngles);

    // release OpenCV memory
    cvReleaseImage(&imageL);
    cvReleaseImage(&imageR);
    cvDestroyWindow("Left Eye");
    cvDestroyWindow("Right Eye");

    printf("\n");
}
```

```
/***CameraHead.h***/

//#include "stdafx.h"
#include "windows.h"
#include "atlbase.h"
#include "commport.h"
#include "string.h"
#include "stdio.h"
//#include "iostream.h"
#include <iostream>
#include "conio.h"
#include <math.h>
#include "cv.h"     //image stuff
////////////////////////////////////////////////////////////////////////
// CCatch
class CameraHead
{
public:

    /**************************************************************************
******

    Name:           CCatch Component Constructor

    **************************************************************************
******/

    CameraHead() :
            // *** Note: If you want to change the command packet layout, the
n
            // you need to change these next 4 constants!

                LEFT_PAN(0),            // index of the left pan motor in the
 arrays
                LEFT_TILT(1),           // index of the left tilt motor in th
e arrays
                RIGHT_PAN(2),           // index of the right pan motor in th
e arrays
                RIGHT_TILT(3),          // index of the right tilt motor in t
he arrays

                NUM_AXES(6),            // # axes on the left/right pan-tilt
units currently.
//              NUM_AXES_ALL(6),            // # axes on the Left/right pan-t
ilt units and overall pan-tilt.

                COMMAND_PAN_ABS("pp"),  // string which indicates you are sen
ding a command to the pan motor.
                COMMAND_TILT_ABS("tp"), // string which indicates you are sen
ding a command to the tilt motor.
                COMMAND_PAN_REL("po"),  // string which commands pan relative
                COMMAND_TILT_REL("to"), // string which commands tilt relativ
e

                QUERY_PAN("pp"),        // get the pan position
                QUERY_TILT("tp"),       // get the tilt position
                QUERY_MIN_PAN("pn"),    // get min pan limit
                QUERY_MAX_PAN("px"),    // get max pan limit
                QUERY_MIN_TILT("tn"),   // get min tilt limit
                QUERY_MAX_TILT("tx"),   // get max tilt limit
                QUERY_PAN_SPEED("ps"),  // get pan speed
                QUERY_TILT_SPEED("ts"), // get tilt speed
                QUERY_PAN_ACCEL("pa"),  // get pan acceleration
                QUERY_TILT_ACCEL("ta"), // get tilt acceleration

                SET_PAN_ACCEL("pa"),    // set pan acceleration
                SET_TILT_ACCEL("ta"),   // set tilt acceleration
                SET_PAN_SPEED("ps"),    // set pan speed
                SET_TILT_SPEED("ts"),   // set tilt speed
```

```
                HALT ALL("h"),          // stop all pan-tilt unit
                RESET("r"),             // reset the pan-tilt unit
                RESTORE DEFAULTS("df"), // restore factory default settings
                DISABLE_LIMITS("ld"),   // disable limits on DP head.

                SYNC DP("a"),           // allow last DP command to finish
                TERSE FEEDBACK("ft"),   // no wordy responses, Newman.
                DISABLE_ECHO("ed"),     // don't echo commands we send.

                // DP PAN
                MOTOR RANGE ANGLES PAN(318.0f), // pan angle range
                MOTOR_RANGE_PULSES_PAN_LEFT(6184.0),  // pulses in left servo
's range
                MOTOR_RANGE_PULSES_PAN_RIGHT(6178.0), // pulses in right serv
o's range
                PULSE_TO_ANG_PAN_LEFT((double) (MOTOR_RANGE_ANGLES_PAN/MOTOR_
RANGE PULSES PAN LEFT)),
                PULSE_TO_ANG_PAN_RIGHT((double) (MOTOR_RANGE_ANGLES_PAN/MOTOR
_RANGE_PULSES_PAN_RIGHT)),

                // DP Tilt
                MOTOR_RANGE_ANGLES_TILT(106.0f),    // tilt angle range
                MOTOR_RANGE_PULSES_TILT(2061.0f),   // pulses in servo's rang
e
                PULSE_TO_ANG_TILT((double) (MOTOR_RANGE_ANGLES_TILT/MOTOR_RAN
GE_PULSES_TILT))


    {

        // TODO: Initialize all variables
        m_pdAxisGain = NULL;
        m_pdAxisOffset = NULL;
        m_plAxisThreshold = NULL;

        m_plCurrentSample = NULL;
        m_pdLinkCurrentHeadCommand = NULL;
        m_pdLinkCurrentHeadSample = NULL;
    //  m_pdLinkOldHeadCommand = NULL;
//      m_pdLinkOldHeadSample = NULL;


        pCommLEFT = NULL;
        pCommRIGHT = NULL;
        m_psOut = NULL;

        m_pdBuffer = NULL;
        m_bFirstCue = false;

        m_iFlagCue = 0;

        m_pdBuff1 = new double[4];
        m_pdBuff2 = new double[4];
        m_pdBuff3 = new double[4];

        for (int i=0; i<4; i++)
        {
            m_pdBuff1[i] = 0;
            m_pdBuff2[i] = 1;
            m_pdBuff3[i] = 2;
        }


        // setup class variables to appropriate sizes
        if ( NUM_AXES > 0 )
        {
            m_psOut = new long[NUM_AXES];
```

```
            m_pdAxisGain =  new double[ NUM_AXES ];
            m_pdAxisOffset = new double[ NUM_AXES ];
            m_plAxisThreshold = new long[ NUM_AXES ];

            m_pdLinkCurrentHeadCommand = new double[ NUM_AXES ];

            // new head will display 2 more values: representing a
            // overall pan/tilt value for both cameras
            m_pdLinkCurrentHeadSample = new double[ NUM_AXES];
            m_plCurrentSample = new long[ NUM_AXES ];

            // Two missing pointers were noticed not having been allocated
            // It is done here:
//          m_pdLinkOldHeadCommand = new double[ NUM_AXES];
//          m_pdLinkOldHeadSample = new double[ NUM_AXES];
        }


        // initialize axis gains and offsets

        for ( int j = 0; j < NUM_AXES; j++ )
        {
            m_pdAxisGain[j] = 1.0;
            m_pdAxisOffset[j] = 0.0;
            m_plAxisThreshold[j] = 0;
        }

        // setup default values for motors and communications parameters
        m_plCurrentSample[ LEFT_PAN  ] = 0;
        m_plCurrentSample[ LEFT_TILT ] = 0;
        m_plCurrentSample[ RIGHT_PAN ] = 0;
        m_plCurrentSample[ RIGHT_TILT] = 0;

        m_bsDevFilename1 = "\\COM1";
        m_bsDevFilename2 = "\\COM2";
        m_bsCParam1 = "";
        m_bsCParam2 = "";
        m_lCTimeout1 = 0;
        m_lCTimeout2 = 0;

    }


    ~CameraHead()
    {
        /*********** Beginning CCatch::~CCatch() **********/
            if ( m_psOut != NULL )
            delete[] m_psOut;

        m_psOut = NULL;


        ATLTRACE("*** Deleting Commports ....");
        if ( pCommLEFT != NULL )
            delete pCommLEFT;
        pCommLEFT = NULL;
        ATLTRACE(" ... PT Left destroyed,");

        if ( pCommRIGHT != NULL )
            delete pCommRIGHT;
        pCommRIGHT = NULL;
        ATLTRACE(" PT Right destroyed ...\n");

        if ( m_pdAxisGain != NULL )
            delete[] m_pdAxisGain;
        m_pdAxisGain = NULL;

        if ( m_pdAxisOffset != NULL )
            delete[] m_pdAxisOffset;
```

```
        m_pdAxisOffset = NULL;

        if ( m_plAxisThreshold != NULL )
            delete[] m_plAxisThreshold;
        m_plAxisThreshold = NULL;

        if ( m_plCurrentSample != NULL )
            delete[] m_plCurrentSample;
        m_plCurrentSample = NULL;

        /********** CCatch::~CCatch() Deleted Current Sample**********/


        if ( m_pdLinkCurrentHeadCommand != NULL )
            delete[] m_pdLinkCurrentHeadCommand;
        m_pdLinkCurrentHeadCommand = NULL;

        /********** CCatch::~CCatch() Deleted Link Head Command**********/


    //  ATLTRACE("********** CCatch::~CCatch() Deleted Link Head Command****
******\n");


        if ( m_pdLinkCurrentHeadSample != NULL )
            delete[] m_pdLinkCurrentHeadSample;
        m_pdLinkCurrentHeadSample = NULL;


        /********** Finishing CCatch::~CCatch() **********/
    //  ATLTRACE("********** Finishing CCatch::~CCatch() **********\n");


    }


public:
//  double m_dPreviousPosition[2];  // for pan and tilt velocity
    long LeftPanSpeed;
    long RightPanSpeed;
    long LeftPanAccel;
    long RightPanAccel;
    long LeftTiltSpeed;
    long RightTiltSpeed;
    long LeftTiltAccel;
    long RightTiltAccel;
//  long VergeLimit( long Val );

public:
    /** conversion function ***/
    BOOL HeadAngles2Positions( double* pVal, long length );
    BOOL HeadPositions2Angles( double* pVal, long Length );
    int ParseStringToInteger( char* pStr );

    /** internal command **/
    BOOL CommandHeadAbsolute(double* Val, long Length);
    BOOL SampleHead(double* Val, long Length);
//  BOOL CommandHeadRelative(double* Val, long Length);
    bool CalculateOverallPanTilt(double* Val);
    BOOL DPSample();
    BOOL ExceedsThreshold( long, long );
    BOOL SendToLeft( char* pStr );
    BOOL SendToRight( char* pStr );

    /** auxillary command **/
    bool IndexIsValid( long Index );

    /** properties **/
    BOOL get_AxisThreshold(long Index, long* pVal);
```

```
    BOOL put_AxisThreshold(long Index, long newVal);
    BOOL get_LeftTiltSpeed(long *pVal);
    BOOL get_RightTiltSpeed(long *pVal);
    BOOL put_LeftTiltSpeed(long newVal);
    BOOL put_RightTiltSpeed(long newVal);
    BOOL get_LeftPanSpeed(long *pVal);
    BOOL get_RightPanSpeed(long *pVal);
    BOOL put_LeftPanSpeed(long newVal);
    BOOL put_RightPanSpeed(long newVal);
    BOOL get_LeftTiltAccel(long *pVal);
    BOOL get_RightTiltAccel(long *pVal);
    BOOL put_LeftTiltAccel(long newVal);
    BOOL put_RightTiltAccel(long newVal);
    BOOL get_LeftPanAccel(long *pVal);
    BOOL get_RightPanAccel(long *pVal);
    BOOL put_LeftPanAccel(long newVal);
    BOOL put_RightPanAccel(long newVal);
//  BOOL get_TiltLimit(long Length, long *pVal);
//  BOOL get_PanLimit(long Length, long *pVal);
//  BOOL get_CommReadTimeout2(long *pVal);
//  BOOL put_CommReadTimeout2(long newVal);
//  BOOL get_CommReadTimeout1(long *pVal);
//  BOOL put_CommReadTimeout1(long newVal);
//  BOOL get_AxisGain(long Index, double *pVal);
//  BOOL put_AxisGain(long Index, double newVal);
//  BOOL get_AxisOffset(long Index, double *pVal);
//  BOOL put_AxisOffset(long Index, double newVal);
//  BOOL get_CommParams2(char *pVal);
//  BOOL put_CommParams2(char newVal);
//  BOOL get_CommParams1(char *pVal);
//  BOOL put_CommParams1(char newVal);
//  BOOL get_DeviceFilename2(char *pVal);
//  BOOL put_DeviceFilename2(char newVal);
//  BOOL get_DeviceFilename1(char *pVal);
//  BOOL put_DeviceFilename1(char newVal);

public:
    /** general interface command **/
    BOOL Reset();
    BOOL Stop();
    BOOL Initialize();
    void MoveHead(double *command);
    void GetHeadPositions(double *val);
    void Home();


//private:

    // constants useful for this class. (OK--kat)
    // DP Pan
    const double MOTOR_RANGE_ANGLES_PAN;        // pan angle range
    const double MOTOR_RANGE_PULSES_PAN_LEFT;   // pulses in left servo's ran
ge
    const double MOTOR_RANGE_PULSES_PAN_RIGHT;  // pulses in right servo's ra
nge
    const double PULSE_TO_ANG_PAN_LEFT;
    const double PULSE_TO_ANG_PAN_RIGHT;

    // DP Tilt
    const double MOTOR_RANGE_ANGLES_TILT;   // tilt angle range
    const double MOTOR_RANGE_PULSES_TILT;   // pulses in servo's range
    const double PULSE_TO_ANG_TILT;

    const long      NUM_AXES;               // how many axes does this pantil
t have?

    const char*     COMMAND_TILT_REL;       // command for relative tilt moti
on
    const char*     COMMAND_PAN_REL;        // command for relative pan motio
```

```
n
    const char*     COMMAND_TILT_ABS;       // command for absolute tilt moti
on
    const char*     COMMAND_PAN_ABS;        // command for absolute pan motio
n

    const long      LEFT_PAN;
    const long      LEFT_TILT;
    const long      RIGHT_PAN;
    const long      RIGHT_TILT;

    const char*     QUERY_TILT;             // get the tilt position
    const char*     QUERY_PAN;              // get the pan position

    const char*     QUERY_MIN_PAN;          // get min pan limit
    const char*     QUERY_MAX_PAN;          // get max pan limit
    const char*     QUERY_MIN_TILT;         // get min tilt limit
    const char*     QUERY_MAX_TILT;         // get max tilt limit
    const char*     QUERY_PAN_SPEED;        // get pan speed
    const char*     QUERY_PAN_ACCEL;        // get pan acceleration
    const char*     QUERY_TILT_SPEED;       // get tilt speed
    const char*     QUERY_TILT_ACCEL;       // get tilt acceleration

    const char*     SET_PAN_ACCEL;          // set pan acceleration
    const char*     SET_PAN_SPEED;          // set pan speed
    const char*     SET_TILT_ACCEL;         // set tilt acceleration
    const char*     SET_TILT_SPEED;         // set tilt speed

    const char*     HALT_ALL;               // stop the pan-tilt unit

    const char*     RESET;                  // reset the pan-tilt unit
    const char*     RESTORE_DEFAULTS;       // restore factory default settin
gs

    const char*     DISABLE_LIMITS;         // disable limits on DP head.
    const char*     DISABLE_ECHO;           // don't echo commands we send.

    const char*     SYNC_DP;                // allow last DP command to finis
h

    const char*     TERSE_FEEDBACK;         // no wordy responses, Newman.

    // COMMPORT parameters
    CComBSTR m_bsDevFilename2;              // Device filenames for 1 & 2
    CComBSTR m_bsDevFilename1;
    CComBSTR m_bsCParam2;                   // Device parameters for 1 & 2
    CComBSTR m_bsCParam1;

    long    m_lCTimeout1;                   // Read timeouts for 1 & 2
    long    m_lCTimeout2;

    double* m_pdAxisGain;                   // Axis gains
    double* m_pdAxisOffset;                 // Axis offsets

    double dTime;                           // Time flag

    double* m_pdLinkCurrentHeadCommand;     // most recent commands you got f
rom vec sig (for new head)
    double* m_pdLinkCurrentHeadSample;      // most recent sample you sent to
 vec sig (for new head)
    long*   m_plAxisThreshold;              // each axis' threshold value. if
                                            // new command - old command <
                                            // threshold, the new_command wil
l not be sent to that axis
 // for the old signal vector 09/10/03
//  double* m_pdLinkOldHeadCommand;     // most recent commands for the old h
ead
//  double* m_pdLinkOldHeadSample;      // most recent sample for the old hea
d
```

```
    long*           m_plCurrentSample;
    long*           m_psOut;                // temp value

    CCommPort*      pCommLEFT;              // COM1 for Left pan/tilt control
    CCommPort*      pCommRIGHT;             // COM2 for right pan/tilt contro
l

    bool m_bFirstCue;           // Will be used as a flag to avg. sound Cues
    double* m_pdBuffer;         // Stores data to be averaged later

    int m_iFlagCue;             // Flag for median calculation
    double* m_pdBuff1;          // Buffer to calculate median from three cues
.
    double* m_pdBuff2;
    double* m_pdBuff3;


};
```

```
/***CameraHead.cpp***/

#include "stdafx.h"

#include "CameraHead.h"
#include <math.h>               //for trig functions
#include <time.h>               //for delay function

//constants
#ifndef PI
#define PI  3.14159
#define R2D (180.0/PI)
#define D2R (PI/180.0)
#endif

BOOL CameraHead::Initialize()
{

    USES_CONVERSION;

    if ( pCommLEFT == NULL )
    {
        // COM1 port
        char* pFN1 = OLE2T(m bsDevFilename1);
        pCommLEFT   = new CCommPort( pFN1, 9600 );  // Capture COM1@9600baud
    }

    if ( pCommRIGHT == NULL )
    {
        // COM2 port.
        char* pFN2 = OLE2T(m bsDevFilename2);
        pCommRIGHT  = new CCommPort( pFN2, 9600 );  // Capture COM2@9600baud
    }

    for ( int axis index = 0; axis_index < NUM_AXES; axis_index++ )
        m_psOut[ axis_index ] = 0;

    // re-initialize pan-tilt unit
    char strOut[35];
    BOOL commResult1 = FALSE;
    BOOL commResult2 = FALSE;

    // it won't hurt to set these commands each time this is called.

    // Enable terse feedback
    sprintf( strOut, "%s ", TERSE_FEEDBACK );

    commResult1 = SendToLeft( strOut );
    commResult2 = SendToRight( strOut );

    if ( commResult1 && commResult2 )
    {
        // Disable DP head limits on startup
        sprintf( strOut, "%s ", DISABLE LIMITS );
        commResult1 = SendToLeft( strOut );
        commResult2 = SendToRight( strOut );

        if ( commResult1 && commResult2 )
        {
            // Disable command echo
            sprintf( strOut, "%s ", DISABLE ECHO );
            commResult1 = SendToLeft( strOut );
            commResult2 = SendToRight( strOut );

            double val[4];

            for ( int i = 0; i < NUM_AXES; i++ )
            {
                val[i] = 0;
```

```
            }

            // set to the home position
            commResult1 = CommandHeadAbsolute(val,NUM_AXES);
        }
        else
        {
            return FALSE;
        }
    }
    else
    {
        return FALSE;
    }

    //set new speed and accel for pan/tilt
    put_LeftPanAccel((long)8000);
    put_RightPanAccel((long)8000);
    put_LeftPanSpeed((long)2500);
    put_RightPanSpeed((long)2500);
    put_LeftTiltAccel((long)8000);
    put_RightTiltAccel((long)8000);
    put_LeftTiltSpeed((long)2500);
    put_RightTiltSpeed((long)2500);

    return TRUE;
}


/*************************************************************************
**

    Name:           CCatch::IndexIsValid()

    Purpose:        Check parameters given by the user.  Parameters must repr
esent
                    a valid axis of the pan-tilt unit.

    Created By:     RMW - watson@vuse.vanderbilt.edu
    Date Modified:  Sunday, August 17, 1997

*************************************************************************
**/

bool
CameraHead::IndexIsValid
    (
    long Index                  // in: index to test for validity
    )
{
    return ( ( Index >= 0 ) && ( Index < NUM_AXES ) );
}


/*************************************************************************
**

    Name:           CCatch::SampleHead()


*************************************************************************
**/

BOOL CameraHead::SampleHead
    (
    double *    Val,                    // out: values representing where the
 pan-tilt motors are now
                                        //  Valid verge range: [MIN_RANGE, MA
X_RANGE]
```

```
    long        Length                  // in: how long is the vector
    )
{

    // Get the new pan and tilt positions.
    BOOL result = FALSE;

    result = DPSample();

    if ( !result )
        return FALSE;

    if ( Val != NULL )
    {
        // Fill up the user's structure
            Val[ LEFT_PAN  ] = (double) m_plCurrentSample[ LEFT_PAN ];  // Va
l[0]
            Val[ RIGHT_PAN ] = (double) m_plCurrentSample[ RIGHT_PAN ]; // Va
l[2]
            Val[ LEFT_TILT ] = (double) m_plCurrentSample[ LEFT_TILT ]; // Va
l[1]
            Val[ RIGHT_TILT] = (double) m_plCurrentSample[ RIGHT_TILT ];// Va
l[3]
    }
    else return FALSE;

    return TRUE;
}


/**************************************************************************
**

    Name:           CCatch::CommandHeadAbsolute()

    Purpose:        Method to tell the pan-tilt unit to go to a certain locat
ion.

**************************************************************************
**/

BOOL CameraHead::CommandHeadAbsolute
    (
    double *        Val,                // in: values to send to the pan-tilt
 motors
                                    //  Valid verge range: [MIN_RANGE(0), MAX
 RANGE(255)]
    long        Length              // in: how long is the vector
    )
{

    if ( Val != NULL )
    {
        if ( ( pCommLEFT != NULL ) && ( pCommRIGHT != NULL ) )
        {
            char strOut[35];
            BOOL result = FALSE;

            // calculate values to send out.
            for ( int i = 0; i < NUM_AXES; i++ )
            {
                m psOut[i] = Val[i] * m pdAxisGain[i] + m_pdAxisOffset[i];
                m_plCurrentSample[i] = Val[i];
            }


            // pan
            if ( ExceedsThreshold( m_psOut[ LEFT_PAN ], LEFT_PAN ) )
            {
```

```
                sprintf( strOut, "%s%d ", COMMAND_PAN_ABS, m_psOut[LEFT_PAN]
);
                SendToLeft( strOut );
            }

            if ( ExceedsThreshold( m_psOut[ RIGHT_PAN ], RIGHT_PAN ) )
            {
                sprintf( strOut, "%s%d ", COMMAND_PAN_ABS, m_psOut[RIGHT_PAN]
 );
                SendToRight( strOut );
            }

            // tilt
            if ( ExceedsThreshold( m_psOut[ LEFT_TILT ], LEFT_TILT ) )
            {
                sprintf( strOut, "%s%d ", COMMAND_TILT_ABS, m_psOut[LEFT_TILT
] );
                SendToLeft( strOut );
            }


            if ( ExceedsThreshold( m_psOut[ RIGHT_TILT ], RIGHT_TILT ) )
            {
                sprintf( strOut, "%s%d ", COMMAND_TILT_ABS, m_psOut[RIGHT_TIL
T] );
                SendToRight( strOut );
            }
        }// if the comm ports were opened properly //

    }// if pVal is usable //
    else return FALSE;

    return TRUE;
}



/*-------------------------------------------------------------------------
---

    Name:           ParseStringToInteger()

    Purpose:        Parse those darn strings returned by the DP head into a
                    usable integer.

-------------------------------------------------------------------------
--*/

int
CameraHead::ParseStringToInteger
    (
    char *  pStr            // change: string to convert
    )
{
    char    newStr[35];
    int     result = -1;

    // chop the first two characters
    strcpy( newStr, &(pStr[2]) );

    // find the position of the newline character
    int newline position = 0;
    char* strJunk = strchr( newStr, '\n' );

    if ( strJunk != NULL )
    {
        newline_position = strJunk - newStr;

        // chop off the end of the string
```

```
        newStr[newline_position] = '\0';

        // ...finally, convert the integer.
        result = atoi( newStr );
    }
    else
        result = -1;

    return result;
}


/*---------------------------------------------------------------------------
---

    Name:           CCatch::DPSample()

    Purpose:        Get the position of the pan and tilt.

---------------------------------------------------------------------------
--*/

BOOL
CameraHead::DPSample
    (
    )
{
    char strOut[35];
    char strIn[35];
    BOOL result = FALSE;


    // get Left pan-tilt unit position
    strcpy( strIn, "" );
    strcpy( strOut, "" );


    if ( pCommLEFT != NULL )
    {
        printf("LEFT LEFT LEFT LEFT\n");
        //get pan position
        sprintf( strOut, "%s ", QUERY_PAN );
        SendToLeft( strOut );

        strcpy( strOut, "" );

        result = pCommLEFT->ReadComm( strIn, 10 );
        printf("PAN1: %d\n", result);
        //do it again
        // Get left pan-tilt unit position
        strcpy( strIn, "" );
        sprintf( strOut, "%s ", QUERY_PAN );
        SendToLeft( strOut );

        strcpy( strOut, "" );
        result = pCommLEFT->ReadComm( strIn, 10 );
        printf("PAN2: %d\n", result);


        strcpy( strIn, "" );
        sprintf( strOut, "%s ", QUERY_PAN );
        SendToLeft( strOut );

        strcpy( strOut, "" );
        result = pCommLEFT->ReadComm( strIn, 10 );
        printf("PAN3: %d\n", result);

        if ( result == TRUE )
        {
```

```
            // the returned command looks like: "* 0", so parse it properly.
            m_plCurrentSample[ LEFT_PAN ] = ParseStringToInteger( strIn );

            // get tilt position
            strcpy( strIn, "" );
            strcpy( strOut, "" );

            sprintf( strOut, "%s ", QUERY_TILT );
            SendToLeft( strOut );

            result = pCommLEFT->ReadComm( strIn, 10 );

            if ( result == TRUE )
            {
                m_plCurrentSample[ LEFT_TILT ] = ParseStringToInteger( strIn
);
            }
            else
                return FALSE;

        }
        else
            return FALSE;

    }
    // get right pan-tilt unit position
    strcpy( strIn, "" );
    strcpy( strOut, "" );

    if ( pCommRIGHT != NULL)
    {

        // get pan position
        sprintf( strOut, "%s ", QUERY_PAN );
        SendToRight( strOut );

        strcpy( strOut, "" );

        result = pCommRIGHT->ReadComm( strIn, 10 );

        if ( result )
        {
            // the returned command looks like: "* 0", so parse it properly.
            m_plCurrentSample[ RIGHT_PAN] = ParseStringToInteger( strIn );

            // get tilt position
            strcpy( strIn, "" );
            strcpy( strOut, "" );

            sprintf( strOut, "%s ", QUERY_TILT );
            SendToRight( strOut );

            result = pCommRIGHT->ReadComm( strIn, 10 );

            if ( result )
            {
                m_plCurrentSample[ RIGHT_TILT] = ParseStringToInteger( strIn
);
            }
            else
                return FALSE;
        }
        else
            return FALSE;
    }

    return TRUE;
}
```

```
/*-------------------------------------------------------------------------
---

    Name:           CCatch::get_LeftPanAccel()

-------------------------------------------------------------------------
--*/

BOOL CameraHead::get_LeftPanAccel
    (
    long *  pVal            // out: the current acceleration value
    )
{
    if ( pVal != NULL )
    {
        char strOut[35];
        char strIn[35];
        BOOL result = FALSE;

        // make the query
        sprintf( strOut, "%s ", QUERY_PAN_ACCEL );
        result = SendToLeft( strOut );

        if ( result )
        {
            // retrieve the result
            strcpy( strIn, "" );
            result = pCommLEFT->ReadComm( strIn, 10 );

            if ( result )
            {
                long acceleration = ParseStringToInteger( strIn );

                *pVal = acceleration;
            }
            else
                return FALSE;
        }
        else
            return FALSE;
    }
    else
        return FALSE;

    return TRUE;
}

/*-------------------------------------------------------------------------
---

    Name:           CCatch::get_RightPanAccel()

-------------------------------------------------------------------------
--*/

BOOL CameraHead::get_RightPanAccel
    (
    long *  pVal            // out: the current acceleration value
    )
{
    if ( pVal != NULL )
    {
        char strOut[35];
        char strIn[35];
        BOOL result = FALSE;

        // make the query
        sprintf( strOut, "%s ", QUERY_PAN_ACCEL );
```

```
        result = SendToRight( strOut );

        if ( result )
        {
            // retrieve the result
            strcpy( strIn, "" );
            result = pCommRIGHT->ReadComm( strIn, 10 );

            if ( result )
            {
                long acceleration = ParseStringToInteger( strIn );

                *pVal = acceleration;
            }
            else
                return FALSE;
        }
        else
            return FALSE;
    }
    else
        return FALSE;

    return TRUE;
}


/*-------------------------------------------------------------------------
---

    Name:           CCatch::put_LeftPanAccel()

-------------------------------------------------------------------------
--*/

BOOL CameraHead::put_LeftPanAccel
    (
    long    newVal              // in: the new acceleration you want.
    )
{
    if ( newVal > 0 )
    {
        char strOut[35];
        BOOL result = FALSE;

        sprintf( strOut, "%s%d ", SET_PAN_ACCEL, newVal );

        result = SendToLeft( strOut );

        if ( !result )
            return FALSE;

        LeftPanAccel = newVal;

    }
    else
        return FALSE;

    return TRUE;
}

/*-------------------------------------------------------------------------
---

    Name:           CCatch::put_RightPanAccel()

-------------------------------------------------------------------------
--*/
```

```
BOOL CameraHead::put_RightPanAccel
    (
    long    newVal              // in: the new acceleration you want.
    )
{
    if ( newVal > 0 )
    {
        char strOut[35];
        BOOL result = FALSE;

        sprintf( strOut, "%s%d ", SET_PAN_ACCEL, newVal );

        result = SendToRight( strOut );

        if ( !result )
            return FALSE;

        RightPanAccel = newVal;

    }
    else
        return FALSE;

    return TRUE;
}

/*---------------------------------------------------------------------------
---

    Name:           CCatch::get_LeftTiltAccel()


---------------------------------------------------------------------------
--*/

BOOL CameraHead::get_LeftTiltAccel
    (
    long *  pVal                    // out: return acceleration value to user
.
    )
{
    if ( pVal != NULL )
    {
        char strOut[35];
        char strIn[35];
        BOOL result = FALSE;

        // make the query
        sprintf( strOut, "%s ", QUERY TILT_ACCEL );
        result = SendToLeft( strOut );

        if ( result )
        {
            // retrieve the result
            strcpy( strIn, "" );

            result = pCommLEFT->ReadComm( strIn, 10 );

            if ( result )
            {
                long acceleration = ParseStringToInteger( strIn );
                *pVal = acceleration;
            }
            else
                return FALSE;
        }
        else
            return E_FAIL;
    }
```

```
    else
        return FALSE;

    return TRUE;
}

/*-------------------------------------------------------------------------
---

    Name:           CCatch::get_RightTiltAccel()


-------------------------------------------------------------------------
--*/

BOOL CameraHead::get_RightTiltAccel
    (
    long *  pVal                    // out: return acceleration value to user
.
    )
{
    if ( pVal != NULL )
    {
        char strOut[35];
        char strIn[35];
        BOOL result = FALSE;

        // make the query
        sprintf( strOut, "%s ", QUERY TILT_ACCEL );
        result = SendToRight( strOut );

        if ( result )
        {
            // retrieve the result
            strcpy( strIn, "" );

            result = pCommRIGHT->ReadComm( strIn, 10 );

            if ( result )
            {
                long acceleration = ParseStringToInteger( strIn );
                *pVal = acceleration;
            }
            else
                return FALSE;
        }
        else
            return E_FAIL;
    }
    else
        return FALSE;

    return TRUE;
}

/*-------------------------------------------------------------------------
---

    Name:           CCatch::put_LeftTiltAccel()

-------------------------------------------------------------------------
--*/

BOOL CameraHead::put_LeftTiltAccel
    (
    long    newVal              // in: the new acceleration value
    )
{
    if ( newVal > 0 )
```

```
    {
        char strOut[35];
        BOOL result = FALSE;

        sprintf( strOut, "%s%d ", SET_TILT_ACCEL, newVal );

        result = SendToLeft( strOut );

        if ( !result )
            return FALSE;

        LeftTiltAccel = newVal;
    }
    else
        return FALSE;

    return TRUE;
}


/*---------------------------------------------------------------------------
---

    Name:           CCatch::put_RightTiltAccel()

---------------------------------------------------------------------------
--*/

BOOL CameraHead::put_RightTiltAccel
    (
    long    newVal              // in: the new acceleration value
    )
{
    if ( newVal > 0 )
    {
        char strOut[35];
        BOOL result = FALSE;

        sprintf( strOut, "%s%d ", SET_TILT_ACCEL, newVal );

        result = SendToRight( strOut );

        if ( !result )
            return FALSE;

        RightTiltAccel = newVal;
    }
    else
        return FALSE;

    return TRUE;
}

/*---------------------------------------------------------------------------
---

    Name:           CCatch::SendToLeft()

---------------------------------------------------------------------------
--*/

BOOL
CameraHead::SendToLeft
    (
    char *  pStr                    // in: string to send.
    )
{
    BOOL commResult = FALSE;
```

```
    if ( pCommLEFT != NULL )
    {
        commResult = pCommLEFT->WriteComm( pStr );
    }
    else
        commResult = FALSE;

    return commResult;
}

/*-------------------------------------------------------------------------
---

    Name:           CCatch::SendToRight()

-------------------------------------------------------------------------
--*/

BOOL
CameraHead::SendToRight
    (
    char *  pStr                    // in: string to send.
    )
{
    BOOL commResult = FALSE;

    if ( pCommRIGHT != NULL )
    {
        commResult = pCommRIGHT->WriteComm( pStr );
    }
    else
        commResult = FALSE;

    return commResult;
}



/*-------------------------------------------------------------------------
---

    Name:           CCatch::get_LeftPanSpeed()

-------------------------------------------------------------------------
--*/

BOOL CameraHead::get_LeftPanSpeed
    (
    long *  pVal                // out: current speed setting of pan motor.
    )
{
    if ( pVal != NULL )
    {
        char strOut[35];
        char strIn[35];
        BOOL result = FALSE;

        // make the query
        sprintf( strOut, "%s ", QUERY_PAN_SPEED );
        result = SendToLeft( strOut );

        if ( result )
        {
            // retrieve the result
            strcpy( strIn, "" );
            result = pCommLEFT->ReadComm( strIn, 10 );

            if ( result )
            {
```

```
                long speed = ParseStringToInteger( strIn );
                *pVal = speed;
            }
            else
                return FALSE;
        }
        else
            return FALSE;
    }
    else
        return FALSE;

    return TRUE;
}

/*-------------------------------------------------------------------------
---

    Name:           CCatch::get_RightPanSpeed()

-------------------------------------------------------------------------
--*/

BOOL CameraHead::get_RightPanSpeed
    (
    long *  pVal                // out: current speed setting of pan motor.
    )
{
    if ( pVal != NULL )
    {
        char strOut[35];
        char strIn[35];
        BOOL result = FALSE;

        // make the query
        sprintf( strOut, "%s ", QUERY_PAN_SPEED );
        result = SendToRight( strOut );

        if ( result )
        {
            // retrieve the result
            strcpy( strIn, "" );
            result = pCommRIGHT->ReadComm( strIn, 10 );

            if ( result )
            {
                long speed = ParseStringToInteger( strIn );
                *pVal = speed;
            }
            else
                return FALSE;
        }
        else
            return FALSE;
    }
    else
        return FALSE;

    return TRUE;
}


/*-------------------------------------------------------------------------
---

    Name:           CCatch::put_LeftPanSpeed()

-------------------------------------------------------------------------
--*/
```

```
BOOL CameraHead::put_LeftPanSpeed
    (
    long    newVal                  // in: whatever you want the new speed to
 be.
    )
{
    if ( newVal > 0 )
    {
        BOOL result = FALSE;
        char strOut[35];

        sprintf( strOut, "%s%d ", SET_PAN_SPEED, newVal );

        result = SendToLeft( strOut );

        if ( !result )
            return FALSE;

        LeftPanSpeed = newVal;
    }
    else
        return FALSE;

    return TRUE;
}

/*-------------------------------------------------------------------------
---

    Name:           CCatch::put_RightPanSpeed()

-------------------------------------------------------------------------
--*/

BOOL CameraHead::put_RightPanSpeed
    (
    long    newVal                  // in: whatever you want the new speed to
 be.
    )
{
    if ( newVal > 0 )
    {
        BOOL result = FALSE;
        char strOut[35];

        sprintf( strOut, "%s%d ", SET_PAN_SPEED, newVal );

        result = SendToRight( strOut );

        if ( !result )
            return FALSE;

        RightPanSpeed = newVal;
    }
    else
        return FALSE;

    return TRUE;
}
/*-------------------------------------------------------------------------
---

    Name:           CCatch::get_LeftTiltSpeed()

-------------------------------------------------------------------------
--*/

BOOL CameraHead::get_LeftTiltSpeed
```

```
    (
    long *  pVal                    // out: the current speed of the tilt mot
or
    )
{
    if ( pVal != NULL )
    {
        char strOut[35];
        char strIn[35];
        BOOL result = FALSE;

        // make the query
        sprintf( strOut, "%s ", QUERY_TILT_SPEED );

        result = SendToLeft( strOut );

        if ( result )
        {
            // retrieve the result
            strcpy( strIn, "" );
            result = pCommLEFT->ReadComm( strIn, 10 );

            if ( result )
            {
                long speed = ParseStringToInteger( strIn );
                *pVal = speed;
            }
            else
                return FALSE;
        }
        else
            return FALSE;
    }
    else
        return FALSE;

    return TRUE;
}

/*-------------------------------------------------------------------------
---

    Name:           CCatch::get_RightTiltSpeed()

-------------------------------------------------------------------------
--*/

BOOL CameraHead::get_RightTiltSpeed
    (
    long *  pVal                    // out: the current speed of the tilt mot
or
    )
{
    if ( pVal != NULL )
    {
        char strOut[35];
        char strIn[35];
        BOOL result = FALSE;

        // make the query
        sprintf( strOut, "%s ", QUERY_TILT_SPEED );

        result = SendToRight( strOut );

        if ( result )
        {
            // retrieve the result
            strcpy( strIn, "" );
            result = pCommRIGHT->ReadComm( strIn, 10 );
```

```
            if ( result )
            {
                long speed = ParseStringToInteger( strIn );
                *pVal = speed;
            }
            else
                return FALSE;
        }
        else
            return FALSE;
    }
    else
        return FALSE;

    return TRUE;
}

/*-------------------------------------------------------------------------
---

    Name:           CCatch::put_LeftTiltSpeed()

-------------------------------------------------------------------------
--*/

BOOL CameraHead::put_LeftTiltSpeed
    (
    long    newVal              // in: new speed of tilt motor
    )
{
    if ( newVal > 0 )
    {
        char strOut[35];
        BOOL result = FALSE;

        sprintf( strOut, "%s%d ", SET TILT_SPEED, newVal );
        result = SendToLeft( strOut );

        if ( !result )
        {
            return FALSE;
        }

        LeftTiltSpeed = newVal;

    }
    else
        return FALSE;

    return TRUE;
}

/*-------------------------------------------------------------------------
---

    Name:           CCatch::put_RightTiltSpeed()

-------------------------------------------------------------------------
--*/

BOOL CameraHead::put_RightTiltSpeed
    (
    long    newVal              // in: new speed of tilt motor
    )
{
    if ( newVal > 0 )
    {
        char strOut[35];
```

```
        BOOL result = FALSE;

        sprintf( strOut, "%s%d ", SET_TILT_SPEED, newVal );
        result = SendToRight( strOut );

        if ( !result )
        {
            return FALSE;
        }

        RightTiltSpeed = newVal;

    }
    else
        return FALSE;

    return TRUE;
}

/*-------------------------------------------------------------------------
---

    Name:           CCatch::Reset()

    Purpose:        Reset the pan-tilt head.

-------------------------------------------------------------------------
--*/

BOOL CameraHead::Reset()
{
    char strOut[35];
    BOOL result1 = FALSE;
    BOOL result2 = FALSE;

    // reset pan and tilt
    sprintf( strOut, "%s ", RESET );
    result1 = SendToLeft( strOut );
    result2 = SendToRight( strOut );


    if ( result1 && result2 )
    {
        m_plCurrentSample[ LEFT_PAN  ] = 0;
        m_plCurrentSample[ LEFT_TILT ] = 0;
        m_plCurrentSample[ RIGHT_PAN ] = 0;
        m_plCurrentSample[ RIGHT_TILT] = 0;
    }
    else
        return FALSE;

    return TRUE;
}

/*-------------------------------------------------------------------------
---

    Name:           CCatch::Stop()

    Purpose:        Stop any pan-tilt motion in progress.

-------------------------------------------------------------------------
--*/

BOOL CameraHead::Stop()
{
    char strOut[35];
    BOOL result1 = FALSE;
    BOOL result2 = FALSE;
```

```
    sprintf( strOut, "%s ", HALT ALL );
    result1 = SendToLeft( strOut );
    result2 = SendToRight( strOut );

    if ( !result1 || !result2 )
        return FALSE;

    return TRUE;
}


/*-------------------------------------------------------------------------
---

    Name:           CCatch::get_AxisThreshold()

    Purpose:        Get the movement threshold for this axis.

    Created By:     RMW - watson@vuse.vanderbilt.edu
    Date Modified:  Friday, August 22, 1997

-------------------------------------------------------------------------
--*/

BOOL CameraHead::get_AxisThreshold
    (
    long    Index,                  // in: which axis you are interested in
    long *  pVal                    // out: the current threshold value
    )
{
    if ( pVal == NULL )
        return FALSE;

    if ( IndexIsValid( Index ) )
        *pVal = m plAxisThreshold[ Index ];
    else return FALSE;

    return TRUE;
}


/*-------------------------------------------------------------------------
---

    Name:           CCatch::put_AxisThreshold()

    Purpose:        Set the movement threshold for this axis.

    Created By:     RMW - watson@vuse.vanderbilt.edu
    Date Modified:  Friday, August 22, 1997

-------------------------------------------------------------------------
--*/

BOOL CameraHead::put_AxisThreshold
    (
    long    Index,                  // in: the axis you are interested in
    long    newVal                  // in: the new threshold value
    )
{
    if ( IndexIsValid( Index ) )
        m plAxisThreshold[ Index ] = newVal;
    else return FALSE;

    return TRUE;
}
```

```
/*--------------------------------------------------------------------------
---

    Name:           CCatch::ExceedsThreshold()

--------------------------------------------------------------------------
--*/

BOOL
CameraHead::ExceedsThreshold
    (
    long    Val,
    long    axis_index
    )
{
        if (abs(Val - m_plCurrentSample[axis_index]) >= m_plAxisThreshold[axi
s_index])
        return TRUE;
    else
        return FALSE;
}



void CameraHead::MoveHead(double *command)
{
        // COMMAND HEAD
        // Setup output values from input vector
        for(int i=0;i<NUM_AXES;i++)
            m_pdLinkCurrentHeadCommand[i] = command[i];

        // convert angles to positions and sent to the head.
        BOOL hrAngleConversionStatus = FALSE;
        hrAngleConversionStatus = HeadAngles2Positions( m_pdLinkCurrentHeadCo
mmand, NUM_AXES );

        // here's where you SET values.
        if ( SUCCEEDED( hrAngleConversionStatus ) )
            CommandHeadAbsolute( m_pdLinkCurrentHeadCommand, NUM_AXES );
}

void CameraHead::GetHeadPositions(double *val)
{
        // SAMPLE HEAD
        SampleHead( m_pdLinkCurrentHeadSample, NUM_AXES );

        // convert angles to positions and sent to the head.
        BOOL hrPositionConversionStatus = FALSE;
        hrPositionConversionStatus = HeadPositions2Angles( m_pdLinkCurrentHea
dSample, NUM_AXES );

        if ( SUCCEEDED( hrPositionConversionStatus ) )
        {
            CalculateOverallPanTilt(m_pdLinkCurrentHeadSample);
            for(int i=0;i<NUM_AXES;i++)
                val[i] = m_pdLinkCurrentHeadSample[i];
        }
}

////////////////////////////////////////////////////////////////////////
// Function to obtain verge angles and a single pan-tilt
// angle from two cameras (08/05/03)
// Val = [LP LT RP RT OverallPan Avg Tilt VergeAngles]
// Notes on the mathematical theory can be found in:
// Cis\common\InetPub\wwwroot\IRL\Doc\Head\Head Change\Isac_Vis_Angles
////////////////////////////////////////////////////////////////////////

bool CameraHead::CalculateOverallPanTilt(double* Val)
```

```
{
    double dtempL, dtempR, Alpha; //AA, BB;
    double ThetaLeft, ThetaRight;

    // Collect and Allocate Catch Angles from Input Vector
    dtempL = Val[0];
    dtempR = Val[2];

    // Converting from degrees to radians
    ThetaLeft   = D2R*(90-dtempL);
    ThetaRight  = D2R*(90+dtempR);

    // Calculate overall pan
    // This equation is to find a singular pan from two cameras
    // The document can be found at Cis\common\InetPub\wwwroot\IRL\Doc\Head\H
ead Change\Finding a Singular Pan

    Alpha = atan (sin(ThetaRight-ThetaLeft)/(2*sin(ThetaLeft)*sin(ThetaRight)
));

    // Store this in the 5th element of the array  as an overall pan angle in
 degrees
    Val[4] = R2D * Alpha;    // clockwise angles are negative

    // Calculate average Tilt
    Val[5] = (Val[1] + Val[3]) / 2;

    // Converting from degrees to radians
    dtempL = D2R*dtempL;
    dtempR = D2R*dtempR;

    return TRUE;

}


/*-------------------------------------------------------------------------
---

    Name:           CCatch::HeadAngles2Positions()

    Purpose:        Convert angles that come in into positions understood by
the head.

-------------------------------------------------------------------------
--*/

BOOL
CameraHead::HeadAngles2Positions
    (
    double *    pVal,               // in/out: angles->positions
    long        length              // in: how big
    )
{
    // Angle 2 Position conversion
    if ( pVal != NULL )
    {
        pVal[ LEFT_PAN  ] =  pVal[ LEFT_PAN  ] / PULSE_TO_ANG_PAN_LEFT;
        pVal[ LEFT_TILT ] =  pVal[ LEFT_TILT ] / PULSE_TO_ANG_TILT;
        pVal[ RIGHT_PAN ] =  pVal[ RIGHT_PAN ] / PULSE_TO_ANG_PAN_RIGHT;
        pVal[ RIGHT_TILT] =  pVal[ RIGHT_TILT] / PULSE_TO_ANG_TILT;
    }
    else
        return FALSE;

    return TRUE;
}

/*-------------------------------------------------------------------------
```

```
---

    Name:           CCatch::HeadPositions2Angles()

    Purpose:        Convert positions that come in into angles understood by
everyone.

------------------------------------------------------------------------------
--*/

BOOL
CameraHead::HeadPositions2Angles
    (
    double *    pVal,               // in/out: positions->angles
    long        length              // in: how big
    )
{
    // Position 2 Angle conversion
    if ( pVal != NULL )
    {
        pVal[ LEFT_PAN  ] = pVal[ LEFT_PAN  ] * PULSE_TO_ANG_PAN_LEFT;
        pVal[ LEFT_TILT ] = pVal[ LEFT_TILT ] * PULSE_TO_ANG_TILT;
        pVal[ RIGHT_PAN ] = pVal[ RIGHT_PAN ] * PULSE_TO_ANG_PAN_RIGHT;
        pVal[ RIGHT_TILT] = pVal[ RIGHT_TILT] * PULSE_TO_ANG_TILT;
    }
    else
        return FALSE;

    return TRUE;
}



void CameraHead::Home()
{
    double buff[4];
    buff[0] = buff[1] = buff[2] = buff[3] = 0.0f;
    MoveHead(buff);
}
```

```
// C++ container classes for obtaining exclusive access
//  to the comm port for simple use such as reading/writing
//  ASCII characters. Handles all setup so the user
//  simply calls the WriteComm or ReadComm functions to read
//  and write to comm port specified in the constructor.
//************
// Summer 1996
// Joe Christopher
// MFC parts taken out by Rick Watson 8/5/1997
//************

//#include "stdafx.h"

// Standard library
#include "windows.h"
#include    <stdlib.h>

class CCommPort
{
public:
    CCommPort( char* Port="\\COM2" , ULONG  Baud=9600 );    // Constructor wi
th default COM2, 9600 baud
    ~CCommPort();
    BOOL WriteComm( char* Output);
    BOOL WriteChar( char Output );
    BOOL ReadComm( char* Input, ULONG Bytes );

protected:
    // Vars. used in setting up and obtaining COMM port
    DWORD m_dwError;
    DWORD m_dwNumBytes;
    COMMTIMEOUTS m_ctTimeouts;

    BOOL    m_bGetComm; // Booleans for evaluating successful setup
    BOOL    m_bSetComm; // of COMM port parameters such as baud, parity
    BOOL    m_bGetTime; // stop bits, timeouts, byte size, etc.
    BOOL    m_bSetTime; //

    DCB     m_dDcb;     // device control block for the port
    HANDLE  m_hCom;     // handle for the comm port

 };
```

```
// Function declarations for CommPort class
// Summer 1996
// Joe Christopher
// MFC parts taken out by Rick Watson 8/5/1997
#include "stdafx.h"
#include "commport.h"

// Constructor
CCommPort::CCommPort( char* Port, ULONG Baud )
{

        // Putting COM port handle creation here
    m hCom = CreateFile( Port,
             GENERIC_READ | GENERIC_WRITE,
             0,      // comm devices must be opened w/exclusive access
             NULL,   // no security attrs
             OPEN_EXISTING, // comm devices must use OPEN_EXISTING
             0,      // not overlapped I/O
             NULL    // hTemplate must be NULL for comm devices
             );

    if( m_hCom == INVALID_HANDLE_VALUE) // Valid comm port handle
    {
        m dwError = GetLastError(); // Get error
        MessageBeep( MB_OK );
        MessageBeep( MB_OK );
    }
    else
    {
        if(m_bGetComm = GetCommState(m_hCom, &m_dDcb)) // Get comm states int
o m_dDcb
        {
            m_dDcb.BaudRate = Baud; // User-passed baud rate
            m_dDcb.ByteSize = 8;    // 8 bits / byte
            m_dDcb.StopBits = ONESTOPBIT;   // 1 stop bit
            m_dDcb.Parity    = NOPARITY;    // No parity
            m_bSetComm= SetCommState(m_hCom, (LPDCB) &m_dDcb);  // Set comm p
ort for new settings
        }
        if(m_bGetTime = GetCommTimeouts(m_hCom, &m_ctTimeouts)) // Get comm t
imeouts into m_ctTimeouts
        {
                m_ctTimeouts.ReadIntervalTimeout = 100; // 1 sec between char
 rxd
                m_ctTimeouts.ReadTotalTimeoutMultiplier = 0;    // No multipl
ier
                m_bSetTime = SetCommTimeouts( m_hCom, &m_ctTimeouts ); // Set
 new timeout values

        }
        // Check for failure in getting or setting parameters for comm ports
        // Print appropriate message
        if( !m_bGetComm || !m_bSetComm || !m_bGetTime || !m_bSetTime )
            MessageBox( NULL,
                "Error initializing comm port",
                NULL,
                MB_ICONEXCLAMATION
                );
    } // end if valid handle

} // end constructor

// Destructor
CCommPort::~CCommPort()
{
    CloseHandle(m_hCom); // Close comm port upon exit
} // end destructor

// WriteComm function takes a CString as a parameter and
```

```
// writes it as ASCII chars to the comm port
BOOL CCommPort::WriteComm( char* Output )
{
    BOOL result = FALSE;

    // Write out data
    result = PurgeComm( m_hCom, PURGE_TXABORT | PURGE_TXCLEAR | PURGE_RXABORT
 | PURGE_RXCLEAR );

    if ( result )
    {
        WriteFile( m_hCom, Output, strlen(Output), &m_dwNumBytes, NULL );

        if ( m_dwNumBytes <= 0 )
            result = FALSE;
    }

    return result;
} // end WriteComm

// WriteChar function takes a character and waits for its successful transmis
sion
//  Use this when you need to transmit ASCII 0x00 because char* see as NULL c
har
BOOL CCommPort::WriteChar( char Output )
{
    // Wait for successful transmission
    BOOL result = FALSE;
    result = TransmitCommChar( m_hCom, Output );

    return result;
}
// ReadComm function takes a LPCTSTR pointer as a param
// and reads in from the comm port the specified number of bytes.
BOOL CCommPort::ReadComm( char* Input, ULONG Bytes )
{
    // Read in data
    BOOL result = FALSE;
    result = ReadFile(m_hCom, Input, Bytes, &m_dwNumBytes, NULL);

    if ( m_dwNumBytes <= 0 )
        result = FALSE;

    return result;
} // end ReadComm
```

```
/*
 *   C++ sockets on Unix and Windows
 *   Copyright (C) 2002
 *
 *   This program is free software; you can redistribute it and/or modify
 *   it under the terms of the GNU General Public License as published by
 *   the Free Software Foundation; either version 2 of the License, or
 *   (at your option) any later version.
 *
 *   This program is distributed in the hope that it will be useful,
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *   GNU General Public License for more details.
 *
 *   You should have received a copy of the GNU General Public License
 *   along with this program; if not, write to the Free Software
 *   Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  US
A
 */

#ifndef   PRACTICALSOCKET INCLUDED
#define __PRACTICALSOCKET_INCLUDED__

#include <string>            // For string
#include <exception>         // For exception class

using namespace std;

/**
 *   Signals a problem with the execution of a socket call.
 */
class SocketException : public exception {
public:
  /**
   *   Construct a SocketException with a explanatory message.
   *   @param message explanatory message
   *   @param incSysMsg true if system message (from strerror(errno))
   *   should be postfixed to the user provided message
   */
  SocketException(const string &message, bool inclSysMsg = false) throw();

  /**
   *   Provided just to guarantee that no exceptions are thrown.
   */
  ~SocketException() throw();

  /**
   *   Get the exception message
   *   @return exception message
   */
  const char *what() const throw();

private:
  string userMessage;  // Exception message
};

/**
 *   Base class representing basic communication endpoint
 */
class Socket {
public:
  /**
   *   Close and deallocate this socket
   */
  ~Socket();

  /**
   *   Get the local address
   *   @return local address of socket
```

```
   *   @exception SocketException thrown if fetch fails
   */
  string getLocalAddress() throw(SocketException);

  /**
   *   Get the local port
   *   @return local port of socket
   *   @exception SocketException thrown if fetch fails
   */
  unsigned short getLocalPort() throw(SocketException);

  /**
   *   Set the local port to the specified port and the local address
   *   to any interface
   *   @param localPort local port
   *   @exception SocketException thrown if setting local port fails
   */
  void setLocalPort(unsigned short localPort) throw(SocketException);

  /**
   *   Set the local port to the specified port and the local address
   *   to the specified address.  If you omit the port, a random port
   *   will be selected.
   *   @param localAddress local address
   *   @param localPort local port
   *   @exception SocketException thrown if setting local port or address fai
ls
   */
  void setLocalAddressAndPort(const string &localAddress,
    unsigned short localPort = 0) throw(SocketException);

  /**
   *   If WinSock, unload the WinSock DLLs; otherwise do nothing.  We ignore
   *   this in our sample client code but include it in the library for
   *   completeness.  If you are running on Windows and you are concerned
   *   about DLL resource consumption, call this after you are done with all
   *   Socket instances.  If you execute this on Windows while some instance
of
   *   Socket exists, you are toast.  For portability of client code, this is

   *   an empty function on non-Windows platforms so you can always include i
t.
   *   @param buffer buffer to receive the data
   *   @param bufferLen maximum number of bytes to read into buffer
   *   @return number of bytes read, 0 for EOF, and -1 for error
   *   @exception SocketException thrown WinSock clean up fails
   */
  static void cleanUp() throw(SocketException);

  /**
   *   Resolve the specified service for the specified protocol to the
   *   corresponding port number in host byte order
   *   @param service service to resolve (e.g., "http")
   *   @param protocol protocol of service to resolve.  Default is "tcp".
   */
  static unsigned short resolveService(const string &service,
                                       const string &protocol = "tcp");

private:
  // Prevent the user from trying to use value semantics on this object
  Socket(const Socket &sock);
  void operator=(const Socket &sock);

protected:
  int sockDesc;              // Socket descriptor
  Socket(int type, int protocol) throw(SocketException);
  Socket(int sockDesc);
};
```

```
/**
 *   Socket which is able to connect, send, and receive
 */
class CommunicatingSocket : public Socket {
public:
  /**
   *   Establish a socket connection with the given foreign
   *   address and port
   *   @param foreignAddress foreign address (IP address or name)
   *   @param foreignPort foreign port
   *   @exception SocketException thrown if unable to establish connection
   */
  void connect(const string &foreignAddress, unsigned short foreignPort)
    throw(SocketException);

  /**
   *   Write the given buffer to this socket.  Call connect() before
   *   calling send()
   *   @param buffer buffer to be written
   *   @param bufferLen number of bytes from buffer to be written
   *   @exception SocketException thrown if unable to send data
   */
  void send(const void *buffer, int bufferLen) throw(SocketException);

  /**
   *   Read into the given buffer up to bufferLen bytes data from this
   *   socket.  Call connect() before calling recv()
   *   @param buffer buffer to receive the data
   *   @param bufferLen maximum number of bytes to read into buffer
   *   @return number of bytes read, 0 for EOF, and -1 for error
   *   @exception SocketException thrown if unable to receive data
   */
  int recv(void *buffer, int bufferLen) throw(SocketException);

  /**
   *   Get the foreign address.  Call connect() before calling recv()
   *   @return foreign address
   *   @exception SocketException thrown if unable to fetch foreign address
   */
  string getForeignAddress() throw(SocketException);

  /**
   *   Get the foreign port.  Call connect() before calling recv()
   *   @return foreign port
   *   @exception SocketException thrown if unable to fetch foreign port
   */
  unsigned short getForeignPort() throw(SocketException);

protected:
  CommunicatingSocket(int type, int protocol) throw(SocketException);
  CommunicatingSocket(int newConnSD);
};

/**
 *   TCP socket for communication with other TCP sockets
 */
class TCPSocket : public CommunicatingSocket {
public:
  /**
   *   Construct a TCP socket with no connection
   *   @exception SocketException thrown if unable to create TCP socket
   */
  TCPSocket() throw(SocketException);

  /**
   *   Construct a TCP socket with a connection to the given foreign address
   *   and port
   *   @param foreignAddress foreign address (IP address or name)
   *   @param foreignPort foreign port
```

```
   *   @exception SocketException thrown if unable to create TCP socket
   */
  TCPSocket(const string &foreignAddress, unsigned short foreignPort)
      throw(SocketException);

private:
  // Access for TCPServerSocket::accept() connection creation
  friend class TCPServerSocket;
  TCPSocket(int newConnSD);
};

/**
 *   TCP socket class for servers
 */
class TCPServerSocket : public Socket {
public:
  /**
   *   Construct a TCP socket for use with a server, accepting connections
   *   on the specified port on any interface
   *   @param localPort local port of server socket, a value of zero will
   *                    give a system-assigned unused port
   *   @param queueLen maximum queue length for outstanding
   *                   connection requests (default 5)
   *   @exception SocketException thrown if unable to create TCP server socke
t
   */
  TCPServerSocket(unsigned short localPort, int queueLen = 5)
      throw(SocketException);

  /**
   *   Construct a TCP socket for use with a server, accepting connections
   *   on the specified port on the interface specified by the given address
   *   @param localAddress local interface (address) of server socket
   *   @param localPort local port of server socket
   *   @param queueLen maximum queue length for outstanding
   *                   connection requests (default 5)
   *   @exception SocketException thrown if unable to create TCP server socke
t
   */
  TCPServerSocket(const string &localAddress, unsigned short localPort,
      int queueLen = 5) throw(SocketException);

  /**
   *   Blocks until a new connection is established on this socket or error
   *   @return new connection socket
   *   @exception SocketException thrown if attempt to accept a new connectio
n fails
   */
  TCPSocket *accept() throw(SocketException);

private:
  void setListen(int queueLen) throw(SocketException);
};

/**
  *   UDP socket class
  */
class UDPSocket : public CommunicatingSocket {
public:
  /**
   *   Construct a UDP socket
   *   @exception SocketException thrown if unable to create UDP socket
   */
  UDPSocket() throw(SocketException);

  /**
   *   Construct a UDP socket with the given local port
   *   @param localPort local port
   *   @exception SocketException thrown if unable to create UDP socket
```

```
   */
  UDPSocket(unsigned short localPort) throw(SocketException);

  /**
   *   Construct a UDP socket with the given local port and address
   *   @param localAddress local address
   *   @param localPort local port
   *   @exception SocketException thrown if unable to create UDP socket
   */
  UDPSocket(const string &localAddress, unsigned short localPort)
      throw(SocketException);

  /**
   *   Unset foreign address and port
   *   @return true if disassociation is successful
   *   @exception SocketException thrown if unable to disconnect UDP socket
   */
  void disconnect() throw(SocketException);

  /**
   *   Send the given buffer as a UDP datagram to the
   *   specified address/port
   *   @param buffer buffer to be written
   *   @param bufferLen number of bytes to write
   *   @param foreignAddress address (IP address or name) to send to
   *   @param foreignPort port number to send to
   *   @return true if send is successful
   *   @exception SocketException thrown if unable to send datagram
   */
  void sendTo(const void *buffer, int bufferLen, const string &foreignAddress
,
           unsigned short foreignPort) throw(SocketException);

  /**
   *   Read read up to bufferLen bytes data from this socket.  The given buff
er
   *   is where the data will be placed
   *   @param buffer buffer to receive data
   *   @param bufferLen maximum number of bytes to receive
   *   @param sourceAddress address of datagram source
   *   @param sourcePort port of data source
   *   @return number of bytes received and -1 for error
   *   @exception SocketException thrown if unable to receive datagram
   */
  int recvFrom(void *buffer, int bufferLen, string &sourceAddress,
               unsigned short &sourcePort) throw(SocketException);

  /**
   *   Set the multicast TTL
   *   @param multicastTTL multicast TTL
   *   @exception SocketException thrown if unable to set TTL
   */
  void setMulticastTTL(unsigned char multicastTTL) throw(SocketException);

  /**
   *   Join the specified multicast group
   *   @param multicastGroup multicast group address to join
   *   @exception SocketException thrown if unable to join group
   */
  void joinGroup(const string &multicastGroup) throw(SocketException);

  /**
   *   Leave the specified multicast group
   *   @param multicastGroup multicast group address to leave
   *   @exception SocketException thrown if unable to leave group
   */
  void leaveGroup(const string &multicastGroup) throw(SocketException);

private:
```

```
  void setBroadcast();
};

#endif
```

```
/*
 *   C++ sockets on Unix and Windows
 *   Copyright (C) 2002
 *
 *   This program is free software; you can redistribute it and/or modify
 *   it under the terms of the GNU General Public License as published by
 *   the Free Software Foundation; either version 2 of the License, or
 *   (at your option) any later version.
 *
 *   This program is distributed in the hope that it will be useful,
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *   GNU General Public License for more details.
 *
 *   You should have received a copy of the GNU General Public License
 *   along with this program; if not, write to the Free Software
 *   Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  US
A
 */

#include "stdafx.h"
#include "PracticalSocket.h"

#ifdef WIN32
  #include <winsock.h>         // For socket(), connect(), send(), and recv()
  typedef int socklen t;
  typedef char raw_type;       // Type used for raw data on this platform
#else
  #include <sys/types.h>       // For data types
  #include <sys/socket.h>      // For socket(), connect(), send(), and recv()
  #include <netdb.h>           // For gethostbyname()
  #include <arpa/inet.h>       // For inet addr()
  #include <unistd.h>          // For close()
  #include <netinet/in.h>      // For sockaddr_in
  typedef void raw_type;       // Type used for raw data on this platform
#endif

#include <errno.h>             // For errno

using namespace std;

#ifdef WIN32
static bool initialized = false;
#endif

// SocketException Code

SocketException::SocketException(const string &message, bool inclSysMsg)
  throw() : userMessage(message) {
  if (inclSysMsg) {
    userMessage.append(": ");
    userMessage.append(strerror(errno));
  }
}

SocketException::~SocketException() throw() {
}

const char *SocketException::what() const throw() {
  return userMessage.c_str();
}

// Function to fill in address structure given an address and port
static void fillAddr(const string &address, unsigned short port,
                     sockaddr_in &addr) {
  memset(&addr, 0, sizeof(addr));  // Zero out address structure
  addr.sin_family = AF_INET;       // Internet address

  hostent *host;  // Resolve name
```

```
  if ((host = gethostbyname(address.c str())) == NULL) {
    // strerror() will not work for gethostbyname() and hstrerror()
    // is supposedly obsolete
    throw SocketException("Failed to resolve name (gethostbyname())");
  }
  addr.sin_addr.s_addr = *((unsigned long *) host->h_addr_list[0]);

  addr.sin_port = htons(port);     // Assign port in network byte order
}

// Socket Code

Socket::Socket(int type, int protocol) throw(SocketException) {
  #ifdef WIN32
    if (!initialized) {
      WORD wVersionRequested;
      WSADATA wsaData;

      wVersionRequested = MAKEWORD(2, 0);              // Request WinSock v2.
0
      if (WSAStartup(wVersionRequested, &wsaData) != 0) {  // Load WinSock DL
L
        throw SocketException("Unable to load WinSock DLL");
      }
      initialized = true;
    }
  #endif

  // Make a new socket
  if ((sockDesc = socket(PF_INET, type, protocol)) < 0) {
    throw SocketException("Socket creation failed (socket())", true);
  }
}

Socket::Socket(int sockDesc) {
  this->sockDesc = sockDesc;
}

Socket::~Socket() {
  #ifdef WIN32
    ::closesocket(sockDesc);
  #else
    ::close(sockDesc);
  #endif
  sockDesc = -1;
}

string Socket::getLocalAddress() throw(SocketException) {
  sockaddr in addr;
  unsigned int addr_len = sizeof(addr);

  if (getsockname(sockDesc, (sockaddr *) &addr, (socklen_t *) &addr_len) < 0)
 {
    throw SocketException("Fetch of local address failed (getsockname())", tr
ue);
  }
  return inet_ntoa(addr.sin_addr);
}

unsigned short Socket::getLocalPort() throw(SocketException) {
  sockaddr in addr;
  unsigned int addr_len = sizeof(addr);

  if (getsockname(sockDesc, (sockaddr *) &addr, (socklen_t *) &addr_len) < 0)
 {
    throw SocketException("Fetch of local port failed (getsockname())", true)
;
  }
  return ntohs(addr.sin_port);
```

```
}

void Socket::setLocalPort(unsigned short localPort) throw(SocketException) {
  // Bind the socket to its port
  sockaddr_in localAddr;
  memset(&localAddr, 0, sizeof(localAddr));
  localAddr.sin_family = AF_INET;
  localAddr.sin_addr.s_addr = htonl(INADDR_ANY);
  localAddr.sin_port = htons(localPort);

  if (bind(sockDesc, (sockaddr *) &localAddr, sizeof(sockaddr_in)) < 0) {
    throw SocketException("Set of local port failed (bind())", true);
  }
}

void Socket::setLocalAddressAndPort(const string &localAddress,
    unsigned short localPort) throw(SocketException) {
  // Get the address of the requested host
  sockaddr_in localAddr;
  fillAddr(localAddress, localPort, localAddr);

  if (bind(sockDesc, (sockaddr *) &localAddr, sizeof(sockaddr_in)) < 0) {
    throw SocketException("Set of local address and port failed (bind())", tr
ue);
  }
}

void Socket::cleanUp() throw(SocketException) {
  #ifdef WIN32
    if (WSACleanup() != 0) {
      throw SocketException("WSACleanup() failed");
    }
  #endif
}

unsigned short Socket::resolveService(const string &service,
                                      const string &protocol) {
  struct servent *serv;        /* Structure containing service information */

  if ((serv = getservbyname(service.c_str(), protocol.c_str())) == NULL)
    return atoi(service.c_str());  /* Service is port number */
  else
    return ntohs(serv->s_port);    /* Found port (network byte order) by name
 */
}

// CommunicatingSocket Code

CommunicatingSocket::CommunicatingSocket(int type, int protocol)
    throw(SocketException) : Socket(type, protocol) {
}

CommunicatingSocket::CommunicatingSocket(int newConnSD) : Socket(newConnSD) {
}

void CommunicatingSocket::connect(const string &foreignAddress,
    unsigned short foreignPort) throw(SocketException) {
  // Get the address of the requested host
  sockaddr_in destAddr;
  fillAddr(foreignAddress, foreignPort, destAddr);

  // Try to connect to the given port
  if (::connect(sockDesc, (sockaddr *) &destAddr, sizeof(destAddr)) < 0) {
    throw SocketException("Connect failed (connect())", true);
  }
}

void CommunicatingSocket::send(const void *buffer, int bufferLen)
    throw(SocketException) {
```

```
  if (::send(sockDesc, (raw type *) buffer, bufferLen, 0) < 0) {
    throw SocketException("Send failed (send())", true);
  }
}

int CommunicatingSocket::recv(void *buffer, int bufferLen)
    throw(SocketException) {
  int rtn;
  if ((rtn = ::recv(sockDesc, (raw type *) buffer, bufferLen, 0)) < 0) {
    throw SocketException("Received failed (recv())", true);
  }

  return rtn;
}

string CommunicatingSocket::getForeignAddress()
    throw(SocketException) {
  sockaddr in addr;
  unsigned int addr_len = sizeof(addr);

  if (getpeername(sockDesc, (sockaddr *) &addr,(socklen_t *) &addr_len) < 0)
{
    throw SocketException("Fetch of foreign address failed (getpeername())",
true);
  }
  return inet_ntoa(addr.sin_addr);
}

unsigned short CommunicatingSocket::getForeignPort() throw(SocketException) {
  sockaddr_in addr;
  unsigned int addr_len = sizeof(addr);

  if (getpeername(sockDesc, (sockaddr *) &addr, (socklen_t *) &addr_len) < 0)
 {
    throw SocketException("Fetch of foreign port failed (getpeername())", tru
e);
  }
  return ntohs(addr.sin_port);
}

// TCPSocket Code

TCPSocket::TCPSocket()
    throw(SocketException) : CommunicatingSocket(SOCK_STREAM,
    IPPROTO_TCP) {
}

TCPSocket::TCPSocket(const string &foreignAddress, unsigned short foreignPort
)
    throw(SocketException) : CommunicatingSocket(SOCK_STREAM, IPPROTO_TCP) {
  connect(foreignAddress, foreignPort);
}

TCPSocket::TCPSocket(int newConnSD) : CommunicatingSocket(newConnSD) {
}

// TCPServerSocket Code

TCPServerSocket::TCPServerSocket(unsigned short localPort, int queueLen)
    throw(SocketException) : Socket(SOCK_STREAM, IPPROTO_TCP) {
  setLocalPort(localPort);
  setListen(queueLen);
}

TCPServerSocket::TCPServerSocket(const string &localAddress,
    unsigned short localPort, int queueLen)
    throw(SocketException) : Socket(SOCK_STREAM, IPPROTO_TCP) {
  setLocalAddressAndPort(localAddress, localPort);
  setListen(queueLen);
```

```
}

TCPSocket *TCPServerSocket::accept() throw(SocketException) {
  int newConnSD;
  if ((newConnSD = ::accept(sockDesc, NULL, 0)) < 0) {
    throw SocketException("Accept failed (accept())", true);
  }

  return new TCPSocket(newConnSD);
}

void TCPServerSocket::setListen(int queueLen) throw(SocketException) {
  if (listen(sockDesc, queueLen) < 0) {
    throw SocketException("Set listening socket failed (listen())", true);
  }
}

// UDPSocket Code

UDPSocket::UDPSocket() throw(SocketException) : CommunicatingSocket(SOCK_DGRA
M,
    IPPROTO_UDP) {
  setBroadcast();
}

UDPSocket::UDPSocket(unsigned short localPort)  throw(SocketException) :
    CommunicatingSocket(SOCK_DGRAM, IPPROTO_UDP) {
  setLocalPort(localPort);
  setBroadcast();
}

UDPSocket::UDPSocket(const string &localAddress, unsigned short localPort)
     throw(SocketException) : CommunicatingSocket(SOCK_DGRAM, IPPROTO_UDP) {
  setLocalAddressAndPort(localAddress, localPort);
  setBroadcast();
}

void UDPSocket::setBroadcast() {
  // If this fails, we'll hear about it when we try to send.  This will allow

  // system that cannot broadcast to continue if they don't plan to broadcast
  int broadcastPermission = 1;
  setsockopt(sockDesc, SOL_SOCKET, SO_BROADCAST,
             (raw_type *) &broadcastPermission, sizeof(broadcastPermission));
}

void UDPSocket::disconnect() throw(SocketException) {
  sockaddr_in nullAddr;
  memset(&nullAddr, 0, sizeof(nullAddr));
  nullAddr.sin_family = AF_UNSPEC;

  // Try to disconnect
  if (::connect(sockDesc, (sockaddr *) &nullAddr, sizeof(nullAddr)) < 0) {
   #ifdef WIN32
    if (errno != WSAEAFNOSUPPORT) {
   #else
    if (errno != EAFNOSUPPORT) {
   #endif
      throw SocketException("Disconnect failed (connect())", true);
    }
  }
}

void UDPSocket::sendTo(const void *buffer, int bufferLen,
    const string &foreignAddress, unsigned short foreignPort)
    throw(SocketException) {
  sockaddr_in destAddr;
  fillAddr(foreignAddress, foreignPort, destAddr);
```

```
  // Write out the whole buffer as a single message.
  if (sendto(sockDesc, (raw_type *) buffer, bufferLen, 0,
             (sockaddr *) &destAddr, sizeof(destAddr)) != bufferLen) {
    throw SocketException("Send failed (sendto())", true);
  }
}

int UDPSocket::recvFrom(void *buffer, int bufferLen, string &sourceAddress,
    unsigned short &sourcePort) throw(SocketException) {
  sockaddr_in clntAddr;
  socklen_t addrLen = sizeof(clntAddr);
  int rtn;
  if ((rtn = recvfrom(sockDesc, (raw_type *) buffer, bufferLen, 0,
                      (sockaddr *) &clntAddr, (socklen_t *) &addrLen)) < 0) {
    throw SocketException("Receive failed (recvfrom())", true);
  }
  sourceAddress = inet_ntoa(clntAddr.sin_addr);
  sourcePort = ntohs(clntAddr.sin_port);

  return rtn;
}

void UDPSocket::setMulticastTTL(unsigned char multicastTTL) throw(SocketExcep
tion) {
  if (setsockopt(sockDesc, IPPROTO_IP, IP_MULTICAST_TTL,
                 (raw_type *) &multicastTTL, sizeof(multicastTTL)) < 0) {
    throw SocketException("Multicast TTL set failed (setsockopt())", true);
  }
}

void UDPSocket::joinGroup(const string &multicastGroup) throw(SocketException
) {
  struct ip_mreq multicastRequest;

  multicastRequest.imr_multiaddr.s_addr = inet_addr(multicastGroup.c_str());
  multicastRequest.imr_interface.s_addr = htonl(INADDR_ANY);
  if (setsockopt(sockDesc, IPPROTO_IP, IP_ADD_MEMBERSHIP,
                 (raw_type *) &multicastRequest,
                 sizeof(multicastRequest)) < 0) {
    throw SocketException("Multicast group join failed (setsockopt())", true)
;
  }
}

void UDPSocket::leaveGroup(const string &multicastGroup) throw(SocketExceptio
n) {
  struct ip_mreq multicastRequest;

  multicastRequest.imr_multiaddr.s_addr = inet_addr(multicastGroup.c_str());
  multicastRequest.imr_interface.s_addr = htonl(INADDR_ANY);
  if (setsockopt(sockDesc, IPPROTO_IP, IP_DROP_MEMBERSHIP,
                 (raw_type *) &multicastRequest,
                 sizeof(multicastRequest)) < 0) {
    throw SocketException("Multicast group leave failed (setsockopt())", true
);
  }
}
```

```
/*****************************************************************************
 |actuation.cpp                                                    Sean Thornton|
 |---------------------------------------------------------------------------|
 | This program uses ISAC's left arm to imitate motions made by a human     |
 | by actuating to joint angles received via UDP from imitation.exe on      |
 | Sally.                                                                    |
 |                                                                           |
 | Most of this program is based on the ISACNeuralPID arm controller        |
 | developed by Ulutas et al.                                                |
 *****************************************************************************/

#include "Headers\stdafx.h"
#include <windows.h>
#include <windef.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <stdarg.h>
#include <math.h>
#include <iostream>
#include <sstream>
#include "Headers\winmotenc.h"
#include "Headers\Control.h"
#include "Headers\NNMuscleClass.h"
#include "Headers\PID.h"
#include "Headers\PracticalSocket.h"

using namespace std;


/*------------------------------< Arm Data >------------------------------*/

// Data for/from MOTENC-Lite DAQ cards
double leftValvesOutputs[12] = {0},
       leftValvesInputs[12] = {0},
       leftAngles[6] = {0};
long leftEncoders[6] = {0};

// Angle/trajectory data
float desiredangleL[6] = {0.0},
      desiredtrajectoryL[6] = {0},
      prevThetasL[6] = {0.0};

// Muscle objects--two (front and back) for each angle/axis/dof
CNNMuscles LeftNNMusclesAngle0F(1,0,1), LeftNNMusclesAngle0B(1,0,-1),
           LeftNNMusclesAngle1F(1,1,1), LeftNNMusclesAngle1B(1,1,-1),
           LeftNNMusclesAngle2F(1,2,1), LeftNNMusclesAngle2B(1,2,-1),
           LeftNNMusclesAngle3F(1,3,1), LeftNNMusclesAngle3B(1,3,-1),
           LeftNNMusclesAngle4F(1,4,1), LeftNNMusclesAngle4B(1,4,-1),
           LeftNNMusclesAngle5F(1,5,1), LeftNNMusclesAngle5B(1,5,-1);


/*-------------------------< UDP Communcation >-------------------------*/

const unsigned short PORT = 3999;
const int MAXLENGTH = 4096;
UDPSocket sock(PORT);


/*---------------------< Files for Experimental Data >---------------------*/

const char *filename = "C:/Bob/actuation/data/angledata.txt";
FILE *angleFile;


/*-------------------------< Program Functions >-------------------------*/

void LeftPID();
```

```
bool AppInit();
void AppExit();

/*****************************************************************************
 * Name:         main                                                        *
 * Description:  Main window program and message processing loop.            *
 *****************************************************************************/
int main()
{
    // variables for UDP communication
    char recvString[MAXLENGTH + 1]; // buffer for echo string + \0
    int numBytes;                   // the number of bytes recieved
    string srcAddr;                 // address of vision computer
    unsigned short srcPort;         // comm. port of vision computer

    int exitloop = 0;

    // call initialization procedure
    printf("Initializing system...\n");
    if(!AppInit())
    {
        exitloop = 1;
    }

    // If the user hasn't cancelled, begin following motions
    printf("\nActivating PID controller...\n");

    while(!exitloop && !kbhit())
    {
        // receive angle information from vision system
        numBytes = sock.recvFrom(recvString, MAXLENGTH, srcAddr, srcPort);
        recvString[numBytes] = '\0';  // Terminate string

        // scan recieved data for angles
        sscanf(recvString,"%f %f %f",&desiredtrajectoryL[0],
                                     &desiredtrajectoryL[1],
                                     &desiredtrajectoryL[2]);

        // look for end of program sentinel
        if (desiredtrajectoryL[2] == 9999)
        {
            exitloop = 1;
        }
        else
        {
            // Set the current desired angles to the current
            // trajectory angles
            for (int i = 0; i < 5; i++)
            {
                desiredangleL[i] = desiredtrajectoryL[i];
            }

            // actuate to the desired position
            LeftPID();

            // print the current joint angles to a file for later comparison
            fprintf(angleFile,"%f\t%f\t%f\n",leftAngles[0],leftAngles[1],
                                                           leftAngles[2]);
        }

        // echo confirmation to vision system
        sock.sendTo(recvString, strlen(recvString), srcAddr, srcPort);
    }

    // shutdown the application
    AppExit();
    return 0;
}
```

```
/*****************************************************************************
 * Name:         LeftPID                                                     *
 * Description:  This function actuates the left arm using a PID             *
 *               controller                                                  *
 *****************************************************************************/
void LeftPID()
{
    double LPID0_output = 0.0,
           LPID1_output = 0.0,
           LPID2_output = 0.0,
           LPID3_output = 0.0,
           LPID4_output = 0.0,
           LPID5_output = 0.0;

    // get the joint angle data
    ReadLeftEncoders();
    RealLeftAngles();

    // get the PID update based on the angle error
    LPID0_output = leftPID0(&desiredangleL[0]);
    LPID1_output = leftPID1(&desiredangleL[1]);
    LPID2_output = leftPID2(&desiredangleL[2]);
    LPID3_output = leftPID3(&desiredangleL[3]);
    LPID4_output = leftPID4(&desiredangleL[4]);
    LPID5_output = leftPID5(&desiredangleL[5]);

    // update the pressure in each muscle to move closer to desired angles
    leftValvesOutputs[0] = leftValvesOutputs[0] - LPID0_output;
    leftValvesOutputs[1] = leftValvesOutputs[1] + LPID0_output;

    leftValvesOutputs[2] = leftValvesOutputs[2] + LPID1_output;
    leftValvesOutputs[3] = leftValvesOutputs[3] - LPID1_output;

    leftValvesOutputs[4] = leftValvesOutputs[4] + LPID2_output;
    leftValvesOutputs[5] = leftValvesOutputs[5] - LPID2_output;
    leftValvesOutputs[6] = leftValvesOutputs[6] - LPID2_output;
    leftValvesOutputs[7] = leftValvesOutputs[7] + LPID2_output;

    leftValvesOutputs[4] = leftValvesOutputs[4] + LPID3_output;
    leftValvesOutputs[5] = leftValvesOutputs[5] + LPID3_output;
    leftValvesOutputs[6] = leftValvesOutputs[6] - LPID3_output;
    leftValvesOutputs[7] = leftValvesOutputs[7] - LPID3_output;

    leftValvesOutputs[8] = leftValvesOutputs[8] + LPID4_output;
    leftValvesOutputs[9] = leftValvesOutputs[9] - LPID4_output;
    leftValvesOutputs[10] = leftValvesOutputs[10] + LPID4_output;
    leftValvesOutputs[11] = leftValvesOutputs[11] - LPID4_output;

    leftValvesOutputs[8] = leftValvesOutputs[8] - LPID5_output;
    leftValvesOutputs[9] = leftValvesOutputs[9] - LPID5_output;
    leftValvesOutputs[10] = leftValvesOutputs[10] + LPID5_output;
    leftValvesOutputs[11] = leftValvesOutputs[11] + LPID5_output;

    SetLeftArmPressures();
}


/*****************************************************************************
 * Name:         AppInit                                                     *
 * Description:  Initialize all libraries and allocate all data needed      *
 *****************************************************************************/
bool AppInit()
{
    bool initialized = true;

    // Initialize DAQ cards and valves (set arms to "home" position)
    printf("\tDAQ cards...");
    if(!vitalInit())
```

```
    {
        MessageBox(0, "Unable to communicate with DAQ cards.",
                   "Left Arm Actuator", MB_OK);
        initialized = false;
    }
    printf("done!\n");

    // Initialize left arm muscles
    printf("\tleft arm muscles...");
    LeftNNMusclesAngle0F.Allocator(); LeftNNMusclesAngle0B.Allocator();
    LeftNNMusclesAngle1F.Allocator(); LeftNNMusclesAngle1B.Allocator();
    LeftNNMusclesAngle2F.Allocator(); LeftNNMusclesAngle2B.Allocator();
    LeftNNMusclesAngle3F.Allocator(); LeftNNMusclesAngle3B.Allocator();
    LeftNNMusclesAngle4F.Allocator(); LeftNNMusclesAngle4B.Allocator();
    LeftNNMusclesAngle5F.Allocator(); LeftNNMusclesAngle5B.Allocator();

    InitializeLeftValves();

    Sleep(3000);
    printf("done!\n");

    // Reset encoder values
    printf("\tencoder angles...");
    ResetLeftEncoders();
    ReadLeftEncoders();
    RealLeftAngles();

    for(int i = 1; i <= 6; i++)
    {
        prevThetasL[i] = leftAngles[i];
    }
    printf("done!\n");

    // Initialize experimental data file
    printf("\texperimental data file...");
    if(!(angleFile = fopen(filename, "w")))
    {
        MessageBox(0, "Unable to open data file.",
                   "Left Arm Actuator", MB_OK);
        initialized = false;
    }
    printf("done!\n");

    return initialized;
}


/*************************************************************************
 * Name:         AppExit                                                 *
 * Description:  Frees all allocated memory before exiting               *
 *************************************************************************/
void AppExit()
{
    // return the arms to the "home" position
    printf("\nReturning arm to home position...");
    TurnBackHomeLeftArm();
    printf("done!\n");

    // relax all muscles
    printf("\nSutting down...");
    CloseValves();
    Sleep(500);
    vitalQuit();

    // close the experimental data file
    if (angleFile)
        fclose(angleFile);
    printf("done!\n\n");
}
```

```
/***Control.h***/

#ifndef _Control_h
#define _Control_h

#include "stdafx.h"
#include <windows.h>
#include <windef.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <math.h>
#include <IOSTREAM>
#include "winmotenc.h"

// left arm data from main.cpp
extern double leftValvesOutputs[12];
extern double leftValvesInputs[12];
extern double leftAngles[6];
extern long leftEncoders[6];
double initialLeftValvesOutputs[12] = {2.3, 1.7, 2.0, 2.0,
                                       0.0, 2.2, 2.4, 0.0,
                                       2.3, 1.7, 1.7, 2.3};

void InitializeLeftValves()
{
    double bufferVoltage = 0.1;
    int i = 0,
        j = 0,
        k = 0;

    for(i = 0; i < 25; i++) // 25x100 ms = 2.5 sec makes the muscles blow up
    {
        // select board zero and set the valve voltages on each channel
        vitalSelectBoard(0);
        for(j = 0; j < 8; j++)
        {
            if(bufferVoltage <= initialLeftValvesOutputs[j] + 0.05)
            {
                vitalDacWrite(j, bufferVoltage);
            }
        }

        // select board one and set the valve voltages on each channel
        vitalSelectBoard(1);
        for(j = 8; j < 12; j++)
        {
            if(bufferVoltage <= initialLeftValvesOutputs[j] + 0.05)
            {
                vitalDacWrite(j - 8, bufferVoltage);
            }
        }

        // increase the reference voltage slowly so as not to harm the arms
        bufferVoltage += 0.1;
        Sleep(100);
    }

    for(int s = 0; s < 12; s++)
    {
        leftValvesOutputs[s] = initialLeftValvesOutputs[s];
    }

    vitalSelectBoard(0);
}

void SetLeftArmPressures()
{
```

```
    int channel = 0;

    vitalSelectBoard(0);
    for(channel = 0; channel < 8; channel++) // giving reference to each chan
nel of board 0
    {
        if (leftValvesOutputs[channel] < 0.0 )
        {
            leftValvesOutputs[channel] = 0.0;
        }
        if (leftValvesOutputs[channel] > 3.6 )
        {
            leftValvesOutputs[channel] = 3.6;
        }
        vitalDacWrite(channel, leftValvesOutputs[channel]);
    }

    vitalSelectBoard(1);
    for(channel = 0; channel < 4; channel++) // giving reference to first 4 c
hannel of board 1
    {
        if (leftValvesOutputs[channel + 8] < 0.0)
        {
            leftValvesOutputs[channel + 8] = 0.0;
        }
        if (leftValvesOutputs[channel + 8] > 3.6)
        {
            leftValvesOutputs[channel + 8] = 3.6;
        }
        vitalDacWrite(channel, leftValvesOutputs[channel + 8]);
    }
}

void CloseValves()
{
    double bufferVoltage = 0.1;
    int i = 0,
        j = 0,
        k = 0;

    for(i = 0; i < 36; i++) // 36x100 ms = 3.6 sec makes the muscles blow up
    {
        vitalSelectBoard(0);
        for(j = 0; j < 8; j++) // giving reference to each channel of board 0
        {
            if(leftValvesOutputs[j] - bufferVoltage > 0)
            {
                vitalDacWrite(j, leftValvesOutputs[j] - bufferVoltage);
            }
        }

        vitalSelectBoard(1);
        for(j = 0; j < 4; j++) // giving reference to first 4 channel of boar
d 1
        {
            if(leftValvesOutputs[j + 8] - bufferVoltage > 0)
            {
                vitalDacWrite(j, leftValvesOutputs[j + 8] - bufferVoltage);
            }
        }

        bufferVoltage += .1;
        Sleep(200); // Wait 100 ms at each step
    }
}

void ReadLeftEncoders()
{
    vitalSelectBoard(0);
```

```
    for(int channel = 0; channel < 4; channel++)
    {
        vitalEncoderRead(channel, &leftEncoders[channel]);
    }

    vitalSelectBoard(1);
    for(channel = 0; channel < 2; channel++)
    {
        vitalEncoderRead(channel, &leftEncoders[channel+4]);
    }

    leftEncoders[2] *= -1;
    leftEncoders[5] *= -1;
}

void ResetLeftEncoders()
{
    vitalSelectBoard(0);
    for(int channel = 0; channel < 4; channel++)
    {
        vitalResetCounter(channel);
    }

    vitalSelectBoard(1);
    for(channel = 0; channel < 2; channel++)
    {
        vitalResetCounter(channel);
    }
}

void RealLeftAngles()
{
    //Encoder Parameters =  -5092 5092 -4244 4244 636.6 -636.6
    leftAngles[0] =  (double) leftEncoders[0] / -5092.0 / 6.28 * 360;
    leftAngles[1] =  (double) -1*leftEncoders[1] / 5092.0 / 6.28 * 360 + 90;
    leftAngles[2] =  ((double) leftEncoders[3] / -4244.0 / 2 / 6.28 * 360
                      - (double) leftEncoders[2] / 4244 /2 /6.28 * 360) -180;
    leftAngles[3] =  ((double) -leftEncoders[3] / - 4244.0 / 6.28 * 360
                      - (double) leftEncoders[2] /  4244 /6.28 * 360 );
    leftAngles[4] =  ((double) -leftEncoders[4] / 636.6 / 2 / 6.28 * 360
                      - (double) leftEncoders[5] / 636.6 /2 /6.28 * 360);
    leftAngles[5] =  ((double) leftEncoders[4] / - 636.6 / 6.28 * 360
                      + (double) leftEncoders[5] / 636.6 /6.28 * 360);
}

void TurnBackHomeLeftArm()
{
    double bufferVoltage[12] = {0.0},
           steps = 25;

    int i = 0,
        j = 0,
        k = 0;

    for(i = 1; i < 12; i++)
    {
        bufferVoltage[i] = (initialLeftValvesOutputs[i] - leftValvesOutputs[i
]) / steps;
    }

    for (j = 0; j < steps; j++)
    {
        vitalSelectBoard(0);
        for (i = 0; i < 8; i++)
        {
            vitalDacWrite(i, leftValvesOutputs[i] + bufferVoltage[i] * j);
        }

        vitalSelectBoard(1);
```

```
        for (i = 0; i < 4; i++)
        {
            vitalDacWrite(i, leftValvesOutputs[i + 8] + bufferVoltage[i + 8]
* j);
        }
        Sleep(100);
    }

    for(int s = 0; s < 12; s++)
    {
        leftValvesOutputs[s] = initialLeftValvesOutputs[s];
    }

    Sleep(300);

}

#endif
```

```
/***PID.h***/

#ifndef _PID_h
#define _PID_h

extern double leftAngles[6];

// P controller function for angle 0 of the left arm
double leftPID0(float *desiredAngle0)
{
    double anglenow, angleerror;    // joint angle data
    double P = 0.0036;              // proportion constant

    // read the current angle
    anglenow = leftAngles[0];

    // calculate the current error
    angleerror = *desiredAngle0 - anglenow;

    // return the controlled amount of angle to compensate
    return P * angleerror;
}

// P controller function for angle 1 of the left arm
double leftPID1(float *desiredAngle1)
{
    double anglenow, angleerror;
    double P = 0.0052;

    anglenow = leftAngles[1];

    angleerror = *desiredAngle1 - anglenow;

    return P * angleerror;
}

// P controller function for angle 2 of the left arm
double leftPID2(float *desiredAngle2)
{
    double anglenow, angleerror;
    double P = 0.006;

    anglenow = leftAngles[2];

    angleerror = *desiredAngle2 - anglenow;

    return P * angleerror;
}

// P controller function for angle 3 of the left arm
double leftPID3(float *desiredAngle3)
{
    double anglenow, angleerror;
    double P = 0.003;

    anglenow = leftAngles[3];

    angleerror = *desiredAngle3 - anglenow;

    return P * angleerror;
}

// P controller function for angle 4 of the left arm
double leftPID4(float *desiredAngle4)
{
    double anglenow, angleerror;
    double P = 0.0015;

    anglenow = leftAngles[4];
```

```
    angleerror = *desiredAngle4 - anglenow;

    return P * angleerror;
}

// P controller function for angle 5 of the left arm
double leftPID5(float *desiredAngle5)
{
    double anglenow, angleerror;
    double P = 0.003;

    anglenow = leftAngles[5];

    angleerror = *desiredAngle5 - anglenow;

    return P * angleerror;
}

#endif
```

```
/***winmotenc.h***/

#ifndef __VITAL_H
#define __VITAL_H


// Number of axes in system
#define VITAL_MAX_AXIS 8

// flags defined bit-exclusive for OR'ing if board can do multiple ways
#define EXT_ENCODER_INDEX_MODEL_MANUAL 0x00000001
#define EXT_ENCODER_INDEX_MODEL_AUTO   0x00000002

long __stdcall vitalInit(void);  //returns number of boards in system
long __stdcall vitalSelectBoard( long BoardID );    //0,1,2,3...15

long __stdcall vitalAioReset(void); //set all DACs ro zero
long __stdcall vitalQuit(void);

long __stdcall vitalGetMaxEncoders(void);
long __stdcall vitalEncoderRead(int axis, long * pCounts);
long __stdcall vitalResetCounter( int nAxis );

long __stdcall vitalReadIndexLevel(int nAxis, int * flag);
long __stdcall vitalEncoderIndexModel(void);
long __stdcall vitalEncoderSetIndexModel(unsigned int model);
long __stdcall vitalEncoderResetIndex(int nAxis);
long __stdcall vitalEncoderReadLatch(int nAxis, int * flag);

long __stdcall vitalDacNum(void);
long __stdcall vitalDacWrite(int nAxis, double volts);
long __stdcall vitalDacWriteAll(int max, double * volts);

long __stdcall vitalReadAnalogInputs(unsigned long nBank, double *value);

long __stdcall vitalDioWrite(int index, int value);
long __stdcall vitalDioReset(void); // Turn off all outputs (open collector)
long __stdcall vitalDioCheck(int index, int *value);
long __stdcall vitalDioRead(int index, int *value);

// Read or Write Direct 32bit Registers on the MOTENC Board
long __stdcall vitalReadReg(int nReg, long * pRegData);
long __stdcall vitalWriteReg(int nReg, long RegData);

// Analog Binary Data
long __stdcall vitalAdcReadBinary(unsigned long nBank, int * AdcBinaryData);
long __stdcall vitalDacWriteBinary(int nAxis, int BinData);


#endif /* __VITAL_H */
```

```
// Train.h: interface for the CTrain class.
//
//////////////////////////////////////////////////////////////////////
#ifndef  NNMuscleClass h
#define _NNMuscleClass_h

#include <iostream>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <malloc.h>
#include <time.h>

class CNNMuscles
{
public:
    CNNMuscles(int leftrightarm, int anglenumber, int forwardbackward);
    virtual ~CNNMuscles();
    void Allocator(void); //
    void Input(void); //
    void Process(float *input,float *output); //
    void Process();
    void WeightsWrite(void); //
    void IsExist(void);

    int forwardbackward;
    int leftrightarm;
    int anglenumber;

private:
    int i, j, k, t, n;
    long epochs;
    int nNeuron;
    int nInput;

    FILE *TrainData,*weights,*hata;

    float wpresent_th[2];//random
    float wpast_th[2];//random
    float wfuture_th[2];//random
    float y_in[2];
    float delta[2];
    float deltaw_th[2];

    int m;
    double alfa;//0.001;////0.00001;////0.001 ;//learning rate
    float mm;
    float *T1;
    float y1;
    float a;

    float **inputx;
    float *z_in;
    float *z;
    float *delta_in;
    float *delta_hidden;
    float *deltav_th;
    float **deltav;
    float **deltaw;
    float **wpresent;
    float **wpast;
    float **wfuture;
    float *vpresent_th;
    float *vpast_th;
    float **vpresent;
    float **vpast;
    float **vfuture;
    float *vfuture_th;
```

```
};
#endif
```

```
// CNNMuscleClass.cpp: implementation of the CNNMuscles class.
//
//////////////////////////////////////////////////////////////////////

#include "Headers\NNMuscleClass.h"
#include <iostream>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <malloc.h>
#include <time.h>

#define PI 3.141592653589
#define Rate 8

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CNNMuscles::CNNMuscles(int leftrightarmvalue, int anglenumbervalue, int forwa
rdbackwardvalue)
{
    leftrightarm=  leftrightarmvalue; // if leftrightarmvalue = 1 --> left ar
m
    anglenumber = anglenumbervalue; // if anglevalue = 0 --> angle0 & =1 -->
Angle1 ...6
    forwardbackward = forwardbackwardvalue; // if forwardbackwardvalue = 1 --
>forward
}

CNNMuscles::~CNNMuscles()
{

    delete vfuture th;
    delete vpast th;
    delete vpresent_th;
    delete z;
    delete z_in;

    for(j=0;j<nNeuron;j++)
    {
        free(*(wpresent+j));
        free(*(wpast+j));
        free(*(wfuture+j));
        free(*(deltaw+j));
    }
    free(wpresent);
    free(wpast);
    free(wfuture);
    free(deltaw);

    for(i=0;i<nInput;i++)
    {
        free(*(vpresent+i));
        free(*(vpast+i));
        free(*(vfuture+i));
        free(*(deltav+i));
    }
    free(vpresent);
    free(vpast);
    free(vfuture);
    free(deltav);

    for(t=0;t<m;t++)
    free(*(inputx+t));
    free(inputx);
}
```

```
void CNNMuscles::Allocator(void)
{

    using namespace std;
    nInput = 1;nNeuron=50;
    int sillybuffer1;
    float sillybuffer2;

    FILE *weights;

    // Left Arm Files
    if (leftrightarm ==1 && anglenumber == 0 && forwardbackward == 1)
        if((weights = fopen(".\\NNWeights\\LeftNNmuscle01FWeights.txt","r"))
== NULL)
            printf("Can't Open File LeftNNmuscle01FWeights.txt\n");
    if (leftrightarm ==1 && anglenumber == 0 && forwardbackward == -1)
        if((weights = fopen(".\\NNWeights\\LeftNNmuscle01BWeights.txt","r"))
== NULL)
            printf("Can't Open File LeftNNmuscle01FWeights.txt\n");

    if (leftrightarm ==1 && anglenumber == 1 && forwardbackward == 1)
        if((weights = fopen(".\\NNWeights\\LeftNNmuscle23FWeights.txt","r"))
== NULL)
            printf("Can't Open File LeftNNmuscle23FWeights.txt\n");
    if (leftrightarm ==1 && anglenumber == 1 && forwardbackward == -1)
        if((weights = fopen(".\\NNWeights\\LeftNNmuscle23BWeights.txt","r"))
== NULL)
            printf("Can't Open File LeftNNmuscle23BWeights.txt\n");

    if (leftrightarm ==1 && anglenumber == 2 && forwardbackward == 1)
        if((weights = fopen(".\\NNWeights\\LeftNNmuscle4567Angle2FWeights.txt
","r")) == NULL)
            printf("Can't Open File LeftNNmuscle4567Angle2FWeights.txt\n");
    if (leftrightarm ==1 && anglenumber == 2 && forwardbackward == -1)
        if((weights = fopen(".\\NNWeights\\LeftNNmuscle4567Angle2BWeights.txt
","r")) == NULL)
            printf("Can't Open File LeftNNmuscle4567Angle2BWeights.txt\n");

    if (leftrightarm ==1 && anglenumber == 3 && forwardbackward == 1)
        if((weights = fopen(".\\NNWeights\\LeftNNmuscle4567Angle3FWeights.txt
","r")) == NULL)
            printf("Can't Open File LeftNNmuscle4567Angle3FWeights.txt\n");
    if (leftrightarm ==1 && anglenumber == 3 && forwardbackward == -1)
        if((weights = fopen(".\\NNWeights\\LeftNNmuscle4567Angle3BWeights.txt
","r")) == NULL)
            printf("Can't Open File LeftNNmuscle4567Angle3BWeights.txt\n");

    if (leftrightarm ==1 && anglenumber == 4 && forwardbackward == 1)
        if((weights = fopen(".\\NNWeights\\LeftNNmuscle891011Angle4FWeights.t
xt","r")) == NULL)
            printf("Can't Open File LeftNNmuscle891011Angle4FWeights.txt\n");
    if (leftrightarm ==1 && anglenumber == 4 && forwardbackward == -1)
        if((weights = fopen(".\\NNWeights\\LeftNNmuscle891011Angle4BWeights.t
xt","r")) == NULL)
            printf("Can't Open File LeftNNmuscle891011Angle4BWeights.txt\n");

    if (leftrightarm ==1 && anglenumber == 5 && forwardbackward == 1)
        if((weights = fopen(".\\NNWeights\\LeftNNmuscle891011Angle5FWeights.t
xt","r")) == NULL)
            printf("Can't Open File LeftNNmuscle891011Angle4FWeights.txt\n");
    if (leftrightarm ==1 && anglenumber == 5 && forwardbackward == -1)
        if((weights = fopen(".\\NNWeights\\LeftNNmuscle891011Angle5BWeights.t
xt","r")) == NULL)
            printf("Can't Open File LeftNNmuscle891011Angle5BWeights.txt\n");

    // Right Arm Files
    if (leftrightarm ==0 && anglenumber == 0 && forwardbackward == 1)
        if((weights = fopen(".\\NNWeights\\RightNNmuscle01FWeights.txt","r"))
 == NULL)
```

```
            printf("Can't Open File RightNNmuscle01FWeights.txt\n");
    if (leftrightarm ==0 && anglenumber == 0 && forwardbackward == -1)
        if((weights = fopen(".\\NNWeights\\RightNNmuscle01BWeights.txt","r"))
 == NULL)
            printf("Can't Open File RightNNmuscle01FWeights.txt\n");

    if (leftrightarm ==0 && anglenumber == 1 && forwardbackward == 1)
        if((weights = fopen(".\\NNWeights\\RightNNmuscle23FWeights.txt","r"))
 == NULL)
            printf("Can't Open File RightNNmuscle23FWeights.txt\n");
    if (leftrightarm ==0 && anglenumber == 1 && forwardbackward == -1)
        if((weights = fopen(".\\NNWeights\\RightNNmuscle23BWeights.txt","r"))
 == NULL)
            printf("Can't Open File RightNNmuscle23BWeights.txt\n");

    if (leftrightarm ==0 && anglenumber == 2 && forwardbackward == 1)
        if((weights = fopen(".\\NNWeights\\RightNNmuscle4567Angle2FWeights.tx
t","r")) == NULL)
            printf("Can't Open File RightNNmuscle4567Angle2FWeights.txt\n");
    if (leftrightarm ==0 && anglenumber == 2 && forwardbackward == -1)
        if((weights = fopen(".\\NNWeights\\RightNNmuscle4567Angle2BWeights.tx
t","r")) == NULL)
            printf("Can't Open File RightNNmuscle4567Angle2BWeights.txt\n");

    if (leftrightarm ==0 && anglenumber == 3 && forwardbackward == 1)
        if((weights = fopen(".\\NNWeights\\RightNNmuscle4567Angle3FWeights.tx
t","r")) == NULL)
            printf("Can't Open File RightNNmuscle4567Angle3FWeights.txt\n");
    if (leftrightarm ==0 && anglenumber == 3 && forwardbackward == -1)
        if((weights = fopen(".\\NNWeights\\RightNNmuscle4567Angle3BWeights.tx
t","r")) == NULL)
            printf("Can't Open File RightNNmuscle4567Angle3BWeights.txt\n");

    if (leftrightarm ==0 && anglenumber == 4 && forwardbackward == 1)
        if((weights = fopen(".\\NNWeights\\RightNNmuscle891011Angle4FWeights.
txt","r")) == NULL)
            printf("Can't Open File RightNNmuscle891011Angle4FWeights.txt\n")
;
    if (leftrightarm ==0 && anglenumber == 4 && forwardbackward == -1)
        if((weights = fopen(".\\NNWeights\\RightNNmuscle891011Angle4BWeights.
txt","r")) == NULL)
            printf("Can't Open File RightNNmuscle891011Angle4BWeights.txt\n")
;

    if (leftrightarm ==0 && anglenumber == 5 && forwardbackward == 1)
        if((weights = fopen(".\\NNWeights\\RightNNmuscle891011Angle5FWeights.
txt","r")) == NULL)
            printf("Can't Open File RightNNmuscle891011Angle4FWeights.txt\n")
;
    if (leftrightarm ==0 && anglenumber == 5 && forwardbackward == -1)
        if((weights = fopen(".\\NNWeights\\RightNNmuscle891011Angle5BWeights.
txt","r")) == NULL)
            printf("Can't Open File RightNNmuscle891011Angle5BWeights.txt\n")
;

    fscanf(weights,"%d\n",&nNeuron);
    fscanf(weights,"%f\n",&sillybuffer2);
    fscanf(weights,"%f\n",&sillybuffer2);
    fscanf(weights,"%d\n",&sillybuffer1);
    fscanf(weights,"%d\n",&sillybuffer1);
    fscanf(weights,"%d\n",&nInput);

    srand( (unsigned)time( NULL ) );

    //1*10
    z_in = (float*)malloc(nNeuron*sizeof(float));

    //1*10
    z = (float*)malloc(nNeuron*sizeof(float));
```

```
    //1*10
    delta_in = (float*)malloc(nNeuron*sizeof(float));

    //1*10
    delta_hidden = (float*)malloc(nNeuron*sizeof(float));

    //1*10
    deltav_th = (float*)malloc(nNeuron*sizeof(float));

    // 14*10
    deltav = (float**)malloc(nInput*sizeof(float*));
    for(i=0;i<nInput;i++)
        *(deltav+i) = (float*)malloc(nNeuron*sizeof(float));

    //10*2
    deltaw = (float**)malloc(nNeuron*sizeof(float*));
    for(i=0;i<nNeuron;i++)
        *(deltaw+i) = (float*)malloc(2*sizeof(float));

    //random 10*2
    wpresent = (float**)malloc(nNeuron*sizeof(float*));
    for(i=0;i<nNeuron;i++)
        *(wpresent+i) = (float*)malloc(2*sizeof(float));

    //random 10*2
    wpast = (float**)malloc(nNeuron*sizeof(float*));
    for(i=0;i<nNeuron;i++)
        *(wpast+i) = (float*)malloc(2*sizeof(float));

    //random 10*2
    wfuture = (float**)malloc(nNeuron*sizeof(float*));
    for(i=0;i<nNeuron;i++)
        *(wfuture+i) = (float*)malloc(2*sizeof(float));

    //random 1*10
    vpresent_th = (float*)malloc(nNeuron*sizeof(float));

    //random 1*10
    vpast_th = (float*)malloc(nNeuron*sizeof(float));

    //random 1*10
    vfuture_th = (float*)malloc(nNeuron*sizeof(float));

    //random 14*10
    vpresent = (float**)malloc(nInput*sizeof(float*));
    for(i=0;i<nInput;i++)
        *(vpresent+i) = (float*)malloc(nNeuron*sizeof(float));

    //random 14*10
    vpast = (float**)malloc(nInput*sizeof(float*));
    for(i=0;i<nInput;i++)
        *(vpast+i) = (float*)malloc(nNeuron*sizeof(float));

    //random 14*10
    vfuture = (float**)malloc(nInput*sizeof(float*));
    for(i=0;i<nInput;i++)
        *(vfuture+i) = (float*)malloc(nNeuron*sizeof(float));

    m=1;
    //**********************READING WEIGHTS NINPUTS AND NEURONS*************
**************

    for (k=0;k<1;k++)
    {
        for (j=0;j<nNeuron;j++)
            fscanf(weights,"%f\n",&wfuture[j][k]);
        fscanf(weights,"%f\n",&wfuture_th[k]);
    }
```

```
    for(j=0;j<nNeuron;j++)
    {
        for(i=0;i<nInput;i++)
            fscanf(weights,"%f\n",&vfuture[i][j]);
    }

    for(j=0;j<nNeuron;j++)
        fscanf(weights,"%f\n",&vfuture_th[j]);

    //***************END OF READING*****************************************
**

    inputx = (float**)malloc(m*sizeof(float*));
    for(i=0;i<m;i++)
        *(inputx+i) = (float*)malloc(nInput*sizeof(float));

    fclose(weights);
}

void CNNMuscles::Process(float *input,float *output)
{
    inputx[0][0] = *input;

    if (leftrightarm ==1 && anglenumber == 0)
        // Here Normalizing the Angles. Forward changes between -65 & +25 to
0-1
        inputx[0][0] = (inputx[0][0] + 65) / 90;
    if (leftrightarm ==0 && anglenumber == 0)
        // Here Normalizing the Angles. Forward changes between -25 & +55 to
0-1
        inputx[0][0] = (inputx[0][0] + 25) / 80;

    if (anglenumber == 1 )
        // Normalizing the input 40-125 to 0-1
        inputx[0][0] = (inputx[0][0] - 40) / 85;
    if (anglenumber == 2)
        // Normalizing the input -215 & -130 to 0-1
        inputx[0][0] = (inputx[0][0] + 215) / 75;
    if (anglenumber == 3)
        // Normalizing the input -40 & 30 to 0-1
        inputx[0][0] = (inputx[0][0] + 40) / 70;
    if (anglenumber == 4)
        // Normalizing the input -40 && 40 to 0-1
        inputx[0][0] = (inputx[0][0] +40) / 80;
    if (anglenumber == 5)
        // Normalizing the input -70 & 70 to 0-1
        inputx[0][0] = (inputx[0][0] +70)/ 140;

    for(t=0;t<1;t++) // m =1 m is the data number
    {
        y_in[0] = wfuture_th[0];
        for(j=0;j<nNeuron;j++)
        {
            z_in[j] = vfuture_th[j];
            for(i=0;i<1;i++)    // nInput yerine 1 yazýyoruz , nInput data sa
yýsý
                z_in[j] += inputx[t][i] * vfuture[i][j];
                z[j]= 2/(1+exp(-2*z_in[j]))-1;// activation function: tansig
  n = 2/(1+exp(-2*n))-1
                y_in[0] += z[j] * wfuture[j][0];
        }
            *output = y_in[0];
    }
}
```

```
/*************************************************************************
 |kinematics.cpp                                                Sean Thornton|
 |-----------------------------------------------------------------------|
 | This program takes the arm angle data files generated in each trial and |
 | converts the data to 3-dimensional cartesian points using a forward     |
 | kinematics for ISAC's left arm relative to his face.                    |
 |                                                                         |
 | Portions of this program are based on MahirKinematics.cpp, part of the  |
 | ISACNeuralPID project created by Ulutas et al.                          |
 *************************************************************************/

#include "stdafx.h"
#include <math.h>
#include <iostream>

using namespace std;

/*-------------------------< Program Constants >---------------------------*/

// D-H parameters
const double A[4] = {246, 0, 325, 290};
const double D[4] = {0, -190, 200, 0};
const double ALPHA[4] = {0, -90, 0, 90};
const double THETA0 = 90;

// degree<->radian conversion
const double PI = 3.14159265359;
const double R2D = (180.0 / PI);

/*---------------------< Files for Experimental Data >---------------------*/

FILE *angleFile, *cartesianFile;
const char *angleFilePath = "e:/documents/thesis/actuation/data/";
const char *cartesianFilePath = "e:/documents/thesis/actuation/data/3d/";
const char *fileExt = ".txt";

/*-------------------------< Program Functions >---------------------------*/

double** transformMatrix(double a, double d, double alpha, double theta);
double** mxMpy(double **mat1, double **mat2);

/*************************************************************************
 * Name:         main                                                    *
 * Description:  Main window program and processing loop.                *
 *************************************************************************/
int main(int argc, char *argv[])
{
    char fileName[100];
    char trial[20];
    char motion[20];
    char speed[10];
    char num[2];
    double *angles = new double[6];
    double **T01, **T12, **T23, **T34,
           **T02, **T03, **T04;
    double theta1, theta2, theta3;

    printf("Processing files...\n");

    // for every motion type,
    for (int i = 0; i < 4; i++)
    {
        // place that type in a string;
        if (i == 0)
            strcpy(motion, "diagonal");
        else if (i == 1)
            strcpy(motion, "horizontal");
        else if (i == 2)
            strcpy(motion, "sagittal");
```

```
else
    strcpy(motion, "verical");

printf("\t%s...", motion);

// for every speed,
for (int j = 0; j < 3; j++)
{
    // place that speed in a string;
    if (j == 0)
        strcpy(speed, "high");
    else if (j == 1)
        strcpy(speed, "med");
    else
        strcpy(speed, "slow");

    // and for every trial,
    for (int k = 1; k <= 5; k++)
    {
        // construct the input file name: path/motionspeedtrial#.txt
        strcpy(trial, motion);
        strcat(trial, speed);
        strcat(trial, itoa(k,num,10));
        strcat(trial, fileExt);

        strcpy(fileName, angleFilePath);
        strcat(fileName, trial);

        // open the file
        if (!(angleFile = fopen(fileName, "r")))
        {
            printf("Can't open %s!\n", fileName);
            return 1;
        }

        // construct the output data file name
        strcpy(fileName, cartesianFilePath);
        strcat(fileName, trial);

        // open the output data file
        if (!(cartesianFile = fopen(fileName, "w")))
        {
            printf("Can't open %s!\n", fileName);
            return 1;
        }

        // as long as there are joint angles in the input file,
        while (!feof(angleFile))
        {
            // read the angles,
            fscanf(angleFile, "%lf\t%lf\t%lf\n", &theta1,  &theta2,
                   &theta3);

            // modify to match the definitions of this kinematics
            theta1 -= 90;
            theta3 += 90;

            // build the transform matrices
            T01 = transformMatrix(A[0],D[0],ALPHA[0],THETA0);
            T12 = transformMatrix(A[1],D[1],ALPHA[1],theta1);
            T23 = transformMatrix(A[2],D[2],ALPHA[2],theta2);
            T34 = transformMatrix(A[3],D[3],ALPHA[3],theta3);
            T02 = mxMpy(T01, T12);
            T03 = mxMpy(T02, T23);
            T04 = mxMpy(T03, T34);

            // save the end-effector coordinates to the output file
            fprintf(cartesianFile, "%f\t%f\t%f\n", T04[0][3],
                    T04[1][3],T04[2][3]);
```

```
                }

                // close the files
                fclose(angleFile);
                fclose(cartesianFile);
            }
        }

        printf("done!\n");
    }

    return 0;
}

/*****************************************************************************
 * Name:         transformMatrix                                             *
 * Description:  Constructs a D-H transform matrix based on the input        *
 *               parameters                                                  *
 *****************************************************************************/
double** transformMatrix(double a, double d, double alpha, double theta)
{
    // allocate a 4x4 array of doubles
    double** matrix = new double*[4];
    for (int i = 0; i < 4; i++)
        matrix[i] = new double[4];

    // convert input angles to radians (for sin and cos)
    alpha /= R2D;
    theta /= R2D;

    // populate the elements of the transformation matrix
    matrix[0][0] = cos(theta);
    matrix[0][1] = -sin(theta)*cos(alpha);
    matrix[0][2] = sin(theta)*sin(alpha);
    matrix[0][3] = a*cos(theta);

    matrix[1][0] = sin(theta);
    matrix[1][1] = cos(theta)*cos(alpha);
    matrix[1][2] = -cos(theta)*sin(alpha);
    matrix[1][3] = a*sin(theta);

    matrix[2][0] = 0;
    matrix[2][1] = sin(alpha);
    matrix[2][2] = cos(alpha);
    matrix[2][3] = d;

    matrix[3][0] = 0;
    matrix[3][1] = 0;
    matrix[3][2] = 0;
    matrix[3][3] = 1;

    // return the pointer to the matrix
    return matrix;
}

/*****************************************************************************
 * Name:         mxMpy                                                       *
 * Description:  Multiplies two 4x4 matrices together                        *
 *****************************************************************************/
double** mxMpy(double **mat1, double **mat2)
{
    // create a 4x4 array of doubles
    double **result = new double*[4];

    // for each row in the array
    for(int i = 0; i < 4; i++)
    {
        // allocate the column
        result[i] = new double[4];
```

```
        // for each element in the column
        for(int j = 0; j < 4; j++)
        {
            // perform matrix multiplication
            result[i][j] = mat1[i][0] * mat2[0][j]
                         + mat1[i][1] * mat2[1][j]
                         + mat1[i][2] * mat2[2][j]
                         + mat1[i][3] * mat2[3][j];
        }
    }

    // return the pointer to the array
    return result;
}
```

APPENDIX B.

EXHAUSTIVE RESULTS

The following pages contain a full listing of all trajectory points detected by the human to be imitated and executed by ISAC, all of the plots thereof which were used to verify the success of the experiments for reference.

Vertical motion at "Low" speed

Human

Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
3446371	20322.58	26419.36	3237500	19303.03	35000	3502869	20081.97	26049.18	3502869	19622.95	31213.12	3621610	20288.13	30491.53
3446371	20548.39	26419.36	3502869	20770.49	27885.25	3502869	20081.97	26049.18	3561250	20066.67	31616.67	3621610	20050.85	30610.17
3446371	20548.39	26419.36	3446371	20548.39	27435.48	3561250	20533.33	26483.33	3561250	20066.67	31500	3502869	19393.44	29606.56
3052500	19000	31800	3502869	20770.49	27540.98	3391667	19888.89	25000	3621610	20525.42	31915.25	3502869	19393.44	29491.8
3446371	20774.19	26080.64	3446371	20548.39	26645.16	3391667	19666.67	24888.89	3502869	19852.46	30524.59	3561250	19833.33	29983.33
3561250	21466.67	26483.33	3446371	20322.58	25629.03	3446371	20096.77	25177.42	3561250	20300	30916.67	3502869	19393.44	29377.05
3446371	20548.39	25516.13	3391667	19666.67	25222.22	3446371	20096.77	25290.32	3446371	19645.16	29806.45	3502869	19393.44	29377.05
3502869	21000	25475.41	3009507	18436.62	30957.75	3446371	20096.77	25064.52	3502869	19852.46	30065.57	3561250	19833.33	29516.67
3391667	20333.33	24333.33	3446371	20096.77	25290.32	3446371	20096.77	24951.61	3446371	19645.16	29467.74	3621610	20288.13	29779.66
3502869	20770.49	25016.39	3502869	20540.98	25360.66	3502869	20311.47	25016.39	3561250	20300	30333.33	3502869	19622.95	28573.77
3561250	21233.33	25316.67	3446371	20322.58	24838.71	3391667	19666.67	24111.11	3446371	19645.16	29129.03	3446371	19193.55	28000
3502869	21000	24672.13	3502869	20770.49	25016.39	3446371	20096.77	24387.1	3502869	19852.46	29491.8	3621610	20288.13	29186.44
3502869	20770.49	24442.62	3446371	20322.58	24387.1	3502869	20311.47	24672.13	3446371	19645.16	29129.03	3446371	19193.55	27548.39
3502869	20770.49	24213.12	3502869	21000	24672.13	3391667	19666.67	23444.45	3502869	20081.97	29262.29	3502869	19393.44	27885.25
3391667	19888.89	23111.11	3561250	21233.33	24733.33	3502869	20311.47	24327.87	3502869	19852.46	29377.05	3446371	19193.55	27096.77
3561250	20766.67	24033.33	3502869	21000	23983.61	3446371	20096.77	23935.48	3446371	19645.16	28790.32	3502869	19393.44	27426.23
3391667	20111.11	22666.67	3446371	20096.77	23258.06	3446371	20096.77	23709.68	3502869	19852.46	29032.79	3621610	20050.85	27881.36
3446371	20096.77	22919.36	3502869	20540.98	23295.08	3391667	19666.67	23333.33	1494231	12580.42	11797.2	3561250	19600	27300
3446371	20322.58	22806.45	3561250	21000	23566.67	3502869	20311.47	23639.34	1473621	12310.34	11537.93	3502869	19393.44	26737.71
3561250	20766.67	23333.33	3561250	21000	23333.33	3502869	20081.97	23754.1	3502869	19852.46	28573.77	3502869	19393.44	26163.93
3446371	20322.58	22354.84	3561250	21000	22983.33	3446371	19870.97	23258.06	3502869	20081.97	28459.02	3502869	19622.95	25704.92
3502869	20540.98	22491.8	3391667	20111.11	21666.67	3446371	19645.16	22919.36	3391667	19000	27444.45	3502869	19393.44	25360.66
3391667	19888.89	21555.55	3502869	21000	22262.29	3502869	20081.97	23180.33	3502869	19393.44	28229.51	3502869	19393.44	24901.64
3561250	21000	22516.67	3561250	21233.33	22166.67	3502869	20081.97	22950.82	3391667	19000	27333.33	3502869	19622.95	24786.88
3684052	21482.76	22931.04	3684052	21724.14	22810.35	3502869	20081.97	22721.31	3391667	19000	27000	3561250	20066.67	24733.33
3561250	21000	21933.33	3502869	21000	21573.77	3502869	20081.97	22721.31	3446371	19193.55	27435.48	3561250	19833.33	24616.67
3561250	21233.33	21583.33	3502869	20770.49	21344.26	3446371	20096.77	22016.13	3502869	19852.46	27540.98	3561250	20066.67	24266.67
3446371	20548.39	20887.1	3502869	21000	21114.75	3391667	19666.67	21444.45	3446371	19645.16	26983.87	3446371	19193.55	23370.97
3502869	21000	20885.25	3502869	20770.49	20885.25	3502869	20540.98	22032.79	3502869	19852.46	27311.47	3561250	20066.67	23916.67
3446371	20548.39	20322.58	3446371	20774.19	20435.48	3502869	20540.98	21918.03	3391667	19444.45	26111.11	3621610	20288.13	24203.39
3446371	20548.39	19983.87	3502869	20770.49	20540.98	3561250	20766.67	22166.67	3338672	19031.25	25703.13	3561250	19833.33	23333.33
3502869	21000	20196.72	3502869	20770.49	20426.23	3446371	20322.58	21225.81	3502869	20081.97	26737.71	3561250	19833.33	23450
3561250	21233.33	20183.33	3561250	21233.33	20416.67	3502869	20540.98	21573.77	3391667	19444.45	25555.55	3621610	20288.13	23372.88
3561250	21466.67	20066.67	3561250	21233.33	20300	3502869	20540.98	21229.51	3446371	19645.16	26080.64	3561250	20066.67	22750
1978472	15037.04	16722.22	3621610	21711.87	20288.13	3391667	19888.89	20222.22	3561250	20300	26716.67	3621610	20288.13	23135.59
3561250	21233.33	19600	3502869	20770.49	19622.95	3446371	20548.39	20548.39	3391667	19222.22	25444.45	3561250	20066.67	22400
3561250	21466.67	19133.33	3502869	21000	19393.44	3502869	20540.98	20885.25	3446371	19193.55	25629.03	3561250	20066.67	22283.33
1925000	14693.69	15702.7	3502869	20770.49	19163.93	3446371	20322.58	20209.68	3391667	19000	25000	3621610	20288.13	22423.73
3561250	21466.67	18666.67	3446371	20322.58	18516.13	3502869	20540.98	20655.74	3391667	19000	24666.67	3621610	20525.42	22186.44
3621610	21949.15	18864.41	3561250	21000	19250	3446371	20096.77	20096.77	3502869	19393.44	25475.41	3621610	20525.42	21949.15
3561250	21700	18433.33	3502869	20540.98	18704.92	3446371	20322.58	19983.87	3502869	19622.95	25360.66	3561250	20300	21350
3502869	21229.51	17672.13	3621610	21237.29	19101.7	3391667	19888.89	19444.45	3391667	18777.78	24333.33	3561250	20066.67	21233.33
3561250	21933.33	17850	3502869	20770.49	18475.41	3502869	20540.98	20081.97	3502869	19393.44	25131.15	3621610	20525.42	21237.29
3502869	21459.02	17442.62	3561250	21233.33	18200	3446371	20322.58	19645.16	3502869	19622.95	24786.88	3621610	20762.71	21000
3502869	21459.02	17098.36	3621610	21474.58	18627.12	3446371	20322.58	19645.16	3446371	19193.55	24387.1	3446371	19645.16	19645.16
3446371	21000	16709.68	3502869	20311.47	17557.38	3446371	20322.58	19419.36	3446371	19193.55	24274.19	3561250	20533.33	20416.67
3561250	21933.33	16916.67	3621610	21237.29	17915.25	3446371	20548.39	19306.45	3446371	19193.55	23935.48	3561250	20533.33	20066.67
3561250	21700	16916.67	3561250	20533.33	17850	3446371	20774.19	19080.64	3391667	19000	23555.55	3621610	20762.71	20169.49
3446371	21000	16032.26	3502869	20311.47	17213.12	3446371	20774.19	18854.84	3446371	19419.36	23709.68	3621610	20762.71	20050.85
3502869	21229.51	16295.08	3684052	21482.76	17741.38	3502869	21229.51	18934.43	3391667	19222.22	23111.11	3561250	20533.33	19600
3502869	21000	16180.33	3561250	21000	17033.33	3561250	21700	19133.33	3391667	19000	23000	2887500	18162.16	25445.95
3502869	20770.49	16065.57	3502869	20540.98	16639.34	3502869	21229.51	18704.92	3502869	19622.95	23524.59	3561250	20533.33	19133.33
3621610	21237.29	16254.24	3621610	21237.29	16966.1	3502869	21229.51	18704.92	3446371	19645.16	23145.16	2967708	18277.78	25763.89
3561250	21000	16100	3502869	20540.98	16524.59	3446371	20774.19	18064.52	3391667	19222.22	22333.33	3561250	20533.33	19016.67
3502869	20540.98	15721.31	3502869	20540.98	16180.33	3502869	21229.51	18360.66	3391667	19444.45	22111.11	3561250	20766.67	18666.67
3502869	20311.47	15377.05	3561250	20766.67	16333.33	3502869	21229.51	18245.9	3338672	19031.25	21984.38	3561250	21000	18666.67
3561250	20533.33	15516.67	3684052	21482.76	16775.86	3446371	21000	17838.71	3446371	19419.36	22241.94	3621610	21237.29	18745.76
3561250	20300	15283.33	3621610	21237.29	16254.24	3561250	21466.67	18200	3502869	19852.46	22606.56	3561250	21000	18200
3561250	20533.33	15283.33	3621610	21000	16135.59	3446371	20774.19	17612.9	3446371	19645.16	22241.94	3621610	21237.29	18389.83
3502869	20081.97	14803.28	3561250	21000	15400	3502869	21000	17786.88	3502869	20081.97	22262.29	3621610	21237.29	18389.83
3621610	21000	14949.15	3502869	20770.49	15147.54	3446371	20548.39	17274.19	3502869	20311.47	22032.79	3446371	20096.77	17387.1
3502869	20311.47	14344.26	3684052	21724.14	15689.66	3502869	20770.49	17557.38	3391667	19444.45	21222.22	3561250	20766.67	17850
3502869	20311.47	14114.75	3561250	21000	15050	3502869	20770.49	17557.38	3391667	19444.45	21000	3621610	21237.29	17915.25
3502869	20311.47	13885.25	3561250	21000	15050	3502869	20770.49	17327.87	3446371	19870.97	21225.81	3561250	20766.67	17383.33
3502869	20540.98	13770.49	3561250	21000	14816.67	3502869	20770.49	17327.87	3391667	19666.67	21000	3561250	20766.67	17150
3621610	21237.29	13762.71	3561250	21000	14350	3561250	21000	17383.33	3391667	19444.45	20666.67	3561250	20766.67	17150
3561250	21000	13300	3621610	21237.29	14711.86	3446371	20322.58	16709.68	3391667	19666.67	20444.45	3502869	20311.47	16868.85
3446371	20322.58	12983.87	3561250	21233.33	14233.33	3502869	20770.49	16868.85	3502869	20081.97	21000	3502869	20311.47	16524.59
3561250	20766.67	13183.33	3621610	21474.58	14237.29	3561250	21233.33	17033.33	3502869	20081.97	20655.74	3561250	20766.67	16566.67
3446371	20096.77	12645.16	3561250	21000	14116.67	3446371	20774.19	16370.97	3446371	19870.97	20322.58	3561250	20766.67	16450
3502869	20311.47	12508.2	3561250	21000	13883.33	3502869	21000	16639.34	3446371	20096.77	19983.87	3561250	20766.67	16450
3502869	20540.98	12393.44	3561250	21000	13650	3446371	20548.39	16145.16	3446371	20096.77	19758.06	3502869	20311.47	15950.82
3446371	19870.97	11854.84	3621610	21237.29	13762.71	3446371	20774.19	15919.35	3502869	20311.47	20081.97	3446371	19645.16	15467.74
1424500	12506.67	7653.333	3502869	20540.98	13196.72	3446371	20774.19	15806.45	3502869	20311.47	19852.46	3561250	20533.33	15866.67
3446371	20096.77	11516.13	3748684	22228.07	14000	3502869	21229.51	15836.07	1415066	12377.48	10615.89	3561250	20533.33	15750
3561250	20766.67	12016.67	3561250	20533.33	13300	3391667	20555.55	15333.33	3391667	19888.89	18888.89	3561250	20300	15633.33
3561250	20766.67	11783.33	3621610	20762.71	13525.42	3502869	21229.51	15950.82	3446371	20096.77	19080.64	3502869	19852.46	15262.29
3502869	20540.98	11245.9	3561250	20533.33	13300	3446371	20774.19	15467.74	3502869	20311.47	19278.69	3561250	20300	15516.67
3502869	20770.49	11016.39	3502869	20081.97	12852.46	3502869	21229.51	15606.56	3391667	19666.67	18555.55	3502869	19852.46	14803.28
3502869	20770.49	11016.39	3502869	20081.97	12622.95	3391667	20555.55	15111.11	3391667	19888.89	18333.33	3502869	19852.46	14688.52
3446371	20322.58	10838.71	3561250	20533.33	12716.67	3502869	21229.51	15377.05	3446371	20322.58	18629.03	3561250	20066.67	14700
3502869	20770.49	11360.66	3621610	21000	12813.56	3446371	20774.19	15129.03	3446371	20322.58	18516.13	3621610	20525.42	15067.8
3561250	21233.33	11666.67	3502869	20540.98	12163.93	3338672	20125	14656.25	3446371	20322.58	18290.32	3561250	20066.67	14350
3502869	20770.49	11704.92	3502869	20540.98	12163.93	3502869	21000	15262.29	3391667	20111.11	17888.89	3621610	20525.42	14593.22
3561250	21000	12250	3446371	20322.58	11854.84	3338672	20125	14328.13	3561250	20766.67	18783.33	3621610	20525.42	14474.58
3502869	20770.49	12393.44	3561250	20766.67	12366.67	3446371	20774.19	14451.61	3391667	19888.89	17666.67	3621610	20525.42	14237.29
3446371	20548.39	12532.26	1463527	12561.64	8102.74	3446371	20548.39	14451.61	3502869	20770.49	18131.15	3621610	20762.71	14237.29

Human

Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
3561250	21466.67	13183.33	3502869	20540.98	12737.71	3446371	20774.19	14225.81	3287308	19707.69	16800	3446371	19645.16	13322.58
3502869	21000	13196.72	3561250	20766.67	13300	3502869	21000	14344.26	3391667	20111.11	17444.45	3502869	20081.97	13540.98
3621610	21711.87	13762.71	3502869	20311.47	13311.48	3561250	21233.33	14700	3338672	19906.25	17062.5	3561250	20300	13766.67
3621610	21711.87	14118.64	3502869	20540.98	13540.98	3502869	21000	14114.75	3502869	20770.49	17672.13	3502869	20081.97	13196.72
3446371	20548.39	13887.1	3502869	20540.98	13426.23	3391667	20333.33	13555.56	3338672	19906.25	16625	3621610	20762.71	13525.42
3502869	21000	14573.77	3502869	20540.98	13770.49	3446371	20548.39	13661.29	3446371	20322.58	17274.19	3446371	19870.97	12983.87
3561250	21466.67	14933.33	3561250	20766.67	14000	3502869	21000	13770.49	3391667	20333.33	16666.67	3561250	20300	13300
3446371	20548.39	14564.52	8547000	40040	56000	3502869	21000	13655.74	3391667	20333.33	16444.45	3621610	20525.42	13288.14
3446371	20322.58	14790.32	3502869	20540.98	14229.51	3446371	20548.39	13322.58	3446371	20774.19	16822.58	3621610	20762.71	13288.14
3502869	20770.49	15262.29	3502869	20081.97	14688.52	3502869	20770.49	13540.98	3446371	20548.39	16596.77	3621610	20762.71	13169.49
3502869	20540.98	15491.8	3561250	20533.33	15166.67	3561250	21233.33	13766.67	3502869	21000	16754.1	3561250	20533.33	12833.33
3446371	20096.77	15693.55	3621610	21000	15661.02	3502869	20770.49	13196.72	3502869	21000	16639.34	3621610	21000	13050.85
3446371	20096.77	15806.45	3561250	20533.33	15283.33	3561250	21466.67	13416.67	3446371	21000	16258.06	3621610	21000	13050.85
3502869	20311.47	16295.08	3561250	20533.33	15633.33	3502869	21000	13081.97	3391667	20333.33	15666.67	3561250	20766.67	12600
3391667	19666.67	16000	3502869	20081.97	15721.31	3502869	21000	13081.97	3446371	20548.39	15806.45	3561250	20766.67	12483.33
3446371	20096.77	16596.77	3502869	20081.97	15721.31	3561250	21466.67	13183.33	3391667	20111.11	15555.56	3502869	20311.47	12278.69
3502869	20311.47	17213.12	3621610	21000	16610.17	3446371	20548.39	12532.26	3446371	20548.39	15693.55	3502869	20311.47	12278.69
3446371	19870.97	17048.39	3621610	20762.71	16728.81	3446371	20774.19	12193.55	3338672	19906.25	14984.38	3502869	20311.47	12508.2
3502869	20081.97	17557.38	3561250	20533.33	16566.67	3338672	20125	11812.5	3391667	20111.11	15333.33	3561250	20533.33	12833.33
3391667	19222.22	17111.11	3621610	20762.71	16966.1	3446371	20548.39	12306.45	3502869	20770.49	15606.56	3502869	20081.97	12967.21
3446371	19645.16	17612.9	3446371	19645.16	16483.87	3391667	20333.33	12333.33	3502869	20540.98	15721.31	3561250	20533.33	13416.67
3446371	19645.16	17725.81	3391667	19444.45	16111.11	3446371	20774.19	12983.87	3446371	20096.77	15016.13	3446371	19870.97	13209.68
3446371	19645.16	17725.81	3502869	20081.97	17098.36	3561250	21466.67	13416.67	3446371	20322.58	15016.13	3502869	20081.97	13426.23
3446371	19645.16	17951.61	3502869	19852.46	17098.36	3502869	21000	13311.48	3391667	20111.11	14666.67	3502869	20081.97	13655.74
3502869	19852.46	18360.66	3446371	19870.97	17048.39	3446371	20548.39	13209.68	3446371	20096.77	14790.32	3502869	19852.46	13885.25
3287308	18630.77	17446.15	3502869	19852.46	17327.87	3446371	20548.39	13435.48	3446371	20548.39	14564.52	3561250	20066.67	14116.67
3391667	19222.22	18111.11	3446371	19645.16	17274.19	3502869	21000	13770.49	3391667	20555.55	14333.33	3684052	21000	14844.83
3446371	19419.36	18516.13	3446371	19645.16	17387.1	3446371	20322.58	13661.29	3446371	20774.19	14338.71	3502869	19622.95	14344.26
3446371	19193.55	18741.94	3502869	19852.46	17901.64	3561250	21000	14466.67	3446371	20774.19	14451.61	3561250	20066.67	14583.33
3446371	19193.55	18967.74	3502869	20081.97	17901.64	3502869	20770.49	14229.51	3446371	20774.19	14338.71	3561250	20066.67	14583.33
3391667	19000	18777.78	3446371	19419.36	17725.81	3391667	20111.11	14111.11	3502869	21229.51	14344.26	3446371	19193.55	14338.71
3446371	19193.55	19532.26	3446371	19645.16	17951.61	3391667	20111.11	14222.22	3391667	20555.55	13888.89	3502869	19622.95	14573.77
3446371	19193.55	19532.26	3561250	20300	18783.33	3446371	20322.58	14564.52	3391667	20777.78	13666.67	3446371	19193.55	14564.52
3391667	19222.22	19222.22	3502869	19852.46	18590.16	3391667	19888.89	14777.78	3391667	20777.78	13444.44	3502869	19622.95	14803.28
3502869	19852.46	19967.21	3502869	20081.97	18819.67	3391667	19888.89	14888.89	3502869	21459.02	13655.74	3502869	19622.95	14918.03
3502869	19852.46	20196.72	3502869	20081.97	18934.43	3446371	20096.77	15016.13	3338672	20562.5	13015.63	3446371	19193.55	14790.32
3502869	19852.46	20426.23	3391667	19444.45	18444.45	3338672	19468.75	14765.63	3502869	21688.53	13426.23	3446371	19193.55	15016.13
3446371	19645.16	20322.58	3502869	19852.46	19393.44	3391667	19888.89	15111.11	3446371	21225.81	13209.68	3446371	19193.55	15016.13
3391667	19222.22	20333.33	3446371	19419.36	19193.55	3338672	19468.75	14875	3446371	21225.81	12983.87	3446371	19193.55	15129.03
3391667	19444.45	20444.45	3502869	19622.95	19622.95	3391667	19888.89	15333.33	3446371	21451.61	12983.87	3391667	19000	15000
3338672	19031.25	20453.13	3502869	19852.46	19737.71	3446371	20096.77	15693.55	3446371	21451.61	12758.06	3446371	19419.36	15354.84
3446371	19645.16	21451.61	3502869	19622.95	20081.97	3446371	20096.77	15806.45	3391667	21000	12666.67	3446371	19193.55	15241.94
3338672	19031.25	20781.25	3502869	19622.95	20311.47	3446371	20096.77	15919.35	3391667	20777.78	12444.44	3502869	19622.95	15836.07
3446371	19645.16	21790.32	3391667	19222.22	19555.55	3338672	19250	15640.63	3338672	20562.5	12140.63	3446371	19419.36	15467.74
3446371	19645.16	22129.03	3502869	19622.95	20540.98	3391667	19666.67	15888.89	3338672	20781.25	12250	3446371	19419.36	15693.55
3446371	19645.16	22129.03	3502869	19622.95	20655.74	3391667	19888.89	16000	3287308	20353.85	11953.85	3502869	19622.95	16065.57
3391667	19222.22	21777.78	3446371	19419.36	20548.39	3446371	20096.77	16370.97	3502869	21688.53	12393.44	3502869	19622.95	15950.82
3446371	19419.36	22354.84	3446371	19193.55	20548.39	3446371	19870.97	16596.77	3391667	20777.78	12111.11	3502869	19622.95	16180.33
3391667	19000	22555.55	3446371	19193.55	21000	3391667	19888.89	16555.55	3391667	21000	11888.89	3502869	19622.95	16409.84
3338672	18812.5	22312.5	3446371	19193.55	20887.1	3338672	19468.75	16625	3391667	21000	11888.89	3502869	19622.95	16409.84
3338672	18593.75	22421.88	3446371	19193.55	20887.1	3391667	19666.67	16777.78	3338672	20562.5	12031.25	3446371	19193.55	16145.16
3391667	19000	23222.22	3391667	18777.78	20888.89	3391667	19666.67	17000	3338672	20562.5	12140.63	3446371	19193.55	16145.16
3391667	19000	23000	3391667	19000	20777.78	3446371	19870.97	17387.1	3287308	20138.46	12169.23	3502869	19622.95	16524.59
3446371	19193.55	23822.58	3446371	19193.55	21338.71	3502869	20081.97	17901.64	3338672	20343.75	12687.5	3502869	19622.95	16754.1
3446371	19193.55	24048.39	3391667	18777.78	21333.33	3446371	19870.97	17612.9	3391667	20555.55	13111.11	3502869	19622.95	16754.1
3391667	18777.78	23777.78	3391667	19000	21444.45	3338672	19250	17281.25	3338672	20125	13234.38	3502869	19622.95	16868.85
3391667	18777.78	24000	3391667	19000	21777.78	3502869	20311.47	18360.66	3338672	20125	13343.75	3446371	18967.74	16709.68
3502869	19163.93	25016.39	3391667	18777.78	21777.78	3338672	19250	17609.38	3391667	20111.11	13777.78	3561250	19600	17266.67
1765909	13016.53	16487.6	3338672	18812.5	21546.88	3391667	19444.45	17888.89	3391667	20111.11	14000	3502869	19163.93	17098.36
3446371	18741.94	25403.23	3446371	19193.55	22693.55	3446371	19870.97	18403.23	3446371	20096.77	14564.52	3391667	18555.55	16444.45
3338672	18156.25	24390.63	3446371	19193.55	22580.64	3391667	19666.67	18333.33	3446371	20548.39	14338.71	3561250	19600	17616.67
3446371	18741.94	25403.23	3502869	19393.44	23295.08	3338672	19250	18156.25	3391667	19888.89	14555.56	3561250	19600	17850
			3621610	19813.56	24084.75	3502869	20311.47	19278.69	3391667	19888.89	14444.44	3502869	19163.93	17672.13
			3561250	19366.67	23800	3446371	19870.97	19193.55	3391667	20111.11	14666.67	3446371	18741.94	17274.19
			3561250	19600	23916.67	3391667	19444.45	18888.89	3446371	20322.58	15016.13	3502869	19163.93	17786.88
			3502869	19393.44	23524.59	3446371	19870.97	19193.55	3391667	19888.89	14666.67	3502869	18934.43	17786.88
			3502869	19163.93	23983.61	3391667	19444.45	19111.11	3391667	19888.89	14777.78	3446371	18741.94	17612.9
			3502869	19163.93	24442.62	3561250	20533.33	20183.33	3391667	19666.67	15111.11	2704747	16215.19	23126.58
			3502869	19163.93	24557.38	3446371	19870.97	19645.16	3391667	19666.67	15111.11	3446371	18516.13	17838.71
			3621610	19813.56	25508.47	3391667	19222.22	19444.45	3338672	19468.75	14984.38	3502869	18934.43	18360.66
			3502869	19163.93	24901.64	3338672	19031.25	19359.38	3391667	19888.89	15222.22	3446371	18516.13	18177.42
			3502869	19622.95	25131.15	3446371	19645.16	20322.58	3338672	19468.75	15093.75	3391667	18111.11	17888.89
			3502869	19622.95	25475.41	3446371	19419.36	20322.58	3391667	19888.89	15333.33	3502869	18704.92	18475.41
			3502869	19393.44	25934.43	3391667	19222.22	20222.22	3391667	19666.67	15444.44	3446371	18516.13	18403.23
			3446371	18741.94	25854.84	3446371	19193.55	20548.39	3446371	20096.77	15806.45	3502869	18475.41	18704.92
			3338672	18375	25484.38	3338672	18812.5	20125	3391667	19888.89	15555.56	3502869	18475.41	18934.43
			3621610	19813.56	27525.42	3391667	19000	20777.78	3287308	19061.54	15184.62	3502869	18475.41	19049.18
			3391667	18555.55	26111.11	3338672	18593.75	20453.13	3338672	19468.75	15531.25	3446371	18064.52	18854.84
			3338672	18156.25	26031.25	3237500	18242.42	19833.33	3338672	19687.5	15750	3561250	18666.67	19483.33
						3338672	18593.75	20671.88	3446371	20096.77	16483.87	3446371	17838.71	19193.55
						3338672	18593.75	20781.25	3338672	19687.5	15859.38	3446371	17838.71	19080.64
						3338672	18812.5	21000	3287308	19061.54	15830.77	3391667	17666.67	19000
						3391667	18555.55	21555.55	3287308	19061.54	15830.77	3391667	17666.67	18888.89
						3391667	18777.78	21555.55	3287308	19061.54	16046.15	3446371	17838.71	19532.26
						3391667	18333.33	21777.78	3287308	19061.54	16153.85	3502869	18016.39	19737.71
						3338672	18375	21546.88	3338672	19468.75	16406.25	3502869	18016.39	19967.21
						3338672	18375	21765.63	3391667	19444.45	17000	3446371	17612.9	19870.97
						3287308	18200	21538.46	3237500	18454.54	16333.33	3446371	17387.1	19870.97
						3338672	18375	22093.75	3287308	18846.15	16800	3391667	17222.22	19777.78

Human

Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
						3446371	18967.74	23032.26	3391667	19444.45	17444.45	3446371	17387.1	20209.68
						3287308	18200	21969.23	3287308	18846.15	16907.69	3391667	17444.45	20000
						3338672	18375	22640.63	3338672	18812.5	17500	3446371	17387.1	20435.48
						3391667	18555.55	23000	3287308	18630.77	17338.46	3446371	17387.1	20887.1
						3338672	18375	22968.75	3338672	19031.25	17718.75	3502869	17786.88	21229.51
						3391667	18555.55	23333.33	3391667	19222.22	18111.11	1415066	11357.62	9456.954
						3446371	18967.74	23709.68	3502869	20081.97	18704.92	3446371	17612.9	21000
						3446371	19193.55	23822.58	3391667	19222.22	18444.45	3446371	17838.71	21225.81
						3338672	18593.75	23296.88	3287308	18630.77	18092.31	3338672	17281.25	20562.5
						3391667	18777.78	23666.67	3446371	19645.16	19080.64	3446371	17612.9	21338.71
						3391667	18777.78	23888.89	3391667	19222.22	18777.78	3391667	17444.45	21111.11
						3391667	18555.55	24222.22	3391667	19222.22	19222.22	3561250	18200	22283.33
						3391667	18555.55	24333.33	3391667	19444.45	19111.11	3502869	17786.88	21918.03
						3502869	19622.95	25245.9	3391667	19222.22	19222.22	3446371	17612.9	21564.52
						3391667	18555.55	24888.89	3391667	19222.22	19333.33	3446371	17387.1	21903.23
						3446371	18741.94	25064.52	3338672	19031.25	19250	3446371	17612.9	22129.03
						3502869	19393.44	25704.92	3287308	18630.77	19276.92	3446371	17387.1	22241.94
						3446371	18741.94	25629.03	3338672	18812.5	19578.13	3446371	17387.1	22241.94
						3502869	19163.93	26049.18	3391667	19222.22	20000	3446371	17387.1	22467.74
						3446371	18967.74	25967.74	3338672	18812.5	20015.63	3391667	17222.22	22222.22
						3391667	18333.33	25666.67	3391667	19222.22	20222.22	3446371	17387.1	22580.64
						3446371	18741.94	26306.45	3391667	19444.45	20333.33	3391667	17222.22	22222.22
						3561250	19366.67	27533.33	3287308	18846.15	19815.38	3391667	17000	22444.45
						3391667	18111.11	26444.45	3391667	19222.22	20777.78	3502869	17557.38	23295.08
						3446371	18516.13	26870.97	3287308	18630.77	20138.46	3502869	17327.87	23639.34
						3391667	18333.33	26444.45	3287308	18630.77	20353.85	3502869	17327.87	23524.59
						3446371	18290.32	26983.87	3391667	19222.22	21111.11	3446371	16935.48	23483.87
						3391667	18111.11	26777.78	3338672	18812.5	20890.63	3502869	17327.87	23983.61
						3287308	17769.23	25846.15	3391667	19444.45	21444.45	3561250	17733.33	24616.67
						3287308	17769.23	26061.54	3391667	19000	21444.45	3446371	17161.29	23596.77
						3287308	17769.23	26384.62	3338672	18812.5	21328.13	3391667	16777.78	23555.55
						3287308	17553.85	26492.31	3502869	19393.44	22606.56	3446371	16935.48	24048.39
						3287308	17553.85	26923.08	3287308	18200	21215.38	3502869	17327.87	24442.62
						3338672	17718.75	27453.13	3338672	18593.75	21656.25	3502869	17327.87	24557.38
						3391667	17666.67	29666.67	3338672	18375	21984.38	3446371	16935.48	24387.1
									3287308	18200	21753.85	3502869	17327.87	24672.13
									3338672	18593.75	22093.75	3446371	17387.1	24387.1
									3287308	18415.38	21969.23	3502869	17557.38	25016.39
									3287308	18415.38	22184.62	3391667	17222.22	24555.55
									3338672	18593.75	22531.25	3502869	17557.38	25245.9
									3391667	18777.78	23222.22	3446371	17387.1	25177.42
									3287308	17984.62	22938.46	3391667	17000	24888.89
									3391667	18555.55	23666.67	3391667	17222.22	24777.78
									3237500	17606.06	23121.21	3446371	17387.1	25290.32
									3391667	18777.78	24444.45	3391667	17000	25222.22
									3391667	18777.78	24555.55	3446371	17612.9	25629.03
									3287308	18630.77	23800	3446371	17161.29	25854.84
									3338672	18812.5	24390.63	3502869	17557.38	26278.69
									3391667	19222.22	25111.11	3287308	16692.31	24876.92
									3446371	19645.16	25854.84	3446371	17612.9	26193.55
									3287308	18630.77	24661.54	3391667	17000	25888.89
									3338672	19250	25265.63	2574398	14759.04	26566.27
									3287308	18846.15	25092.31	3391667	17222.22	26111.11
									3391667	19222.22	26777.78	3446371	17387.1	26645.16
									3287308	18415.38	26276.92	3502869	17786.88	27311.47
												3391667	17000	26444.45
												2348077	14230.77	23461.54
												2348077	14230.77	23615.38
												3446371	17612.9	27322.58
												3502869	17557.38	28114.75
												2348077	14230.77	23769.23
												3502869	17786.88	28344.26
												3502869	17557.38	28688.53
												3391667	17000	27888.89
												3391667	17222.22	27555.55
												2670938	15050	29400
												2670938	15050	29225
												2670938	15050	29400
												3502869	17557.38	29377.05
												3502869	18016.39	29836.07

ISAC

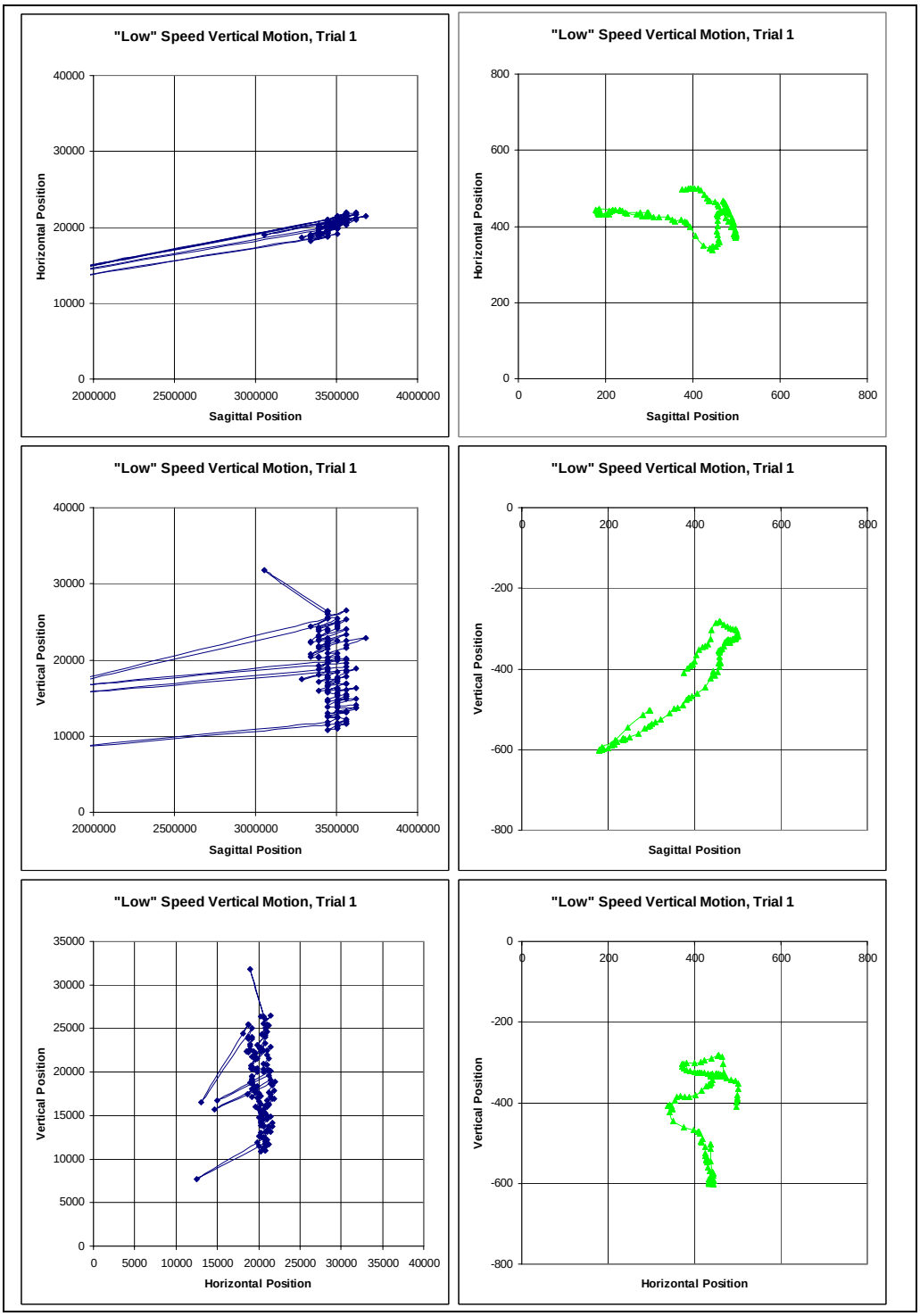
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
296.7681	436.7661	-502.5142	292.9969	443.0739	-501.0898	293.8098	442.2637	-509.0154	295.3158	440.4072	-510.2218	294.2662	441.5132	-507.17
296.9477	436.7604	-502.2404	293.0867	443.0158	-500.8278	293.9489	442.1474	-508.924	295.4152	440.348	-510.062	294.3018	441.4555	-507.0675
295.2143	436.8149	-503.5282	293.2252	442.8999	-500.7706	292.3068	442.169	-509.8613	292.0161	440.7534	-511.6467	291.7038	441.3257	-508.3548
279.9105	437.2424	-513.7057	292.2662	442.9671	-501.5589	284.5063	442.7151	-516.2864	283.9227	441.8926	-517.9661	270.6548	440.9297	-521.8125
244.9486	436.5704	-546.1045	282.0627	444.396	-513.0517	266.1608	443.0834	-528.8193	251.0528	441.2023	-539.0246	236.1292	441.1247	-552.8985
216.0285	442.7124	-575.233	258.8054	442.5597	-547.1453	225.3202	442.3953	-562.4192	208.7298	442.5415	-582.4192	193.8508	446.3431	-588.1716
184.8871	445.5646	-593.4872	201.8407	441.3344	-602.1155	194.9073	450.2208	-588.7241	149.8727	448.9536	-635.3192	163.6068	447.8124	-614.4994
178.4879	444.2908	-603.3918	145.6571	446.6603	-636.5015	178.3043	456.0853	-595.6414	113.1621	451.2017	-651.6206	147.2235	451.9735	-624.5706
183.6523	442.7125	-600.5598	126.1174	449.8829	-639.6025	182.8496	454.3978	-593.0857	112.1196	452.2354	-644.9601	143.9997	454.5393	-619.0053
182.5005	441.5771	-597.4968	144.3198	446.797	-631.0367	186.7572	452.4282	-588.7599	125.1087	452.1267	-632.7949	131.0821	454.3658	-619.62
180.084	440.0259	-601.068	159.8303	443.9131	-611.9501	182.437	452.6175	-587.5131	132.0275	451.0158	-621.8639	119.8809	453.9696	-624.6005
180.9184	437.3811	-601.5363	167.3269	445.5732	-588.3411	176.7482	453.8685	-590.0376	122.184	451.6266	-613.3468	110.0155	453.9783	-624.5308
183.788	436.0794	-599.6231	164.7281	447.5953	-589.6032	168.1567	454.1931	-594.6563	108.2073	451.7008	-620.3851	104.451	454.1602	-626.0437
183.9595	434.173	-599.83	158.3574	447.9125	-600.4339	169.2983	454.3177	-594.0334	96.096	453.3319	-625.182	99.64467	454.81	-626.2227
183.7774	432.5448	-600.6012	158.6578	447.8212	-600.8496	170.8078	453.7872	-593.6161	90.95564	454.3125	-625.5126	94.02496	454.701	-627.2672
184.7287	432.0373	-600.783	150.316	447.2772	-603.9442	171.2504	453.8069	-593.5009	85.44021	454.6453	-628.2182	87.60617	455.4549	-628.1376
185.7594	432.5283	-600.1762	142.4684	447.95	-616.4168	173.8323	455.2583	-591.6702	79.11595	454.0285	-635.3006	82.2356	456.2607	-628.7163
189.7886	432.9156	-598.6396	136.2688	449.4079	-618.9638	171.333	455.7387	-593.2643	72.32449	453.4283	-637.6973	78.65105	457.2457	-628.6478
199.1462	432.4643	-595.3297	140.6082	448.2068	-618.3114	173.4158	455.9624	-594.1012	69.78703	453.5822	-638.2259	75.57012	456.9648	-629.1475
207.3013	432.6159	-590.5128	145.8935	447.6175	-615.7534	175.2013	457.4635	-593.0689	66.90483	453.9082	-639.3756	74.01067	456.7704	-629.5315
207.7148	437.5555	-587.5119	154.7959	444.4888	-612.4004	177.0847	457.957	-591.9576	95.77899	464.995	-651.1943	74.1738	456.8303	-629.1221
209.5006	438.4737	-586.3529	164.0263	443.4146	-609.5684	179.3982	458.1077	-591.4041	149.9538	475.4787	-647.4515	74.47155	456.8675	-628.6245
207.6614	440.5607	-588.2849	175.2154	441.5553	-604.8853	181.5907	458.4426	-590.7658	192.5266	472.6706	-617.3949	75.55724	457.0032	-627.6648
213.9698	440.7218	-586.7578	183.8656	440.4215	-601.6084	184.7839	459.2407	-589.156	191.828	474.6781	-599.3132	81.48353	458.2772	-626.2619
221.4639	442.9727	-581.4713	186.9325	439.5464	-600.1518	189.8925	459.7545	-586.7823	173.2623	476.7428	-598.8418	88.74925	459.0618	-622.1804
233.2064	443.1284	-574.0502	192.3772	437.2521	-599.7647	193.1626	460.4334	-584.8047	141.7718	465.3812	-612.7739	94.00731	459.6665	-619.9429
234.4192	442.1578	-571.9254	202.3047	436.0687	-595.6762	195.1445	461.9882	-583.1476	111.466	461.5198	-629.1637	98.36423	460.6706	-618.7244
234.7232	440.9275	-575.5172	217.1803	435.769	-588.555	196.7305	462.1972	-582.8066	96.56063	461.9709	-630.2839	101.7766	461.1197	-617.8572
239.0179	441.0184	-573.1415	230.8345	431.3898	-581.1196	202.3202	462.8985	-580.5342	94.55836	460.8242	-630.3258	107.2519	461.8404	-616.1794
249.3499	434.2192	-570.0896	237.0801	429.8356	-577.9913	210.6794	464.1087	-576.0391	98.77964	460.4513	-627.6645	116.9129	462.6841	-613.8025
271.1127	430.553	-561.4655	239.2757	425.7043	-579.7469	219.6349	465.1766	-570.6058	103.0773	460.5204	-623.1002	126.3729	463.9008	-610.2918
284.4242	427.8995	-547.8592	248.9298	420.1672	-578.9434	225.5413	464.6208	-567.517	99.86926	460.9695	-621.7082	135.1816	462.747	-607.9563
292.0916	426.2451	-542.8884	268.4131	416.0951	-567.2122	227.4669	464.1495	-566.4565	96.32653	460.9677	-622.5902	140.466	461.8985	-607.2627
296.3442	427.1791	-540.9316	282.5387	409.464	-557.5596	231.3328	463.523	-566.7308	95.16759	460.8098	-623.0274	149.1006	463.3834	-604.3092
302.0604	428.5794	-536.216	292.0754	407.2674	-551.6003	239.2088	460.1185	-563.0384	95.09921	460.8005	-623.0215	154.9509	463.914	-601.5264
309.7277	424.5562	-531.9293	298.8567	409.9749	-545.4639	251.3144	455.7807	-560.2065	95.4364	460.8464	-622.9245	162.3602	463.6227	-598.7337
322.2764	425.3122	-526.1648	300.8695	408.9512	-544.4848	262.395	455.5153	-552.3156	95.22581	460.7444	-623.0215	174.2261	462.8005	-594.5073
342.9249	424.5453	-509.3013	308.399	408.944	-541.7254	273.8897	456.0925	-543.1443	97.55684	459.7048	-622.9913	179.1702	463.0971	-592.2922
353.545	417.1072	-497.9585	314.3059	407.223	-536.623	277.94	455.9593	-538.7769	97.90859	459.5532	-622.9611	189.1691	464.0053	-587.6934
360.0809	413.7112	-496.8411	322.3242	405.5992	-532.0567	279.9934	455.524	-538.508	102.0391	460.0865	-621.5368	196.6064	463.6626	-583.1041
373.8565	417.6006	-490.6674	325.8884	404.1751	-529.1019	284.7992	454.3689	-535.5802	103.5963	460.2784	-619.3178	206.1428	461.4991	-579.4788
381.1365	412.0613	-476.4422	332.0302	406.4755	-525.3035	289.8328	453.9343	-532.1529	106.2716	460.6603	-618.1875	211.8654	460.9333	-576.7107
385.5549	409.4043	-472.2571	338.9733	405.3663	-517.6099	298.2368	454.1025	-525.1122	111.187	461.5391	-616.3384	220.3263	460.7733	-572.8554
388.1758	408.1255	-471.3111	348.578	406.6303	-510.204	305.0791	454.2868	-519.0143	113.7689	462.3354	-614.234	232.5439	458.9657	-566.7351
394.2525	397.8248	-466.6927	358.6713	408.4624	-497.717	310.9098	454.4438	-513.3747	115.2008	463.1694	-613.2373	245.3621	457.902	-559.4582
406.5025	375.6214	-460.3862	361.5666	407.812	-492.2525	313.1441	454.1909	-511.8232	117.7394	463.5086	-612.3899	256.0813	455.2894	-553.4135
424.8006	350.2605	-445.6309	365.9526	407.7453	-490.6243	316.5581	454.4693	-509.3254	121.8321	464.0554	-610.6548	262.9017	453.7319	-550.2163
437.756	342.4561	-424.3607	368.1759	408.0574	-488.0612	320.1366	454.8202	-505.5758	124.9667	464.4743	-609.3048	276.3948	455.1247	-542.5192
444.1127	337.9719	-406.6743	373.5652	408.116	-482.3426	325.5781	454.6435	-502.0699	131.1258	465.3287	-606.9448	285.4086	454.3293	-534.7477
442.8671	342.2452	-406.0945	380.3649	405.7651	-474.0335	331.8704	454.1468	-496.8643	136.9789	466.0465	-605.1599	294.1511	453.2276	-528.8482
444.5493	347.6985	-413.8102	389.0164	404.9692	-465.8694	337.7837	452.2065	-492.4837	143.4948	466.9158	-601.9772	299.3895	447.2371	-528.3935
447.9536	348.1555	-415.6738	395.1557	404.6745	-455.0625	345.3123	450.1534	-487.044	150.3578	467.8314	-599.4337	308.2182	444.066	-526.9792
453.6267	346.0891	-408.3976	398.6344	404.3006	-450.3262	350.0572	445.3126	-483.4538	157.621	468.8004	-596.272	314.4324	441.6939	-519.5544
457.562	354.0582	-394.0682	403.6142	406.1987	-445.8582	357.7149	431.5395	-484.6791	163.9698	469.6474	-593.1729	313.0359	442.2012	-519.9103
460.2428	359.0848	-385.6237	406.0056	407.8733	-440.4027	371.7863	407.2563	-482.6308	166.1962	469.9445	-592.0121	304.3131	443.6739	-526.3905
459.0026	366.5412	-384.1353	410.6321	407.1178	-436.6156	388.6773	384.3033	-475.7072	171.3712	471.3883	-591.4049	285.9729	443.7585	-541.8466
457.9775	376.5963	-385.5062	414.7038	406.8798	-430.4153	402.2125	373.0536	-463.3994	177.5367	473.1383	-588.891	280.2036	449.9843	-552.2393
456.0175	388.0512	-385.1683	422.4567	408.1317	-421.1085	407.0826	369.4937	-457.6821	189.5621	474.7978	-582.9485	282.0026	450.4018	-547.1884
457.8226	401.8459	-380.5572	426.9182	406.6707	-412.8315	407.9022	368.3524	-457.2879	199.6456	475.3587	-576.6176	299.2793	449.246	-537.9365
457.4577	415.5781	-370.2249	431.927	406.4447	-407.4143	405.3787	376.4253	-457.5981	202.0124	475.4092	-574.8091	339.6391	447.6708	-505.0571
457.6901	426.9206	-359.7448	434.0826	411.1096	-401.878	405.1374	377.594	-457.3089	201.352	475.6393	-575.1906	363.5461	438.9652	-473.8106
456.0956	432.4635	-355.4932	436.4526	411.331	-399.3548	413.643	375.5557	-451.3501	204.2605	474.9765	-574.0522	379.9239	410.5249	-467.3949
456.8548	434.482	-356.5453	436.9	411.5445	-398.9846	418.0183	381.5412	-438.821	210.5136	474.1105	-571.122	382.8312	399.5521	-478.6148
457.8402	434.9913	-354.4842	443.5618	408.5595	-398.0583	422.4812	386.4256	-427.5217	219.2115	473.9426	-566.2882	390.5123	395.9645	-475.3031
460.5192	436.0918	-353.0222	446.5512	407.3675	-393.7513	422.0157	388.4669	-426.5753	229.5815	475.2462	-559.1562	395.7812	399.1128	-458.3784
463.8114	440.183	-349.7249	455.3031	406.3351	-384.5913	418.9348	396.4402	-429.379	239.8731	476.1238	-551.5678	400.2264	403.5272	-449.2403
468.9446	441.0354	-342.6611	460.3718	400.8078	-374.0101	416.9049	402.3366	-429.1446	244.589	476.1003	-548.3786	406.8016	399.2924	-443.9576
472.3845	441.4611	-335.5412	465.7893	393.9926	-366.5404	416.9543	403.2751	-428.7188	247.5739	477.5322	-545.4568	413.8333	398.5387	-435.6919
473.9076	440.612	-332.2769	468.6203	391.7168	-361.1669	420.5773	406.3125	-425.8982	250.4288	478.0979	-543.1115	417.5591	406.3671	-423.4244
476.3729	438.7221	-330.9522	470.2221	392.3275	-356.962	426.4149	410.7647	-415.2751	257.414	478.9517	-538.8824	419.8996	412.4312	-416.2382
478.5429	437.9415	-330.9983	471.6861	394.9268	-352.4208	430.6432	409.6715	-405.9865	261.8458	478.9803	-535.2283	420.6702	417.2576	-412.0414
480.0925	436.6043	-330.9272	473.8823	394.953	-346.7732	430.5057	409.2743	-407.0162	266.9838	478.8461	-531.5567	423.1465	419.6743	-409.0546
480.6838	436.4052	-331.8809	478.8564	391.9522	-342.6381	433.1429	409.621	-410.8619	273.9526	479.7859	-525.5051	428.3494	419.2637	-405.096
481.2941	436.393	-333.2478	481.7027	389.797	-340.8774	434.7584	409.4826	-407.9192	279.9838	480.499	-520.3523	434.8063	418.3551	-403.908
481.5102	437.6094	-334.21	483.365	388.6823	-342.3202	440.4593	410.0839	-399.5684	289.7159	476.2607	-514.8287	438.6785	423.4937	-400.9998
481.0647	440.1542	-334.1921	489.1703	388.1616	-344.9062	445.752	409.3894	-387.8092	295.7361	472.8152	-512.1235	440.689	431.9837	-394.0674
477.6467	450.7023	-333.5183	492.2138	386.7662	-345.9985	449.0147	408.8597	-380.3966	309.249	472.8419	-505.6765	441.9005	436.9244	-385.1478
472.986	461.3505	-333.298	493.3102	387.192	-343.8323	452.2583	408.4139	-375.9722	336.4131	475.4868	-479.7156	441.4992	445.1329	-377.5373
470.0648	466.7076	-334.6792	494.5933	387.6918	-339.1827	454.2162	408.1616	-373.3371	353.7623	471.8573	-449.4789	438.6455	454.4845	-374.0971
469.5734	467.6282	-334.8675	496.0972	387.416	-334.2999	456.6496	407.868	-371.7024	360.5863	467.6277	-441.1209	438.0146	460.2232	-369.9066
471.4535	464.3137	-333.8634	497.3497	384.9063	-332.1163	458.6237	407.263	-368.4364	355.7062	467.0499	-455.5008	436.7878	467.1869	-363.3161
473.9758	460.1548	-330.6202	497.8854	384.7444	-330.4633	462.002	407.1483	-364.594	353.5395	472.5682	-475.6231	438.6494	473.9237	-353.9512
476.0339	456.6204	-328.542	497.689	386.7512	-329.3087	465.0539	406.7095	-358.0151	351.6229	471.9931	-482.9977	441.7258	477.2591	-338.727
477.9878	451.4605	-327.813	496.2591	391.3535	-329.6835	467.744	404.3387	-354.5705	355.5725	468.8549	-479.0365	443.408	477.1958	-331.0699
478.7399	448.7312	-328.2501	492.204	404.6547	-329.9355	469.2046	405.4577	-350.7363	356.2758	468.1721	-476.4825	445.6632	474.8159	-333.1323
482.1445	440.3296	-328.407	490.2921	411.0106	-329.7505	470.5237	405.2518	-349.7763	354.7156	467.4258	-475.9658	448.0412	474.5131	-337.9682
486.213	430.6196	-327.464	489.5742	413.1284	-329.7505	473.2164	407.2712	-347.9705	360.0946	466.1341	-469.4744	449.8085	474.6241	-342.2666

ISAC

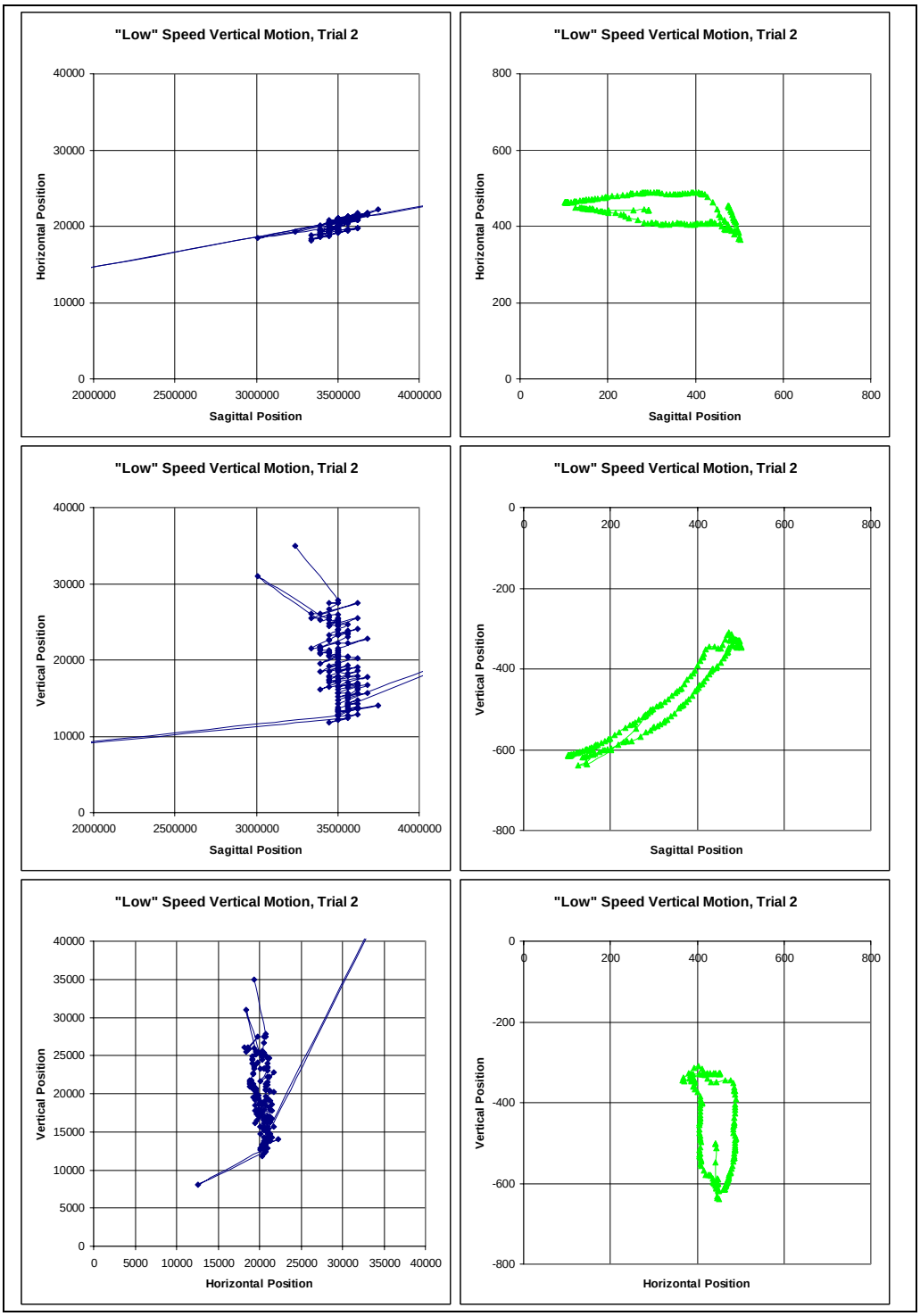
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
489.1944	421.909	-326.2326	488.1989	417.1616	-330.285	475.0115	410.1173	-345.7734	366.0977	467.0595	-458.4841	455.152	468.8921	-346.234
490.8064	416.9112	-326.6407	488.3741	417.2451	-330.4777	473.0664	421.1614	-344.7663	376.9227	464.2385	-442.725	459.7357	465.7958	-343.6662
491.9435	413.3448	-326.7343	487.3414	421.4371	-328.8825	472.7886	425.8284	-342.5338	382.862	460.0553	-433.0894	462.8499	464.26	-336.4046
492.9738	410.2493	-326.5452	486.7454	423.8163	-327.9435	473.5903	426.0685	-342.1516	389.3603	459.2	-426.6092	465.1293	464.4422	-328.8598
494.1378	406.757	-326.2619	484.6892	429.4457	-328.1955	474.732	428.5922	-340.214	390.9359	459.5659	-422.812	467.1274	464.4273	-322.0875
494.8628	399.61	-323.8605	481.4769	437.9859	-327.9769	476.196	425.8658	-338.6479	392.4612	460.3258	-421.2945	467.6012	465.4755	-318.6286
497.6457	389.6847	-321.6963	479.5634	442.5188	-328.2588	476.7447	425.2895	-338.6636	392.6532	459.7822	-422.1846	468.36	464.663	-318.4103
500.0042	380.8764	-318.4096	476.3851	449.8443	-328.7917	480.0467	422.4567	-337.9538	397.4957	460.7468	-420.8234	468.9967	465.3453	-319.4731
499.0779	374.5797	-314.2096	475.2025	452.9311	-328.3537	483.3784	421.1729	-334.5577	402.2	460.2754	-413.4714	469.3461	467.7121	-320.3191
498.268	371.5926	-309.2947	475.5953	452.6547	-327.4139	485.7727	419.5507	-328.2785	407.0308	460.2825	-405.7648	469.972	467.5521	-321.4181
497.758	370.1674	-304.2476	476.1756	451.6279	-326.9438	486.6986	419.781	-323.8169	410.4322	460.0724	-399.7516	471.3498	465.9135	-321.89
496.3739	374.8845	-302.7737	476.4983	451.0672	-326.6617	486.8451	420.0545	-323.0054	412.434	458.6575	-397.7785	473.0695	463.9976	-321.7956
494.1258	381.0084	-302.326	480.5961	441.6643	-326.0973	487.3848	418.9926	-324.1651	415.1232	456.7538	-398.7955	474.7729	461.4081	-321.4483
488.3588	399.3943	-302.1191	484.7152	430.7614	-327.5736	488.595	416.4801	-324.4815	418.6436	451.5289	-399.1789	477.3604	455.7031	-321.5428
482.5945	413.6962	-299.5369	491.704	411.3797	-330.2501	490.4961	411.8422	-323.3558	425.8087	439.5802	-399.8813	481.8353	445.1483	-322.1102
476.4415	423.0949	-294.549	498.3538	383.4573	-339.4068	492.6695	406.4044	-322.2297	438.8381	424.1609	-395.4943	486.218	435.0627	-321.6682
467.8323	438.579	-289.483	501.418	371.0125	-346.1961	493.7762	404.8034	-321.6361	449.6191	414.3005	-385.6118	489.6435	426.5238	-321.0369
459.6953	452.8874	-284.0509	501.9551	366.0006	-344.0125	494.4918	403.7033	-322.202	455.0726	406.962	-375.7595	490.879	422.9624	-321.3525
456.7613	456.6458	-282.426	498.6106	368.0569	-337.4719	495.1932	402.8855	-324.5295	457.0487	405.919	-372.3636	491.4726	421.1318	-321.8258
450.169	464.7558	-286.9286	488.975	379.9484	-326.0333	496.313	402.3124	-326.2367	457.8996	411.0813	-371.4075	490.9688	423.2426	-320.8787
439.6214	466.4386	-303.8607	479.2225	393.6366	-313.5786	495.9475	403.2923	-327.0561	457.2218	417.7854	-367.0803	490.5414	424.9692	-319.8052
437.5692	467.841	-324.9512	473.8959	403.9606	-309.2724	495.634	404.6457	-326.5525	457.1797	425.4413	-362.2882	488.5767	429.691	-320.3741
431.7173	474.4532	-339.3164	469.1195	412.185	-315.8796	493.5778	411.2458	-326.0172	455.7059	429.913	-357.636	486.5861	434.7701	-320.5636
425.6346	483.9903	-343.0829	465.5078	416.3737	-327.8084	491.2441	418.2086	-325.8284	455.2055	433.4623	-357.5243	484.9493	439.0642	-320.3741
417.6537	494.8466	-345.6339	458.9902	422.7043	-340.8442	491.205	418.0217	-326.2061	455.7134	436.0182	-354.5318	482.7572	443.7257	-319.3357
411.2336	500.102	-352.1305	454.5426	431.5399	-349.1163	490.3248	419.9615	-326.8352	457.1829	438.927	-349.3597	480.1561	449.6854	-319.5254
404.3806	500.1721	-366.3355	449.1203	444.5891	-349.5834	491.7715	415.9075	-326.5831	456.3084	448.154	-340.5953	478.1325	454.298	-319.4311
400.2979	499.6421	-380.5656	439.8626	464.0658	-345.238	492.4707	415.3907	-324.3793	457.3898	451.2257	-334.7728	475.3236	459.6271	-318.3951
396.3421	499.1286	-388.2685	427.9842	477.1748	-344.2514	493.4467	413.6227	-322.9299	458.8553	450.9706	-332.4207	473.2676	459.3801	-314.4597
394.4231	499.2994	-390.4278	420.5135	484.0628	-352.4683	493.6306	412.0943	-323.2132	461.2494	450.6323	-333.6597	471.3122	459.888	-309.1765
389.7705	498.6077	-391.3588	416.1567	485.3969	-361.5403	493.1917	413.1661	-323.1815	463.0011	450.4671	-337.1321	469.1595	461.1336	-302.875
383.0854	497.5711	-397.9947	414.2028	484.3591	-369.7546	491.9942	415.0425	-321.8272	466.9046	447.5607	-342.327	467.9169	461.0938	-300.2804
375.8931	496.9557	-410.6786	414.3239	484.5414	-371.4199	490.1475	414.9718	-318.4709	471.1951	441.9337	-343.8653	467.5843	461.8293	-300.1567
364.222	495.8948	-430.5127	411.3144	486.5686	-372.3276	490.2443	414.4947	-314.8739	475.127	434.8508	-342.6991	465.6497	466.5149	-300.242
351.9975	496.0966	-449.4543	406.4295	488.296	-380.3953	486.928	421.8339	-312.5085	479.528	430.2742	-337.8637	460.2447	477.8943	-299.5889
341.0313	496.9586	-464.1668	400.7363	488.4705	-391.6223	484.9776	425.7726	-311.8288	482.3011	429.1537	-329.8774	450.5391	496.2403	-299.4651
334.0766	496.3975	-471.4971	394.2121	488.0435	-404.8359	481.4139	433.0168	-310.9362	484.672	427.0046	-323.1021	443.0604	509.1384	-299.7451
330.8843	495.6474	-474.4691	390.4353	488.3046	-410.4464	475.2815	445.7199	-308.1249	486.8819	421.1788	-320.8176	440.9456	512.419	-300.1507
323.9404	492.8854	-479.168	383.608	487.5842	-417.5798	466.9132	459.8522	-303.7738	489.1469	416.0701	-319.7548	440.5456	513.3033	-299.3131
317.5507	492.3441	-485.9497	377.0336	486.8906	-427.2603	457.1979	476.5716	-299.5056	490.7452	412.3895	-319.4732	439.9344	513.2129	-298.2525
307.9256	490.9984	-495.7319	369.7393	485.733	-438.2218	452.5023	484.9458	-298.2885	493.7877	405.1799	-320.57	442.7167	508.9027	-297.2598
302.7784	490.2787	-501.8926	363.556	485.0867	-448.135	449.3077	490.695	-298.7512	497.1661	394.6546	-321.2916	443.3987	508.3373	-295.6138
300.8829	490.0137	-503.646	359.8033	485.3753	-451.7782	447.3227	494.2137	-299.059	501.8787	378.7247	-320.9153	440.4285	511.8729	-294.5042
295.6941	488.124	-506.0157	354.5494	484.5929	-455.0536	447.6037	493.778	-298.1998	508.0035	360.016	-320.2537	438.7917	512.6682	-293.6846
289.752	487.949	-510.9497	350.3543	484.1494	-458.7988	449.2256	491.1519	-296.1419	509.6635	354.5305	-320.1577	438.4266	512.1455	-291.8687
279.303	485.9526	-519.1234	342.3603	483.3044	-465.6519	448.5102	491.0815	-294.9409	510.2257	351.9226	-320.3465	438.8083	509.3914	-289.7149
270.6859	484.7771	-526.8846	333.9169	484.0367	-474.3586	446.5353	492.6463	-293.3886	508.392	361.0996	-319.8106	434.5701	508.2425	-291.4774
260.7915	483.6003	-534.3534	324.4138	486.8629	-482.21	445.347	492.7098	-291.4172	506.0027	370.4965	-320.4725	430.5412	507.9585	-301.585
253.2126	483.4049	-539.9383	317.1163	489.1181	-488.475	444.6554	492.6389	-289.6809	505.4204	373.3577	-321.2596	429.4428	508.6298	-308.5141
248.7987	482.6761	-542.826	314.4421	489.2452	-489.3104	438.7594	490.3043	-295.2592	504.9724	376.0037	-321.3536	429.2194	508.415	-311.1473
246.2563	482.1566	-544.2949	311.8135	488.8982	-490.5654	436.4614	489.8964	-306.6015	504.0825	380.5217	-320.2178	429.3297	507.2406	-312.6564
242.5124	481.4199	-545.4565	305.7125	488.0928	-492.9423	433.8219	489.9966	-317.0832	503.3814	382.7143	-320.565	427.7704	507.0251	-317.7478
235.1506	480.6127	-549.8485	297.5804	488.1861	-499.7905	434.2364	490.0377	-321.8556	501.9423	388.0428	-320.6594	425.5832	506.9042	-325.8922
228.9918	479.9622	-554.126	291.23	488.4086	-505.5268	432.8152	489.8968	-324.3653	500.6453	393.049	-319.9337	422.8802	507.3417	-333.0606
224.3755	479.2158	-557.1009	286.2757	488.2231	-510.4803	431.3423	490.1089	-328.6484	496.0283	407.9337	-319.6185	421.2754	507.5664	-337.9845
220.1602	478.6281	-559.9245	282.0703	487.8802	-514.0337	426.2427	491.5497	-339.191	492.5108	418.231	-319.8394	419.366	507.7449	-340.5686
211.697	477.3044	-564.7927	280.2367	487.6209	-515.8812	419.5472	494.7103	-352.2194	489.1612	427.4904	-319.8079	417.2187	508.4221	-344.2967
202.8112	476.3941	-570.1109	279.112	487.5234	-516.6458	413.5257	495.941	-364.0136	484.8467	438.6869	-319.7765	414.7512	508.0652	-349.3538
194.0991	474.9504	-575.2057	278.4302	487.9801	-516.8895	410.4763	495.7558	-372.2465	480.1691	450.0641	-319.7765	411.703	507.3594	-357.3936
188.2513	474.3958	-578.4523	274.7467	487.4518	-519.0335	409.0581	494.8998	-373.3047	476.5917	457.6735	-319.148	409.4023	506.9402	-363.7646
185.4399	473.8729	-580.0797	265.6932	486.0944	-526.4683	405.2714	492.2456	-373.7032	474.5782	461.2126	-318.1751	405.5773	506.3901	-368.6551
183.3656	473.6675	-581.2071	257.0724	485.6046	-533.5713	403.5544	492.6478	-375.9731	472.5557	463.4115	-315.8902	403.7766	506.3913	-371.9678
192.2114	475.8356	-577.9769	252.5607	485.6251	-537.5577	397.388	492.5391	-382.0839	472.0157	462.9204	-313.1725	401.4174	506.1368	-373.2807
231.9197	489.317	-560.6773	252.4139	485.4337	-537.19	388.7555	491.5562	-397.8403	471.2883	462.424	-309.9679	396.1807	503.3379	-378.797
258.3696	481.7861	-540.4284	251.5056	485.0179	-536.4099	380.5712	490.6244	-414.1361	469.6862	463.2128	-306.6891	390.1691	502.33	-390.0271
			243.9113	482.7564	-539.9165	373.0298	490.2384	-428.494	467.9651	464.5537	-304.1981	384.2526	501.9178	-404.5029
			233.9731	482.764	-546.851	367.0647	487.5375	-439.2427	466.3348	467.5741	-302.4057	382.4102	502.1524	-409.9518
			220.8805	480.427	-556.3624	361.1614	485.5197	-449.6412	464.0154	471.1085	-301.0606	381.3971	501.8449	-410.6218
			209.0621	478.651	-564.6091	356.5164	487.1314	-453.773	461.1042	475.1462	-299.6965	380.7347	501.4238	-411.5062
			198.586	477.1211	-571.3728	350.931	486.6551	-456.5029	460.0692	476.7609	-298.5927	375.3407	499.4561	-414.4689
			195.2222	476.3153	-573.7353	345.9693	484.0506	-463.0544	458.5016	479.7347	-297.6697	349.6333	493.5904	-433.5188
			196.0142	475.441	-573.7685	338.8612	482.5057	-471.9133	455.1561	485.4503	-296.296	340.8587	496.3419	-459.7631
			194.4005	474.859	-574.7621	331.4977	481.8061	-480.6487	450.7589	493.0776	-295.7521	336.3604	499.445	-468.9733
			187.1972	473.8605	-578.8188	329.5002	482.7117	-482.8594	446.0234	501.1615	-295.5131	341.9916	499.2553	-458.7969
			177.9809	472.583	-583.8606	330.5346	482.8933	-480.8234	443.3372	504.7607	-295.1031	347.3759	494.9389	-452.4441
			171.1094	472.6278	-587.0643	330.2241	481.5333	-479.6567	443.1643	505.113	-294.242	349.156	493.4528	-457.2815
			165.6511	470.7579	-590.2889	327.1718	482.1141	-481.2308	442.7653	505.0613	-292.8456	348.5654	498.1904	-456.5469
			162.9196	470.3809	-591.6192	315.5945	481.0774	-489.2621	441.3472	504.8775	-290.7598	348.117	499.1844	-453.6981
			156.5374	469.5369	-593.7657	306.9026	482.1708	-499.3078	440.689	504.5127	-289.2978	346.4256	498.334	-456.5168
			151.2109	469.0871	-596.0654	299.8289	487.5116	-503.7005	438.5388	502.9386	-290.5398	345.1476	497.0998	-458.1372
			146.4087	468.3808	-598.6147	297.299	488.7322	-503.6845	436.2158	502.6453	-300.3729	337.5836	496.3174	-459.4014
			144.2262	468.0761	-599.447	296.0605	487.0764	-506.0532	435.1573	502.7874	-306.5122	332.3352	496.8703	-466.1083
			141.5134	467.6974	-600.5973	294.7768	487.2271	-507.2061	435.4968	501.3605	-310.0964	329.7214	496.415	-469.9882
			136.0462	466.9341	-602.7159	293.7284	488.4986	-507.0694	435.3521	500.4258	-312.4019	330.0316	496.4604	-469.9371
			128.8184	465.9249	-605.5109	292.3238	488.1765	-507.7401	434.3378	500.3024	-315.9442	331.684	495.6902	-468.5504
			124.9215	465.6563	-606.894	286.1554	487.1382	-512.9862	432.675	500.3735	-321.6536	331.7127	494.6123	-469.4788
			124.3507	465.4535	-607.1979	276.084	486.2423	-522.2526	431.0937	500.271	-327.0123	329.4777	494.3696	-472.3239
			121.3293	464.1066	-608.5523	267.8982	485.106	-529.9183	429.6445	500.1842	-331.7238	325.0386	493.6738	-476.2919
			115.1125	463.3561	-610.7723	261.5966	484.7527	-534.2969	428.1446	500.5419	-334.6705	317.7108	492.5049	-485.2324

ISAC

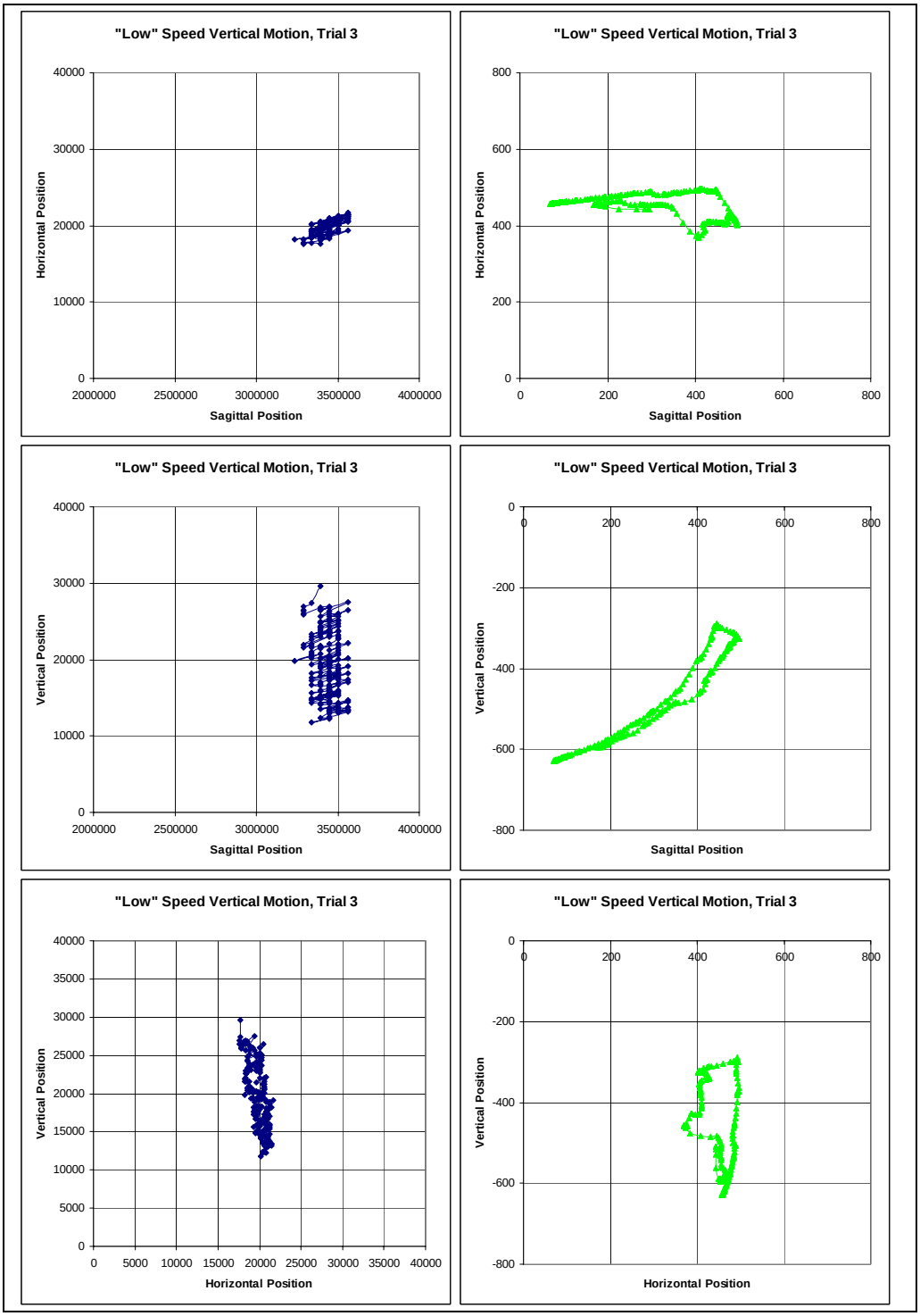
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
			110.0587	462.7285	-612.5237	260.6721	484.7962	-534.6782	425.8617	501.2485	-338.6075	314.0315	492.5372	-490.0754
			108.4363	462.7257	-613.1806	259.8062	484.2705	-533.6043	422.2832	502.9438	-344.606	312.3492	492.3669	-491.0856
			107.9446	462.6042	-613.3684	258.1815	483.9861	-534.0256	418.186	504.987	-350.124	312.2048	491.3244	-490.36
			107.9446	462.6042	-613.3684	251.3543	483.2566	-536.6737	414.1221	505.0503	-357.0991	310.4634	490.8793	-492.3743
			107.9872	462.583	-613.3684	246.3146	482.6588	-540.6133	411.8836	504.6534	-362.7595	304.9105	490.6435	-495.6732
			107.8223	462.5605	-613.4219	238.6941	481.4255	-545.946	409.4881	504.3233	-368.2917	300.6783	490.0509	-500.9098
			106.9978	462.4481	-613.6886	232.2066	480.6721	-550.7338	405.955	503.8364	-372.4389	292.1322	488.6621	-507.7549
			105.5398	462.5416	-614.0063	229.1949	480.248	-553.2955	400.2965	503.0566	-376.8188	289.1731	489.0754	-513.1304
			103.5187	462.6032	-614.7342	223.3042	479.1669	-557.2411	395.9085	502.5366	-382.6282	291.8421	489.6469	-509.9423
			103.4762	462.6236	-614.7342	218.0374	478.9233	-560.5456	388.9704	501.4957	-393.2385	309.8054	492.4123	-496.85
			103.3935	462.6119	-614.7603	210.0243	477.0247	-566.0817	379.7639	499.3311	-410.6659	315.5257	491.0995	-481.869
			103.1456	462.5768	-614.8385	202.0852	476.7533	-570.8045	372.2134	499.5064	-424.8648	317.8983	487.9854	-485.1363
			103.0187	462.5589	-614.9568	195.0536	475.7533	-574.8791	365.1267	499.1534	-434.8926	309.6588	491.5788	-493.8352
			103.0039	462.5568	-614.9875	192.4875	475.3883	-576.4433	361.9879	498.7133	-440.218	293.6215	489.256	-505.4318
			102.7942	462.5271	-615.1317	192.2619	475.3562	-576.5746	356.3093	497.225	-444.2085	269.2247	483.8671	-529.6742
						188.9777	474.0639	-578.3803	351.47	496.6311	-449.7808	254.4571	484.712	-542.4418
						179.9867	473.2769	-583.0274	344.9305	495.8738	-457.7944	250.213	486.233	-544.2851
						172.2924	472.5161	-587.0077	337.5138	494.8427	-467.6926	252.508	485.278	-542.8942
						164.6134	471.1923	-590.8606	333.7566	494.7548	-472.5089	257.0282	482.914	-539.9269
						161.3722	471.3816	-592.0427	332.8196	494.5512	-473.2852	247.7713	482.9744	-543.6932
						154.4678	469.1781	-595.6366	333.0664	493.9357	-470.0497	235.9327	483.2667	-551.23
						148.6496	469.6508	-597.4639	328.6347	492.682	-471.1824	228.5002	482.1102	-556.6031
						138.9639	467.5084	-601.5958	322.0273	492.6199	-478.3958	228.8422	481.4361	-556.5209
						130.1985	467.0068	-604.8267	314.5676	491.5844	-487.4845	231.1174	479.1729	-556.2677
						127.552	466.5597	-605.8588	309.0142	490.8135	-493.981	231.237	479.4475	-555.6233
						127.1271	466.2796	-606.1373	307.8166	491.1848	-495.2216	226.2881	479.4315	-558.2655
						125.6562	466.0687	-606.7219	309.0248	491.0172	-493.4053	215.7672	477.4314	-565.1457
						119.3891	464.933	-609.2086	307.8354	490.2471	-494.7947	205.1217	476.7647	-571.5039
						111.67	464.0627	-611.8079	300.887	489.2909	-501.2322	196.4752	475.7304	-576.4436
						106.2073	463.3332	-613.801	295.2099	489.4146	-506.5969	195.2825	475.6515	-576.7219
						102.6852	462.8793	-614.8479	291.3037	488.8661	-510.542	192.9134	474.962	-577.8305
						101.984	462.752	-615.0817	288.0249	488.849	-513.47	185.8895	473.5938	-581.7601
						100.8861	462.4383	-615.5555	283.0349	488.1413	-516.7802	178.7261	473.1411	-585.5644
						96.51926	461.7896	-616.9217	276.5985	487.3506	-522.4545	176.5278	473.0775	-587.062
						91.80101	461.2371	-618.2651	269.4791	486.2785	-528.654	176.3275	472.5988	-586.9495
						91.04627	461.1287	-618.7654	261.9099	485.1453	-534.8475	171.861	471.7746	-588.1465
						91.51536	461.1961	-618.6835	256.0662	485.0008	-539.466	165.4128	471.3051	-591.2707
						93.45917	461.4753	-618.3568	252.6591	484.961	-541.5864	158.6661	470.2789	-594.392
						94.46902	461.4726	-618.2661	252.3614	484.8048	-541.7174	157.6895	470.2531	-595.3359
						94.46902	461.4726	-618.2661	250.6124	484.3823	-542.9689	158.3113	470.1172	-594.456
						94.02149	461.4089	-618.4484	246.6353	483.8063	-545.9579	156.8472	469.6535	-594.3776
						94.02149	461.4089	-618.4484	242.2159	483.1662	-549.1562	153.6801	469.5051	-595.619
						92.50497	461.1929	-618.9127	239.946	482.8914	-550.7344	146.7819	468.1537	-598.3598
						89.15334	460.6685	-620.039	237.7838	482.5778	-551.7562	143.6166	468.3346	-599.7378
						86.16274	460.2896	-621.7372	235.7187	482.2783	-552.9646	141.5862	468.047	-600.6806
						85.33188	460.1713	-622.3459	228.117	480.6097	-557.1696	142.2262	468.0694	-600.0955
						83.04297	459.8007	-623.0002	221.9592	479.9812	-561.1421	140.5188	467.6596	-600.7429
						80.52483	459.4867	-624.5247	212.4509	478.4205	-567.6599	135.3718	466.9379	-602.7205
						77.24175	458.9768	-625.5992	207.4705	478.2289	-570.7687	130.5476	466.3884	-604.6981
						75.92038	458.8309	-626.1062	203.081	477.4519	-573.5688	125.0587	465.4612	-606.7958
						75.64885	458.7922	-626.1977	200.8468	477.1288	-574.9088	123.9259	465.6069	-607.1638
						73.74096	458.3573	-626.7868	194.971	475.2055	-578.9152	124.0786	465.4153	-607.1926
						72.06525	458.1414	-627.3234	189.9871	474.7268	-581.8238	124.9916	465.299	-606.9812
						72.19774	458.1199	-627.4894	189.5074	474.7464	-582.0728	120.7227	464.7055	-608.4681
						70.87692	457.9144	-627.8048	190.2666	474.3296	-581.9764	112.6578	463.6125	-611.3038
						68.88156	457.6342	-628.3092	182.7615	472.8672	-586.2881	106.9671	463.1984	-612.9973
						68.93177	457.6413	-628.3485	171.4928	472.4024	-591.9143	104.5881	462.8606	-613.8958
						68.97335	457.6277	-628.3485	167.4026	471.8991	-594.1365	103.7843	462.5616	-614.2985
									170.0003	471.9917	-592.9713	101.3053	462.213	-615.0746
									169.9588	471.1945	-593.1874	99.11129	461.9299	-616.0219
									161.7246	470.0647	-596.0661	100.6757	462.3308	-615.8164
									149.4316	469.1225	-600.8839	108.2103	463.375	-614.3723
									143.6169	468.4723	-603.6247	113.0032	463.3233	-613.2002
									143.9966	468.3884	-603.0817	110.2741	463.4465	-613.6731
									145.0012	468.3574	-602.4243	108.1497	463.3937	-615.0052
									144.6033	468.3014	-602.4059	107.872	462.6754	-615.6248
									142.5914	468.0185	-602.9089	98.92669	461.3975	-619.3684
									137.3622	466.9196	-604.5423	91.22431	460.9148	-623.184
									132.9099	467.0444	-605.7115	85.5236	460.153	-625.0529
									130.6612	466.4998	-606.6324	83.43235	459.455	-625.7505
									129.7993	466.0304	-606.5641	82.58602	459.426	-625.9259
									123.0196	464.9344	-609.0318	79.70043	459.0673	-626.6379
									114.58	464.1653	-611.6322	76.80911	458.6636	-627.3523
									106.2577	462.99	-614.2782	72.88841	458.1162	-628.4146
									105.7008	463.0723	-614.323	68.15691	457.5325	-629.633
									105.4378	462.5541	-614.5746	66.08604	457.2793	-631.9048
									103.4941	462.6261	-614.9898	64.42109	457.0264	-632.847
									101.0517	462.3066	-615.7396	61.84226	456.6638	-633.6117
									100.4718	462.2244	-615.9186	58.50233	456.1941	-635.4068
									100.2232	462.1892	-615.995	56.46932	455.9082	-636.7709
									100.3061	462.201	-615.9695	54.92642	455.6913	-637.2629
									99.24344	462.0505	-616.2686	53.26409	455.4575	-637.697
									96.91953	461.7214	-616.9695			
									95.72555	461.5523	-617.376			
									96.18516	461.5427	-617.2776			
									96.34118	461.5398	-617.192			
									96.31063	461.5355	-617.253			
									96.31063	461.5355	-617.253			



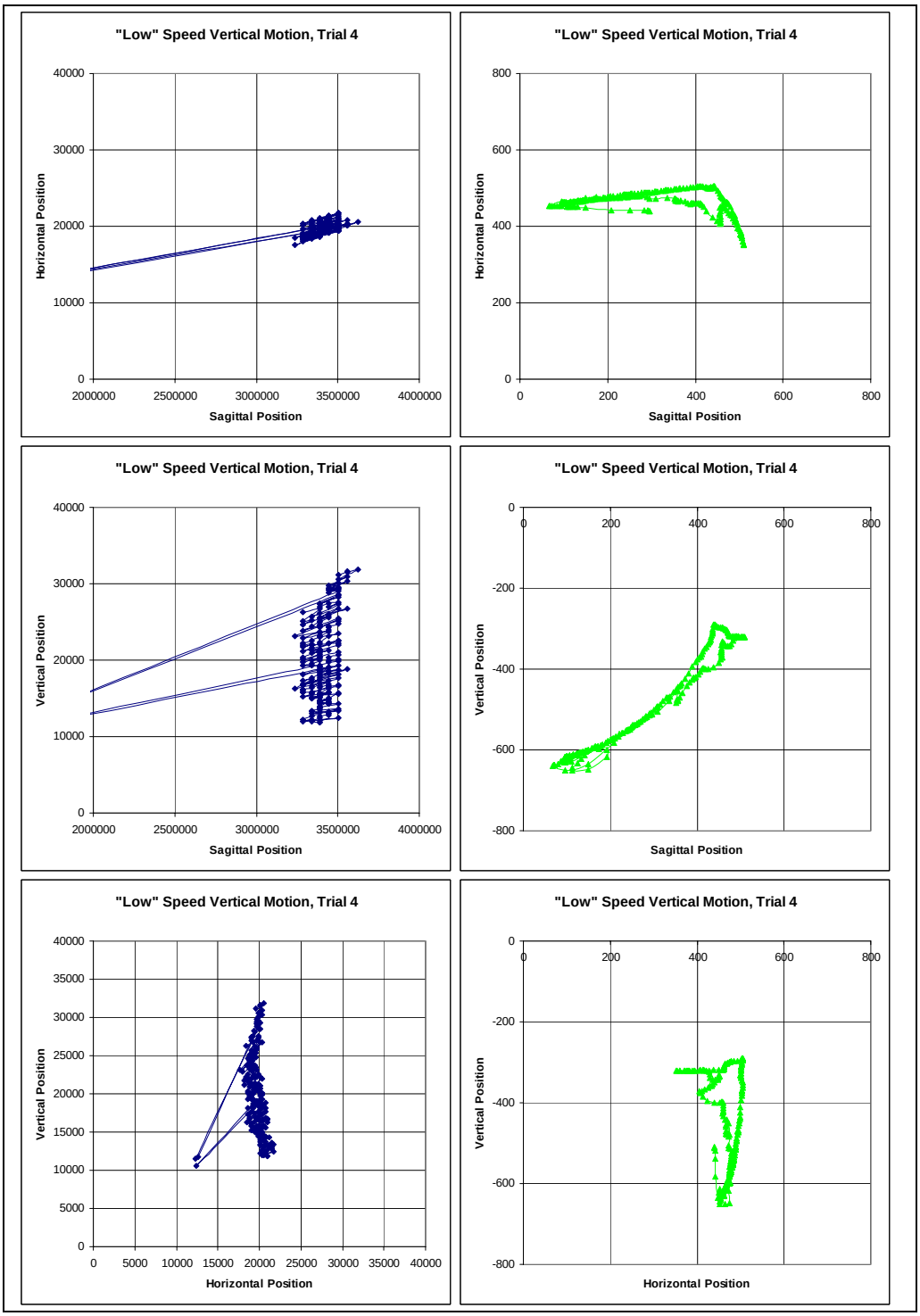
"Low" speed vertical motion, trial 1; human motion is shown on the left, that of ISAC on the right



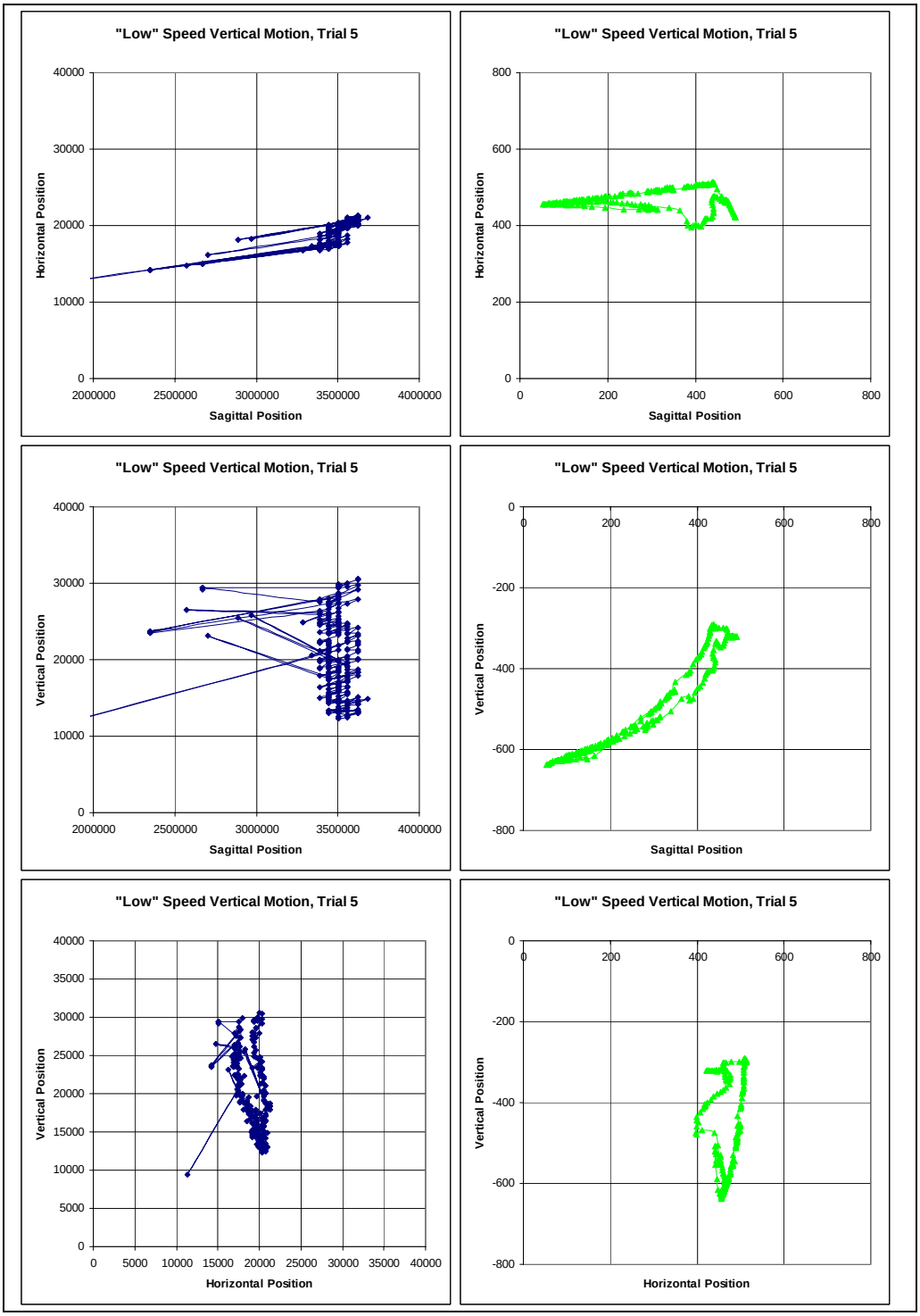
"Low" speed vertical motion, trial 2; human motion is shown on the left, that of ISAC on the right



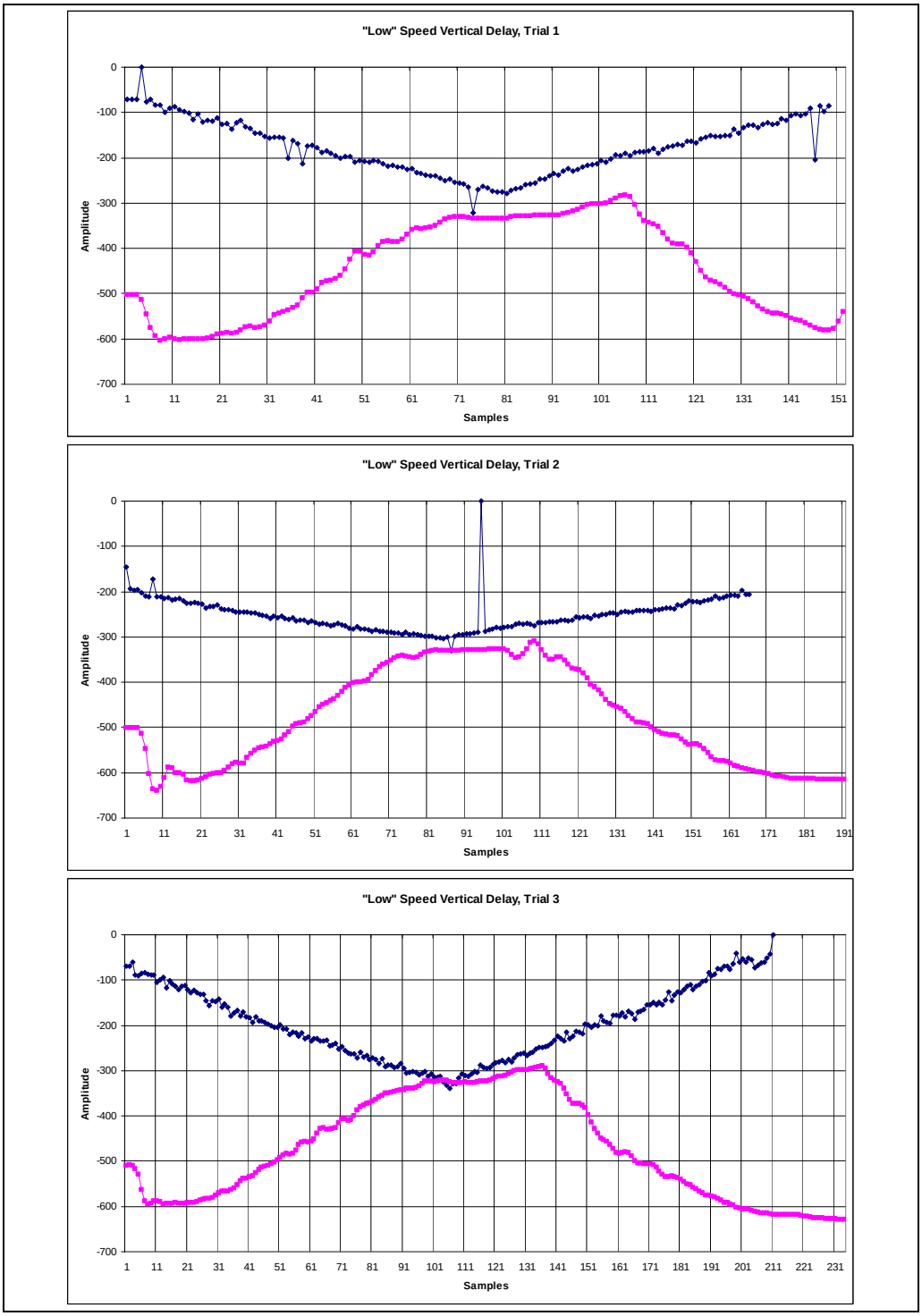
"Low" speed vertical motion, trial 3; human motion is shown on the left, that of ISAC on the right



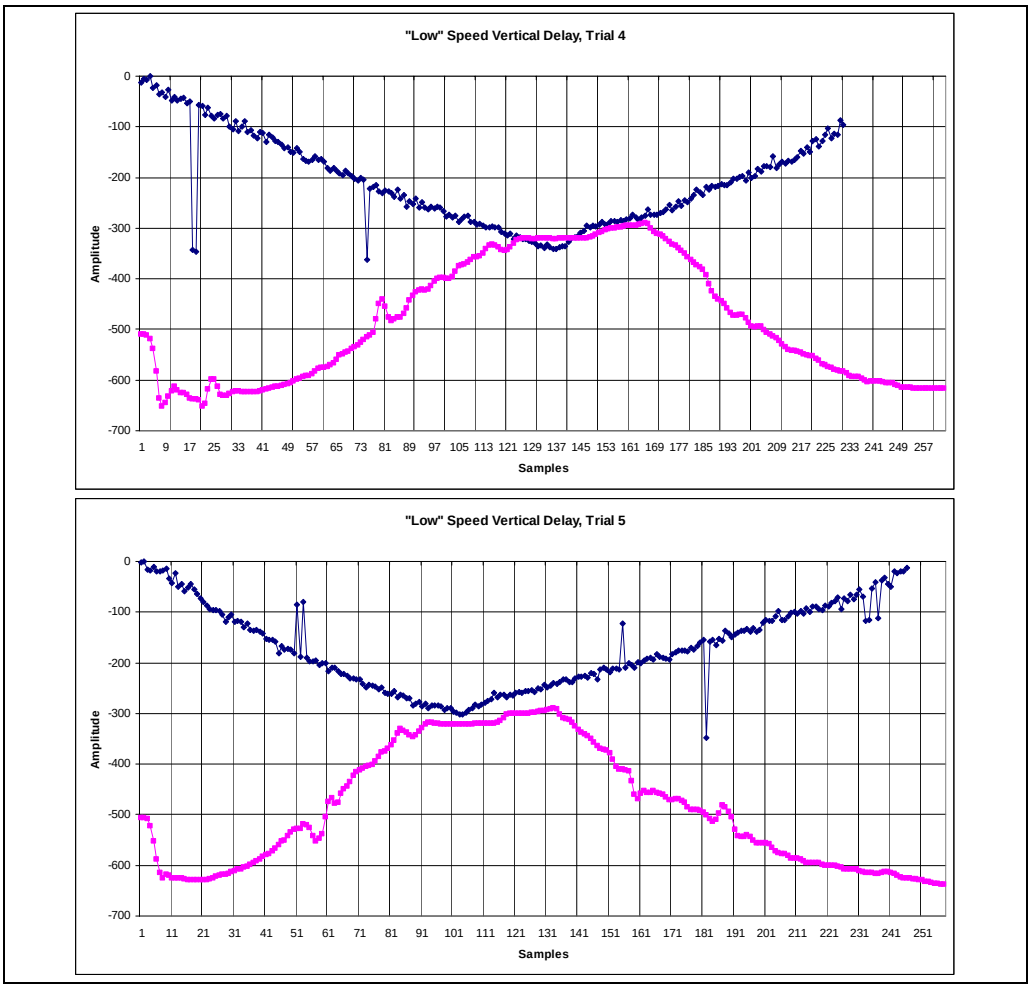
"Low" speed vertical motion, trial 4; human motion is shown on the left, that of ISAC on the right



"Low" speed vertical motion, trial 5; human motion is shown on the left, that of ISAC on the right



"Low" speed vertical delay, trials 1-3; human motion is shown on the on top, that of ISAC on the on bottom



"Low" speed vertical delay, trials 4-5; human motion is shown on the on top, that of ISAC on the on bottom

Vertical motion at "Medium" speed

Human

Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
3446371	18967.74	30822.58	5623027	26157.89	44394.74	3446371	17387.1	28112.9	3561250	18200	30683.33	3561250	18666.67	34066.67
3446371	18967.74	30709.68	3446371	18290.32	31161.29	3391667	17000	27777.78	3561250	18200	30566.67	3621610	19101.7	34762.71
3446371	18967.74	30596.77	3621610	19338.98	32983.05	3446371	17161.29	28112.9	3621610	18627.12	30966.1	3561250	18666.67	34183.33
3391667	19000	29777.78	3561250	19133.33	32433.33	3561250	17733.33	28933.33	3561250	18200	30216.67	3561250	18666.67	33833.33
3446371	19193.55	30145.16	3502869	18704.92	31672.13	3446371	17161.29	27774.19	3446371	17387.1	29016.13	5087500	23666.67	40833.33
3391667	18777.78	29000	3391667	18111.11	30444.45	3446371	17161.29	27548.39	3561250	17733.33	29400	3561250	18666.67	32900
3338672	18593.75	28000	3446371	18290.32	30483.87	3446371	17387.1	26983.87	3502869	17327.87	28114.75	3561250	18900	31850
3446371	19193.55	28451.61	3561250	19133.33	30683.33	3446371	17387.1	26419.36	3446371	16935.48	27209.68	3561250	19133.33	30916.67
3391667	19000	27666.67	3446371	18741.94	29129.03	3391667	17222.22	25333.33	3446371	16935.48	26532.26	3502869	18704.92	29606.56
3502869	19163.93	27885.25	3502869	19163.93	28803.28	3446371	17387.1	25177.42	3446371	16709.68	25967.74	3391667	18111.11	28000
3561250	19600	27766.67	3561250	19600	28816.67	3338672	16843.75	23734.38	3502869	16868.85	25819.67	2811513	16394.74	30486.84
3391667	18777.78	25888.89	3502869	19393.44	27655.74	3391667	17000	23777.78	3502869	17098.36	25245.9	3446371	18290.32	26983.87
3446371	18741.94	25629.03	3561250	19600	27416.67	3338672	17062.5	22968.75	3391667	16777.78	23777.78	3502869	18475.41	26737.71
3446371	18741.94	25064.52	3502869	19393.44	26852.46	3338672	16625	22421.88	3502869	17557.38	23983.61	3391667	17666.67	25333.33
3502869	19163.93	24672.13	3446371	18967.74	26193.55	3446371	16935.48	22693.55	3391667	17222.22	23000	3561250	18433.33	26133.33
3391667	19000	23333.33	3446371	18741.94	25629.03	3446371	17161.29	22467.74	3446371	17387.1	22580.64	3446371	17612.9	24725.81
3446371	19419.36	22806.45	3502869	19163.93	25360.66	3502869	17327.87	22377.05	3502869	17786.88	22606.56	3446371	17612.9	24274.19
3502869	19852.46	22606.56	3446371	19193.55	24500	3502869	17327.87	21918.03	3391667	17000	21222.22	3446371	17387.1	23822.58
3446371	19645.16	21451.61	3446371	18967.74	23935.48	3502869	17327.87	21573.77	3446371	16935.48	20887.1	3446371	17161.29	23258.06
3502869	20081.97	21344.26	3502869	19163.93	23754.1	3446371	17161.29	20774.19	3446371	16935.48	20209.68	3502869	17327.87	23180.33
3502869	20081.97	20885.25	3391667	18555.55	22444.45	3446371	17161.29	20661.29	3561250	17500	20300	3502869	17557.38	22491.8
3502869	20311.47	20311.47	3391667	18555.55	22000	3391667	16777.78	20000	3391667	16777.78	19000	3561250	17966.67	22400
3446371	20096.77	19419.36	3446371	18967.74	22016.13	3338672	16625	19140.63	3446371	16935.48	18629.03	3446371	17387.1	21338.71
3446371	20322.58	18967.74	3391667	18333.33	21000	3446371	17161.29	19419.36	3502869	17098.36	18360.66	3502869	17557.38	21344.26
3446371	20096.77	18516.13	3391667	18333.33	20777.78	3391667	16555.55	18555.55	3502869	17327.87	17557.38	3502869	17557.38	20885.25
3502869	20311.47	18475.41	1515426	12262.41	9581.561	3391667	16555.55	18222.22	3502869	17327.87	17213.12	3502869	17557.38	20426.23
3446371	20096.77	17612.9	3338672	18375	19687.5	3502869	17327.87	18475.41	3446371	17161.29	16258.06	3502869	17327.87	19967.21
3561250	20766.67	17616.67	3391667	18555.55	19555.55	3446371	17161.29	17838.71	3502869	17327.87	16180.33	3502869	17327.87	19622.95
3446371	20322.58	16822.58	3391667	18555.55	19222.22	3391667	17000	17111.11	3446371	16935.48	15241.94	3502869	17327.87	18934.43
3561250	21000	16916.67	3446371	18741.94	19193.55	3391667	16777.78	16777.78	3446371	16935.48	14790.32	3502869	17098.36	18819.67
3502869	21000	16180.33	3502869	19163.93	19393.44	3391667	16777.78	16222.22	3446371	16935.48	14338.71	3502869	17098.36	18590.16
3446371	20548.39	15467.74	3391667	18555.55	18333.33	3338672	16625	15640.63	3446371	16935.48	13774.19	3446371	17161.29	17612.9
3561250	21000	15400	3502869	19163.93	18360.66	3391667	16777.78	15555.56	3446371	16709.68	13435.48	3502869	17327.87	17442.62
3561250	21000	15050	3502869	19163.93	18245.9	3391667	17000	15222.22	3446371	16709.68	12870.97	3502869	17327.87	16983.61
3502869	21000	14229.51	3446371	18967.74	17838.71	3391667	16777.78	14777.78	3446371	16483.87	12419.35	3502869	17327.87	16754.1
3561250	21466.67	14000	3446371	18967.74	17387.1	3338672	16406.25	14218.75	3446371	16709.68	11967.74	3502869	17098.36	16295.08
3446371	20774.19	13322.58	3502869	19163.93	17442.62	3391667	16777.78	14222.22	3391667	16333.33	11333.33	3502869	17098.36	16065.57
3502869	21229.51	13081.97	3446371	18967.74	16935.48	3391667	16555.55	13888.89	3446371	16935.48	11177.42	3621610	17440.68	16016.95
3502869	21000	12622.95	3446371	18741.94	16822.58	3391667	16777.78	13444.44	3502869	17557.38	11016.39	3502869	17098.36	15262.29
3502869	21229.51	12393.44	3391667	18555.55	16444.45	3391667	16777.78	13333.33	3446371	17161.29	10612.9	3502869	17327.87	14918.03
3446371	20774.19	11854.84	3446371	18741.94	16370.97	3391667	17000	12888.89	3502869	17327.87	10672.13	3502869	17327.87	14459.02
3446371	20774.19	11629.03	3391667	18333.33	15555.56	3391667	17000	12888.89	3391667	16777.78	10222.22	3502869	17327.87	13770.49
3502869	21000	11360.66	3502869	18934.43	15950.82	3287308	16476.92	11953.85	3338672	16625	10281.25	3446371	17387.1	13209.68
3502869	21000	11131.15	3502869	18934.43	15721.31	3338672	16625	11812.5	3338672	16625	10390.63	3391667	17222.22	12555.56
3446371	20774.19	11177.42	3502869	19163.93	15262.29	3338672	16625	11812.5	3338672	16625	10937.5	3446371	17612.9	12532.26
3391667	20111.11	11777.78	3502869	19393.44	15147.54	3391667	16777.78	12444.44	3391667	17000	11444.44	3561250	17966.67	12600
3446371	20322.58	12193.55	3391667	19000	14333.33	3338672	16187.5	12796.88	3338672	16625	11703.13	3391667	17222.22	11444.44
3502869	20540.98	12967.21	3502869	19622.95	14344.26	3338672	15968.75	13125	3338672	16843.75	12140.63	3502869	18016.39	11475.41
3446371	20322.58	13322.58	3502869	19622.95	14229.51	3391667	16111.11	13777.78	3502869	17327.87	13311.48	3338672	17281.25	10718.75
3446371	20322.58	13661.29	3391667	18777.78	13222.22	3338672	15968.75	14000	3391667	17000	13111.11	3561250	18433.33	11083.33
3446371	20322.58	14225.81	3391667	18777.78	12888.89	3338672	15968.75	14656.25	3446371	16935.48	14225.81	3446371	17838.71	10951.61
3502869	20540.98	14918.03	3446371	19193.55	12532.26	3391667	16111.11	15000	3446371	17161.29	14564.52	3502869	18016.39	11245.9
3391667	19888.89	14777.78	3391667	18777.78	12000	3391667	16111.11	15666.67	3391667	17000	14666.67	3446371	17387.1	11403.23
3446371	20096.77	15580.65	3446371	18967.74	11854.84	3338672	15968.75	15968.75	3338672	16843.75	15203.13	3502869	18016.39	11934.43
3502869	20081.97	16295.08	3391667	18555.55	11333.33	3338672	15750	16296.88	3446371	17387.1	16370.97	3338672	16843.75	12031.25
3391667	19444.45	16111.11	3446371	18967.74	11290.32	3338672	15750	16734.38	3446371	17387.1	16935.48	3391667	17222.22	12555.56
3446371	19645.16	16935.48	3338672	18375	10609.38	3391667	16111.11	17444.45	3446371	17387.1	17274.19	3391667	17222.22	13000
3446371	19645.16	17274.19	3446371	19193.55	10951.61	3287308	15400	17123.08	3338672	16625	17281.25	3446371	17387.1	13774.19
3446371	19645.16	17725.81	3446371	19193.55	10612.9	3237500	15272.73	17287.88	3502869	17557.38	18704.92	3446371	17387.1	14225.81
3391667	19222.22	17777.78	3338672	18593.75	10062.5	3237500	15060.61	17606.06	3391667	16777.78	18444.45	3502869	17786.88	14803.28
3446371	19419.36	18516.13	3446371	19193.55	10161.29	3287308	15184.62	18200	3446371	17161.29	19080.64	3446371	17612.9	15241.94
3446371	19419.36	18854.84	3391667	19222.22	10222.22	3287308	15184.62	18738.46	3446371	16935.48	19532.26	3502869	17786.88	15836.07
3446371	19193.55	19193.55	3446371	19645.16	10500	3237500	14848.48	18772.73	3502869	17327.87	19967.21	3502869	17786.88	16295.08
3446371	19193.55	19645.16	3391667	19222.22	10222.22	3237500	14848.48	19090.91	3502869	17327.87	20426.23	3446371	17612.9	16596.77
3502869	19622.95	20311.47	3446371	19193.55	10951.61	3287308	14969.23	19815.38	3446371	17161.29	20548.39	3446371	17612.9	16822.58
3391667	18555.55	20111.11	3446371	18967.74	11290.32	3287308	14969.23	20138.46	3446371	17161.29	20887.1	3446371	17838.71	17048.39
3446371	18741.94	20661.29	3446371	19193.55	11741.94	3338672	15093.75	20890.63	3391667	16777.78	21111.11	3338672	17281.25	16953.13
3391667	18555.55	20888.89	3446371	19193.55	12080.65	1249561	10029.24	9865.497	3446371	17161.29	22016.13	3391667	17444.45	17666.67
3446371	18741.94	21564.52	3502869	19163.93	12622.95	1249561	10029.24	9947.368	3391667	16555.55	22000	3391667	17444.45	18222.22
3446371	18741.94	22129.03	3391667	18555.55	12555.56	3287308	14969.23	21969.23	3502869	17327.87	23065.57	3502869	18016.39	19163.93
3502869	18934.43	22836.07	3446371	18290.32	13209.68	3287308	14753.85	22400	3446371	17161.29	23145.16	3446371	17612.9	19419.36
3446371	18516.13	23032.26	3391667	18111.11	13333.33	1249561	10029.24	10274.85	3446371	16935.48	23709.68	3391667	17222.22	19666.67
3561250	19133.33	24033.33	3446371	18516.13	14000	3287308	14753.85	23261.54	3391667	16555.55	24000	3502869	17786.88	21000
3446371	18741.94	23709.68	3391667	18111.11	14222.22	3237500	14636.36	22909.09	3446371	16935.48	24612.9	3391667	17222.22	21000
3446371	18741.94	24048.39	3446371	18290.32	14790.32	3287308	14753.85	23584.62	3391667	16555.55	24666.67	3391667	17222.22	21555.55
3391667	18555.55	24000	3391667	18111.11	14888.89	3287308	14753.85	24230.77	3446371	16935.48	25629.03	3391667	17000	22222.22
3502869	19163.93	25245.9	3391667	17888.89	15222.22	3237500	14636.36	23757.58	3446371	16935.48	25741.94	3502869	17786.88	23409.84
3561250	19366.67	26133.33	3391667	18111.11	15777.78	3237500	14424.24	24287.88	3446371	17161.29	26193.55	3391667	17222.22	23333.33
3502869	18934.43	25934.43	3338672	17718.75	16078.13	3287308	14753.85	24876.92	3391667	17222.22	26666.67	3446371	17387.1	24161.29
3391667	18333.33	25666.67	3391667	17666.67	16666.67	3237500	14212.12	24818.18	3446371	17612.9	27548.39	3446371	17387.1	24725.81
3561250	19366.67	27300	3446371	18290.32	17274.19	3287308	14323.08	25738.46	3287308	16692.31	26600	3502869	17786.88	25934.43
3502869	18934.43	27426.23	3391667	18111.11	17666.67	3287308	14323.08	26061.54	3446371	17612.9	28338.71	3391667	17222.22	25666.67
3446371	18741.94	27096.77	3338672	17500	17828.13	3391667	14777.78	27222.22	3446371	17387.1	28790.32	3446371	17387.1	26645.16
3446371	18741.94	27548.39	3338672	17062.5	18375	3287308	14323.08	26707.69	3391667	17222.22	29000	3391667	17000	26666.67
3502869	19163.93	28573.77	3338672	17062.5	18812.5	2428125	12568.18	26647.73	3391667	17000	29111.11	3391667	17222.22	27333.33
3446371	18741.94	28338.71	3446371	17612.9	19983.87	3287308	13892.31	27353.85	3502869	17786.88	30524.59	3446371	17387.1	28112.9
3502869	18934.43	29262.29	3502869	18016.39	20770.49	3237500	13787.88	27575.76	3446371	17612.9	30483.87	3446371	17838.71	28903.23

Human

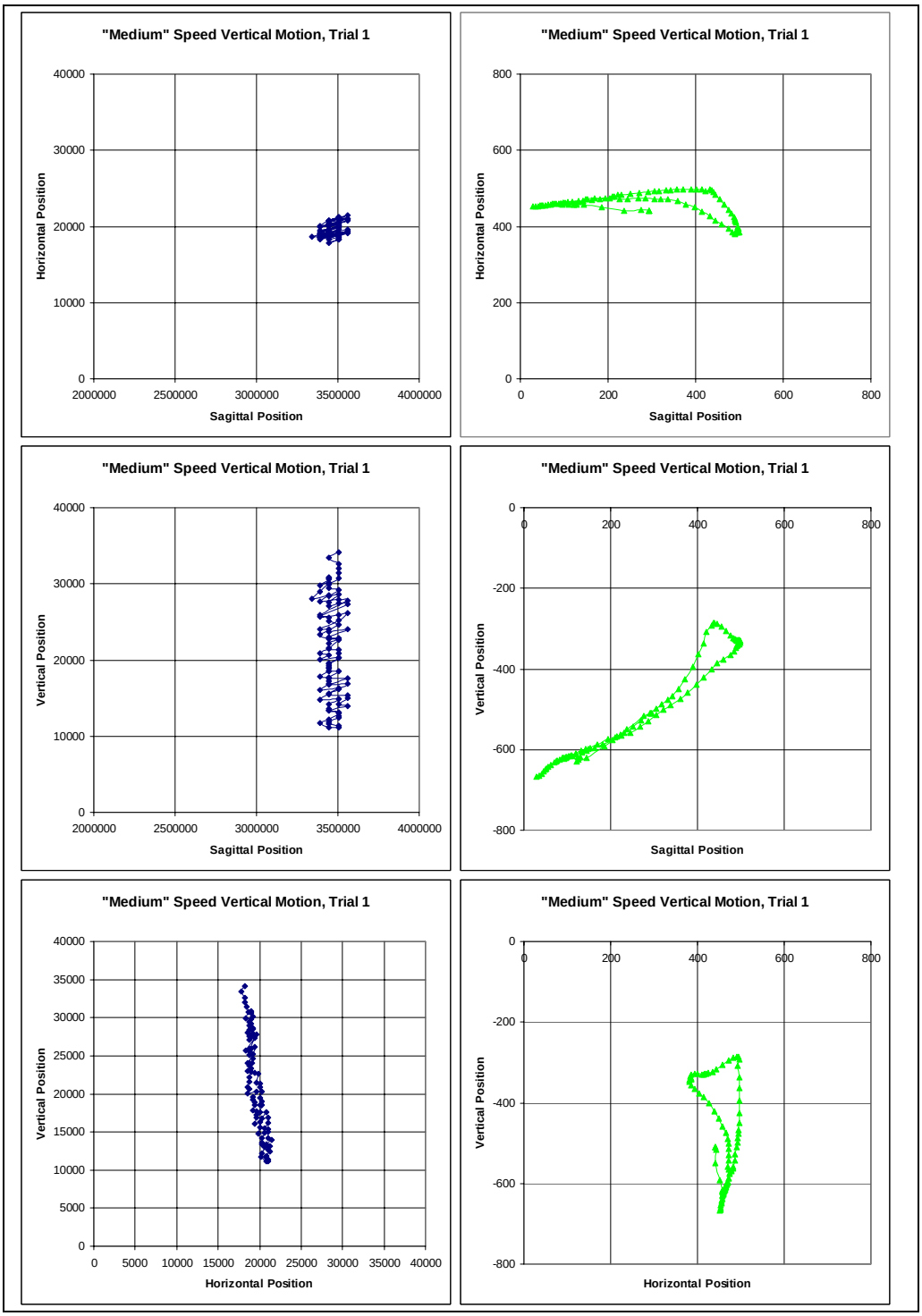
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
3446371	18741.94	29467.74	3338672	17281.25	20453.13	3287308	14107.69	28107.69	3561250	18200	31733.33	3446371	17838.71	29354.84
3446371	18290.32	29919.36	3502869	18016.39	21803.28	3338672	14437.5	28875	3446371	17612.9	31161.29	3561250	18200	31150
3502869	18704.92	30754.1	3446371	17838.71	21677.42	3391667	14555.56	29555.55	3502869	18016.39	32131.15	3391667	17444.45	30222.22
3502869	18475.41	31442.62	3338672	17281.25	21546.88	3287308	13892.31	29076.92	3446371	17838.71	31951.61	3446371	17838.71	31048.39
3502869	18245.9	32016.39	3287308	16907.69	21538.46	3446371	14677.42	30709.68	3446371	17612.9	32177.42	3391667	17666.67	31111.11
3502869	18245.9	32590.16	3391667	17222.22	22777.78	3287308	13892.31	30261.54	3561250	18200	33833.33	3502869	18245.9	32475.41
3446371	17838.71	33419.36	3391667	17444.45	23222.22	3237500	13575.76	30757.58	3391667	17222.22	32888.89	3561250	18666.67	33600
3502869	18245.9	34081.97	3446371	17612.9	23822.58	3287308	13892.31	32415.38	3338672	16843.75	32593.75	3446371	18064.52	32854.84
			3446371	17838.71	24387.1	3237500	13575.76	32136.36	3446371	17387.1	33983.87	3338672	17718.75	32156.25
			3446371	17387.1	25064.52	3391667	14333.33	34222.22	4645109	20695.65	38043.48	4969186	22953.49	40860.46
			3446371	17161.29	25854.84							3446371	18290.32	34096.77
			3446371	16935.48	26306.45							3391667	17888.89	33888.89
			3446371	16935.48	28000							3502869	18245.9	35459.02
			3502869	17327.87	29032.79							3287308	17338.46	33600
			3446371	16935.48	28903.23							3502869	18245.9	36491.8
			3446371	16935.48	29129.03									
			3502869	17098.36	29836.07									
			3446371	16483.87	30822.58									
			3561250	17266.67	31966.67									
			3391667	15666.67	32222.22									
			3561250	16333.33	34183.33									
			3446371	16032.26	33532.26									
			3446371	16032.26	33532.26									
			3502869	16180.33	34426.23									
			3561250	16566.67	35000									
			3502869	16180.33	34770.49									
			3391667	15888.89	33666.67									
			4969186	20023.26	47046.51									
			3502869	16180.33	35803.28									
			3446371	15806.45	35338.71									
			3338672	15531.25	34453.13									
			3446371	16032.26	36016.13									
			3502869	16180.33	36836.07									
			3391667	15666.67	35777.78									

ISAC

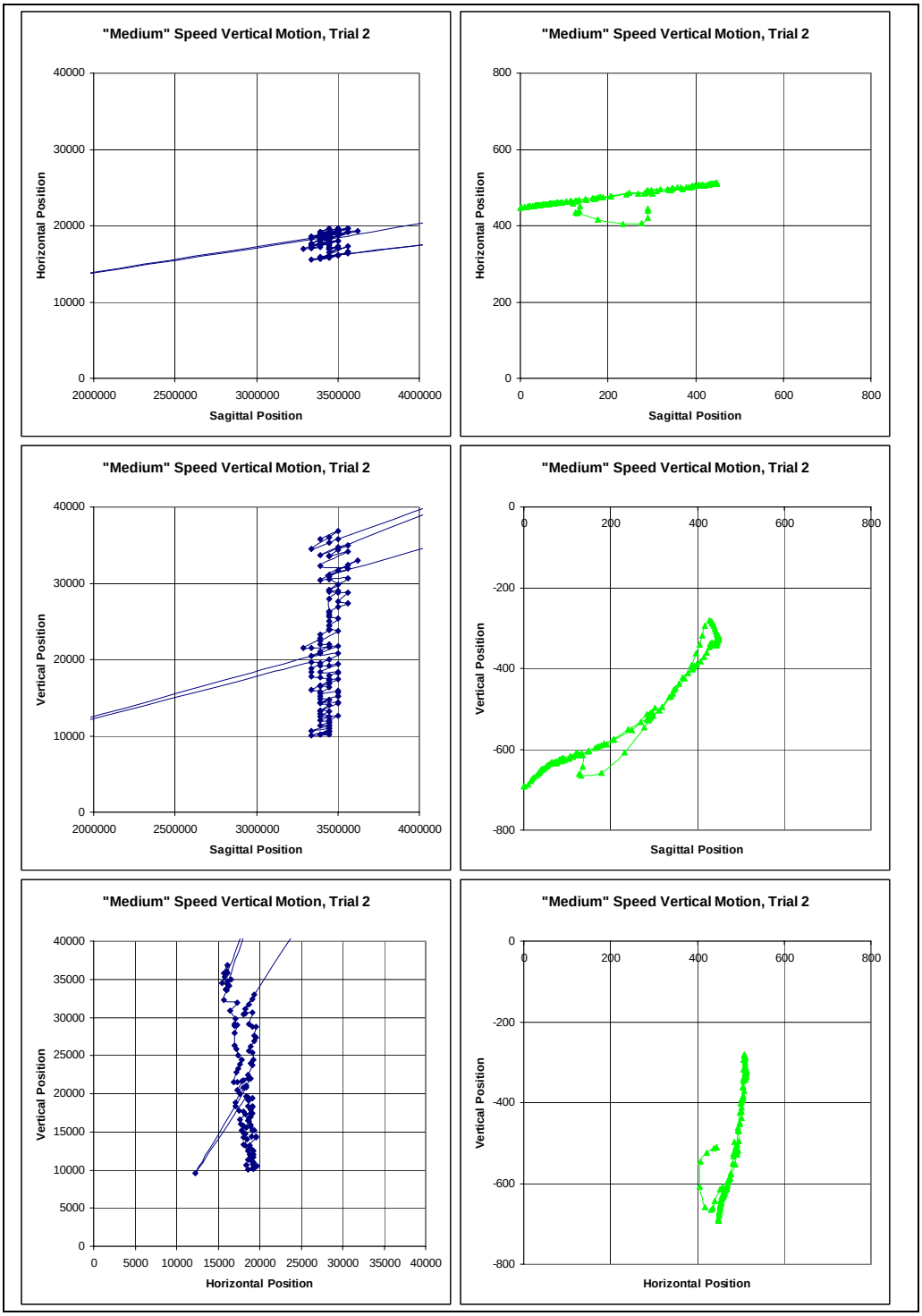
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
293.8132	441.6916	-509.0852	291.9221	443.7118	-511.2739	298.7761	434.6937	-504.1285	293.8542	441.5196	-508.5955	294.0161	441.8603	-509.0385
293.8893	441.5762	-509.051	291.7958	443.7128	-511.3879	298.5289	434.9895	-504.1857	293.9552	441.4608	-508.5384	293.9656	441.7466	-509.0503
293.3914	442.0405	-509.1199	291.4224	439.014	-513.0579	296.3803	435.5816	-505.2253	293.7872	441.8064	-508.4812	292.8295	441.7631	-509.7361
275.5435	442.497	-516.9628	290.3235	422.0201	-523.4155	266.0721	434.9825	-524.9654	288.306	442.7835	-511.5134	265.7715	440.6101	-524.4984
237.1443	442.021	-549.411	277.5143	406.499	-545.615	235.7934	440.0864	-558.5276	256.1482	441.9483	-533.7189	231.8112	443.9581	-562.0526
186.4597	450.6032	-591.2241	233.2769	405.8383	-608.0813	191.4618	451.0342	-583.9327	206.4978	446.4468	-580.9712	183.8742	448.5481	-619.6317
144.8711	457.7118	-620.426	178.0247	417.2678	-657.5713	160.0609	458.4974	-599.3241	140.2929	454.5973	-628.9102	136.986	449.929	-666.96
121.5806	458.5887	-629.6913	130.7779	432.5526	-666.258	148.9844	459.1368	-603.531	99.01284	459.1495	-642.3024	105.572	449.2432	-678.1155
125.732	458.8723	-624.135	126.5094	435.9151	-659.7031	146.3975	458.7918	-603.9346	99.661	460.5403	-634.4578	111.5192	448.777	-667.1774
129.1051	457.1026	-618.4428	134.9025	440.7256	-643.6767	144.47	458.4415	-604.7641	114.8432	459.8501	-625.7013	122.875	447.4626	-656.9305
123.3333	457.8855	-617.4801	136.6206	451.7538	-614.4537	138.8523	459.2176	-605.6948	118.5088	459.3096	-620.1241	135.9585	445.9466	-628.8767
110.4705	458.1156	-616.8104	120.9819	459.5932	-607.2911	130.9107	461.1943	-607.2501	120.1872	461.2901	-606.8348	130.3837	447.8385	-620.5498
96.75404	458.4515	-620.1679	91.90818	459.7974	-627.2054	127.0197	463.2252	-607.4004	119.2927	461.4133	-606.8416	115.8949	448.5284	-627.0983
92.70968	458.8202	-621.8529	69.79683	457.3347	-633.2851	126.9035	463.7562	-607.3186	116.4957	461.0843	-611.5846	100.648	448.9913	-635.3393
91.69468	460.0051	-621.0981	68.01886	458.4253	-632.7152	134.0781	465.403	-605.2714	117.2099	461.1683	-611.2383	91.84628	452.014	-636.67
93.53114	460.1303	-620.5897	71.00369	458.0909	-633.8801	148.0077	467.3784	-600.8309	118.283	461.1817	-610.9314	87.31909	455.3635	-634.4072
98.60332	461.052	-619.0475	76.3113	458.3241	-632.7118	161.3023	468.2068	-594.4381	117.9393	461.3386	-612.4215	84.59491	456.1827	-634.9946
107.667	462.3509	-616.8824	75.94887	459.2122	-633.3712	172.1324	469.8651	-589.412	134.5431	466.0763	-610.1572	86.15217	458.2393	-634.0165
118.6691	463.8363	-612.4336	76.67386	459.4454	-632.1096	174.6294	470.1873	-588.3404	150.8368	468.4638	-600.6465	94.18963	460.0972	-630.5079
133.0743	465.6533	-607.4866	85.84863	460.5659	-629.0947	177.3506	473.0708	-586.2513	165.1296	469.8022	-594.3178	104.6657	461.3955	-624.7869
143.5935	467.0651	-603.583	94.74703	461.7591	-625.4041	186.5618	476.9721	-579.9974	181.0536	472.8837	-586.2793	114.4218	462.6437	-620.105
163.5374	469.8179	-597.1402	103.8873	462.814	-622.1984	198.8356	478.6254	-573.9268	195.5595	474.8857	-578.694	124.694	464.1058	-616.6253
181.0356	471.6754	-587.4234	113.445	464.3457	-618.5475	216.0067	481.2433	-563.7872	212.6578	477.2455	-569.7851	134.2039	465.2595	-613.0818
204.9423	473.5462	-577.0097	127.0862	466.3986	-613.4573	228.1198	481.8986	-555.7959	235.3929	480.278	-556.1677	143.903	467.6536	-608.4907
225.7475	472.1571	-566.1227	147.7445	469.378	-605.0567	241.3711	484.783	-546.2041	253.3259	482.7459	-543.4194	160.0813	471.5361	-600.5444
244.9328	470.9523	-557.9916	165.1095	471.4171	-596.4394	254.5796	486.838	-535.9443	273.8365	485.5684	-529.1254	175.8773	473.846	-592.3463
268.9372	472.7322	-543.4279	184.2137	474.5098	-586.4339	270.4473	488.397	-525.3755	286.5969	486.8221	-518.3373	193.0227	476.1307	-582.882
286.0944	472.8691	-528.8151	205.1299	477.5608	-576.5034	287.4754	489.6577	-511.7379	306.8541	490.0452	-503.1531	208.7905	478.4206	-573.3843
305.4375	472.3924	-514.7963	249.0268	487.5714	-552.3242	299.9712	491.2013	-503.2049	323.9043	492.3881	-484.8073	220.7449	480.1567	-565.5511
320.6276	471.14	-500.6507	284.0722	486.8523	-512.7058	309.9052	492.2231	-500.2617	339.3993	494.5173	-466.0383	236.7193	482.7964	-554.4447
337.3081	470.4007	-488.9115	301.9288	485.1088	-498.2917	316.6133	493.1798	-498.5436	363.0526	497.7676	-446.0088	252.3836	485.0902	-542.3571
360.1202	466.5013	-473.2939	296.3288	488.858	-507.9676	336.5291	496.3129	-489.5034	380.2537	499.479	-428.5766	272.7549	487.9521	-525.7098
377.0113	456.9006	-457.8592	291.1063	492.1121	-520.2907	350.0113	498.3983	-470.6899	394.765	500.8594	-407.2383	286.1661	488.7114	-513.8884
397.86	450.2328	-439.6153	288.1303	491.9194	-527.8858	364.194	500.2807	-450.0735	405.603	504.048	-387.6799	297.9309	490.777	-503.3303
415.2197	439.7404	-420.7933	299.0448	492.7798	-518.2219	380.5625	501.5633	-425.1968	417.0228	506.2519	-370.882	309.8878	493.0349	-493.5852
432.1952	426.2429	-400.9935	319.6772	495.5044	-495.5652	391.2219	503.0643	-398.8358	428.7331	507.9807	-354.4111	323.976	495.0781	-482.1946
444.5247	414.4106	-386.1876	335.7698	495.7661	-471.2175	397.4404	503.94	-382.0365	434.6838	507.6106	-343.1565	342.2821	497.733	-461.4427
459.7833	404.8868	-375.8858	343.3718	492.8427	-460.8617	403.5333	505.3169	-374.2081	438.1602	507.8089	-336.3169	355.9219	498.4781	-441.5897
476.1419	393.8006	-364.9264	346.5035	497.2175	-452.8131	409.5942	506.5287	-360.4448	438.3636	508.8624	-335.7868	368.7299	500.056	-421.4028
484.589	385.4138	-356.2272	348.9132	498.5432	-448.4014	419.7226	508.6044	-347.1764	438.9014	512.0271	-340.0381	376.4762	501.231	-409.4208
489.1296	380.8516	-347.8956	358.5212	500.7878	-438.44	430.4482	510.0592	-340.2512	442.2173	514.2134	-344.2848	383.7996	503.0099	-403.3399
490.7008	381.5643	-342.4512	365.9159	501.4774	-422.2555	436.2064	510.8885	-341.0535	443.4401	514.776	-345.2962	394.6858	504.9934	-400.1737
491.3124	384.6435	-341.3434	378.5271	501.3581	-412.0439	438.0045	511.1475	-344.7772	445.2714	514.6692	-340.4341	404.4583	506.4029	-395.4392
493.6059	385.1911	-341.8715	388.7269	500.5466	-402.3041	440.3242	512.6109	-348.3351	447.156	514.9505	-333.3396	416.2678	508.1061	-381.9771
495.9866	384.6081	-339.5808	391.5565	501.8521	-396.7522	442.4157	512.9172	-347.4283	448.0121	514.8864	-328.8152	423.9842	509.219	-363.2708
498.1857	384.3288	-334.7125	393.5889	504.3274	-393.3868	444.4819	512.8398	-341.7711	446.3361	514.1594	-326.3657	431.316	509.9998	-342.4941
499.2021	385.4302	-330.3045	398.8174	505.3322	-388.4675	446.4222	513.1223	-335.3033	446.4864	514.1816	-323.5599	435.6698	509.8809	-328.4838
497.6629	393.3714	-327.8607	406.7928	506.4776	-381.6264	446.1422	512.6052	-328.9417	446.4219	513.7901	-322.5828	437.3923	509.939	-323.6332
495.5704	399.9203	-328.3945	414.7747	507.091	-371.1983	446.6987	512.6857	-325.8262	446.2267	513.8568	-322.866	438.691	510.31	-326.701
492.2812	409.4514	-329.4301	421.9145	506.0358	-359.9353	446.8591	512.1368	-324.9745	446.8155	513.0842	-322.3624	442.1575	511.5569	-332.7591
490.8857	414.0758	-329.6847	427.2176	506.6757	-347.0588	447.3278	510.965	-325.414	447.1112	513.0317	-321.3863	445.1024	512.0748	-338.7053
490.4628	416.2715	-328.4296	428.9014	508.6449	-339.1979	447.3293	510.87	-324.3433	446.9748	512.9165	-320.474	447.2784	511.8155	-339.9663
489.1302	420.5913	-327.4556	433.1132	510.0728	-336.2309	446.4758	510.75	-320.7584	447.0084	512.8259	-319.8135	448.8455	511.4654	-335.7761
487.4225	423.9735	-326.569	437.1425	510.4632	-338.1477	443.3174	510.3059	-314.9521	445.4625	512.0316	-317.1856	450.084	511.6405	-330.465
481.851	435.1614	-323.1982	440.9891	509.7906	-341.6691	439.5679	509.7786	-304.409	444.1079	511.837	-311.2319	450.0673	511.6382	-326.8203
475.7339	443.5722	-316.4169	444.1837	509.7659	-343.3716	435.9806	509.2742	-293.967	441.8525	511.5131	-304.9351	446.9866	511.2025	-322.4554
465.0838	457.6703	-305.3902	445.1665	510.5659	-341.1155	433.0803	508.8664	-286.6741	437.8844	510.9432	-297.3971	446.4044	511.5006	-316.9943
454.6944	471.4356	-295.2113	446.1777	511.5635	-336.1633	431.0527	508.5813	-283.2386	435.2503	510.4719	-290.3753	446.0046	511.5388	-313.6785
445.0429	483.3108	-287.9475	446.8815	511.6637	-332.6381	430.7655	508.5409	-283.9733	430.9293	509.6679	-282.7335	445.94	511.5296	-314.2736
440.1997	489.8082	-285.4729	447.2036	511.3284	-329.6395	424.7408	506.6064	-295.2447	428.8352	509.3684	-276.6942	446.1735	511.2776	-314.7421
436.4283	495.5239	-286.1831	447.5942	511.0025	-328.4742	420.8455	506.1577	-314.5117	425.9136	508.9506	-272.2506	446.2768	511.2923	-314.0839
432.1651	496.2274	-291.4971	447.7555	511.0253	-327.908	415.5214	506.3973	-334.8888	419.1936	506.4659	-277.2962	446.1601	511.2757	-312.6465
421.5479	493.2598	-308.0464	447.6564	511.1066	-327.3082	410.85	505.9162	-351.8577	409.5477	502.3231	-302.7625	445.3043	511.1544	-311.5026
413.4466	496.3669	-336.8914	447.4914	511.1786	-326.708	404.8801	504.6409	-367.203	400.3263	501.4361	-342.6894	444.35	511.019	-309.3045
401.9329	497.5516	-364.3976	447.5741	511.1903	-325.7946	395.5474	502.4021	-388.9631	388.9134	500.0729	-385.6806	441.5222	510.618	-303.5918
389.1492	497.6149	-395.1334	447.4858	511.1778	-324.7858	385.3196	501.8193	-408.9952	371.8327	497.7771	-429.6173	435.9114	509.8222	-292.8623
371.9799	497.5577	-425.9243	447.2122	511.1392	-323.8085	371.915	500.3456	-428.7898	358.5871	496.1509	-458.9874	429.4562	508.9068	-277.5443
356.2394	496.6011	-449.4135	447.2505	511.1446	-323.2737	360.2109	498.6196	-445.5417	344.7783	495.2565	-477.8217	425.3089	508.3186	-267.3581
342.4502	495.7513	-467.4151	447.3146	511.1536	-322.802	346.8721	496.3687	-460.7699	329.1155	494.0326	-493.9518	422.3423	507.8979	-264.4773
332.679	495.3988	-477.5342	447.3789	511.1627	-322.4876	338.7058	495.6691	-472.398	318.1449	493.0512	-501.648	416.2877	505.6157	-285.5065
317.3396	493.2142	-487.2768	447.4811	511.1771	-321.9529	340.8132	498.6333	-470.566	298.9517	488.894	-502.6218	415.7218	506.4255	-317.5493
306.1339	493.2975	-497.9493	447.5836	511.1916	-321.5755	352.7097	500.1644	-456.019	289.85	488.9804	-506.0061	406.8152	505.173	-353.6199
291.6695	490.843	-508.7348	447.6338	511.1987	-321.2295	368.7414	501.7299	-431.8461	274.5505	487.1808	-519.4968	398.1568	503.6142	-385.013
270.5521	487.3301	-526.8497	447.3127	511.1534	-320.3182	369.781	491.9009	-424.701	256.338	482.1401	-538.1658	384.2827	501.6743	-416.4997
250.9776	485.672	-542.9028	447.4963	511.1793	-319.0609	369.3179	496.8058	-438.6999	235.1848	480.7754	-553.9185	370.6311	499.7656	-439.7235
231.3628	482.4827	-557.7994	446.8934	511.0942	-318.1218	358.5093	498.5352	-451.0071	216.652	478.8741	-566.5478	353.7529	497.4058	-461.6437
223.1508	482.1797	-563.209	446.9393	511.1006	-316.8671	333.1187	490.9213	-468.93	205.9082	477.6257	-573.4882	331.9052	494.3511	-487.1218
214.4438	479.2904	-566.7888	444.5614	510.7648	-313.4782	293.2098	483.6436	-509.7821	191.8759	474.1141	-581.5151	313.313	491.9566	-501.9093
209.5477	479.105	-569.2384	443.0279	510.5483	-306.9071	244.1151	483.8246	-552.6878	186.0603	473.7457	-586.0111	297.6167	489.8178	-511.7059
193.5848	475.0117	-574.8805	440.8059	510.2345	-299.8193	215.2079	482.4573	-576.0963	168.827	470.6082	-591.1519	281.5078	486.7477	-522.1659
169.2389	472.9151	-587.5059	437.9792	509.8353	-294.0466	208.4807	479.904	-580.7994	156.1304	469.7381	-597.831	261.2369	483.4305	-536.954
152.9344	471.0763	-596.5629	436.1319	509.5744	-290.4239	212.4141	478.7051	-577.5751	141.3966	467.7149	-604.0186	237.334	481.0754	-555.8552
148.776	470.5982	-598.2304	434.8759	509.3971	-288.1959	205.1725	475.9333	-579.247	130.1975	466.1491	-608.874	216.4838	478.1644	-570.7072
143.1525	467.8236	-598.6858	433.4275	509.1925	-285.8056	180.756	474.0552	-590.3635	125.8424	465.571	-609.3431	195.2964	475.2063	-583.5101
131.9601	466.7159	-602.922	432.0597	508.7228	-283.1001	155.8385	471.1384	-602.5252	117.0015	463.8992	-612.1046	183.5565	473.7366	-590.1279
119.0185	465.6206	-609.0072	427.2989	507.9622	-280.2466	140.0698	468.8135	-609.386	107.9707	463.0414	-615.7261	165.178	470.1539	-597.2185
108.3135	463.7734	-613.0151	416.9601	504.9973	-293.1912	136.9327	468.4177	-610.0243	95.97766	461.3894	-620.4794	154.5379	469.4064	-601.5922

ISAC

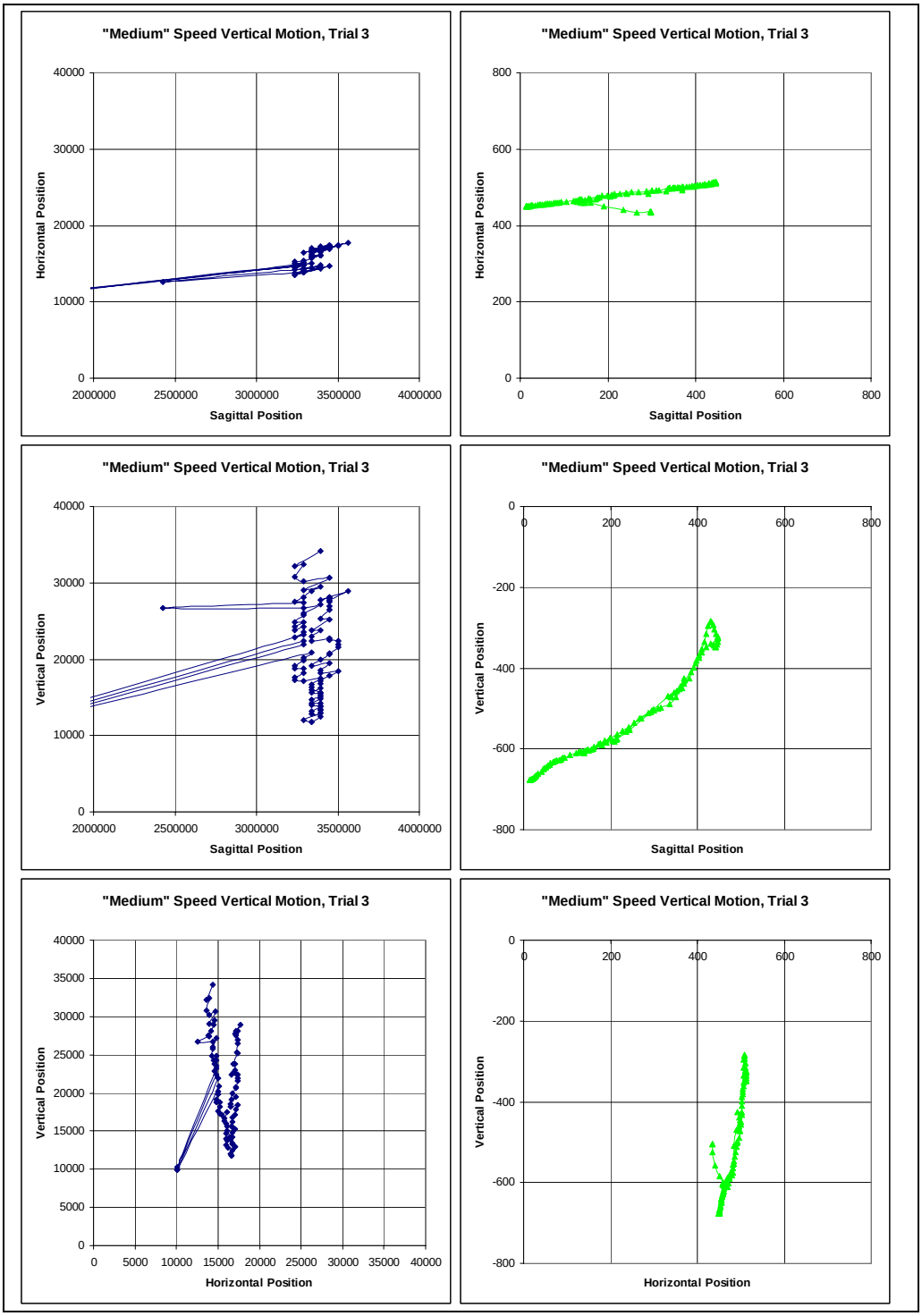
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
102.7076	462.9089	-615.2974	410.7132	505.6333	-317.5195	133.972	466.4504	-608.5841	88.7452	460.4234	-623.3283	137.0572	466.8774	-609.0547
97.56022	462.0896	-617.4428	405.8362	505.0354	-339.4632	130.6821	465.2693	-608.6815	83.60784	459.7021	-625.723	123.4696	465.0269	-614.6878
88.54597	460.6525	-620.8168	397.6285	503.9665	-361.7163	121.4229	464.2081	-610.2718	79.50132	459.1254	-629.4785	112.8186	463.5505	-618.29
81.7508	459.7937	-624.7317	385.6312	501.0355	-390.8558	105.7122	462.0869	-615.397	71.63747	457.9615	-635.0345	103.5318	462.2633	-624.8821
76.95899	459.1055	-627.415	370.6633	496.8257	-423.5414	93.60162	460.911	-620.4688	64.16079	456.9529	-641.2759	95.14395	460.8066	-627.6132
74.10344	458.6953	-629.4204	339.8409	493.697	-468.3779	89.32984	460.5055	-621.8342	58.6858	456.1506	-645.1869	87.00939	460.1107	-629.6928
69.52134	457.8804	-631.4657	312.1699	492.1368	-503.3332	85.75438	460.0035	-624.8517	55.58273	455.7335	-646.8057	80.90233	459.1916	-634.8457
62.63569	456.9025	-638.1021	291.7852	492.4729	-522.6131	82.00398	458.9078	-627.4053	53.55632	455.4497	-647.3703	71.87733	457.5188	-641.3871
56.60079	456.0454	-643.1225	288.298	489.8402	-524.5449	76.62802	458.9317	-628.5443	50.41995	455.0106	-649.4377	63.50324	456.6965	-645.7142
51.68281	455.395	-648.0434	283.7189	483.8908	-525.8826	70.83039	458.0861	-630.5809	47.22207	454.5628	-651.6252	59.19205	456.1345	-647.0589
48.72428	454.9268	-649.2759	269.4475	483.0691	-532.8436	67.58792	457.6058	-632.7986	45.04018	454.2573	-652.8353	58.07478	455.9794	-647.2253
45.53113	454.4733	-652.6901	241.5605	481.073	-550.3856	62.45705	456.8771	-634.8946	41.57573	453.7584	-656.6108	56.83868	455.673	-649.4148
40.44739	453.7513	-659.351	208.1498	477.0009	-575.6527	58.61003	456.3829	-639.6525	31.97799	451.1965	-673.5221	51.54419	452.8968	-660.6831
35.18419	453.0038	-664.4162	190.2934	474.8574	-588.1617	54.83399	455.7612	-642.2533	11.09418	448.0992	-696.9983	44.86926	451.3148	-667.2282
29.14178	452.1456	-667.0225	179.8748	473.8867	-590.6073	50.80963	455.318	-645.2111	-0.120349	446.7191	-705.1134	44.29252	450.613	-666.6978
			172.9077	471.5991	-592.6417	47.34634	454.7916	-647.3215	-0.625943	446.3733	-705.7406	44.37587	450.1856	-667.3049
			168.4299	470.5545	-595.1434	45.46232	454.5226	-650.1761				43.56914	449.9339	-669.116
			150.0378	468.2838	-602.7056	40.5411	453.7783	-656.693				39.8791	450.0261	-673.6749
			134.0997	466.8569	-609.3346	33.76955	452.6669	-661.5146				30.13874	449.3901	-680.6617
			128.1626	466.146	-611.9533	28.99387	452.1132	-665.3592				20.9578	448.696	-684.8525
			126.0637	465.8796	-612.2516	26.18771	451.7152	-668.4223				16.90058	448.642	-687.2464
			118.3341	463.0394	-614.2466	25.82086	451.6632	-668.7361				10.3617	448.3488	-691.9821
			106.5418	462.8417	-617.0975	24.92195	451.5357	-669.358						
			91.35665	460.8618	-621.8237	22.25759	451.1579	-671.2479						
			86.02785	460.156	-623.5734	20.1356	450.8569	-673.3248						
			83.07287	459.7381	-624.3552	18.72488	450.6568	-674.5336						
			79.85493	458.8097	-627.4453	17.8507	450.5329	-675.3503						
			74.72987	458.3938	-629.0819	16.09657	450.2841	-676.4443						
			68.59117	457.5356	-630.9483	14.12511	450.0045	-677.2175						
			64.71383	456.9934	-632.6449	13.45648	449.9097	-677.6279						
			62.73202	456.7163	-634.7081	13.55388	449.9235	-677.4436						
			59.39526	456.1453	-636.4037	14.72757	450.0899	-676.2602						
			56.50875	455.88	-639.444									
			52.54123	455.2113	-641.4125									
			50.70644	455.0036	-643.0849									
			47.32536	454.5322	-646.5987									
			44.81249	454.1819	-648.0949									
			42.22699	453.8214	-648.816									
			41.16211	453.673	-649.6985									
			40.6446	453.6008	-650.3747									
			39.27242	453.4095	-652.5558									
			37.82733	453.208	-654.187									
			36.16198	452.9759	-655.7665									
			34.76572	452.7812	-657.4839									
			32.0975	452.4092	-659.8207									
			28.08114	451.8493	-664.3153									
			22.84732	451.1196	-670.2253									
			18.21534	450.5014	-674.2856									
			16.35527	450.2498	-678.0776									
			9.525356	449.0934	-686.9822									
			-0.171076	447.894	-692.1261									
			-1.712991	447.6851	-692.3947									
			-1.378262	447.7317	-692.1061									
			1.703447	448.1077	-691.7244									



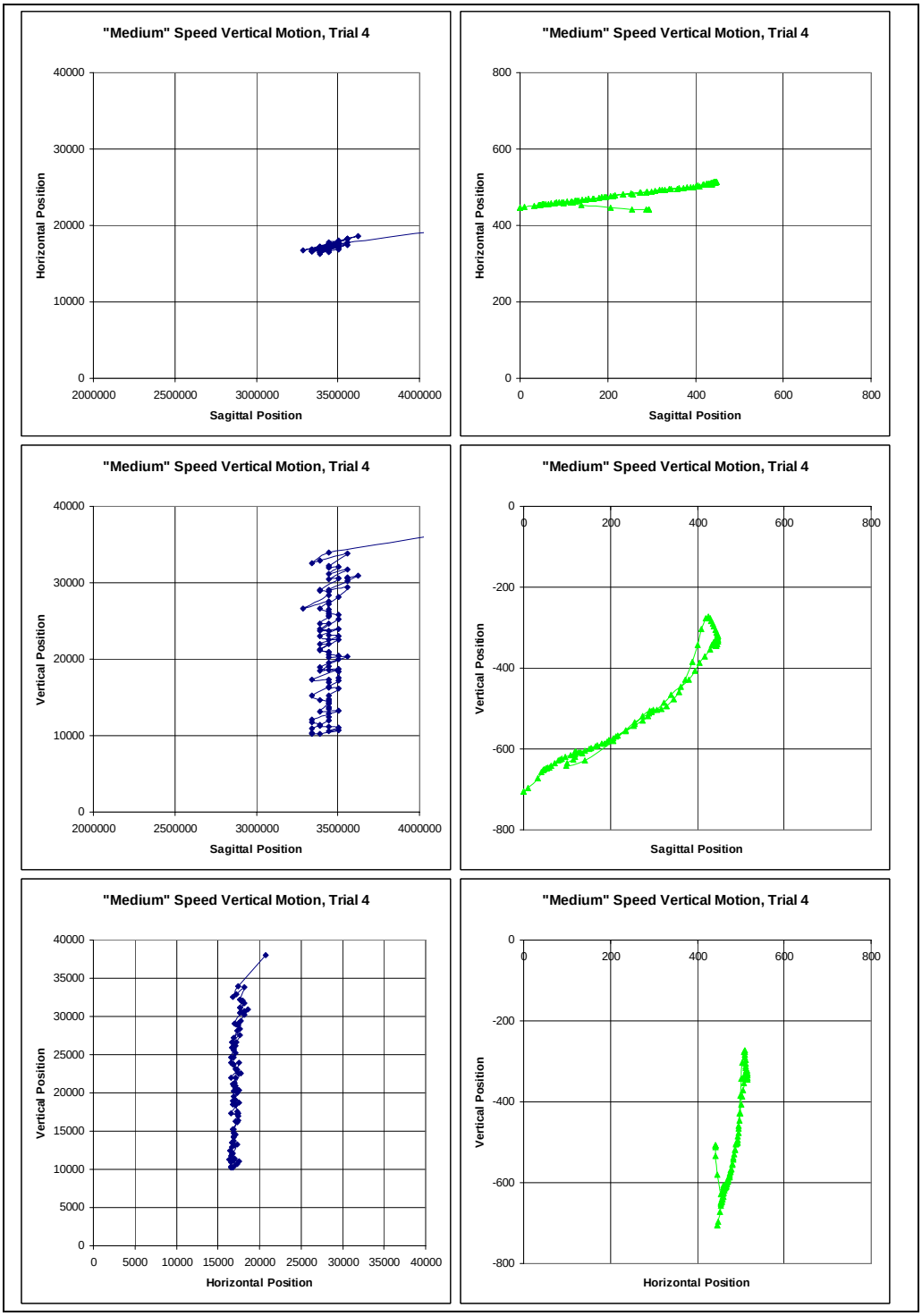
"Medium" speed vertical motion, trial 1; human motion is shown on the left, that of ISAC on the right



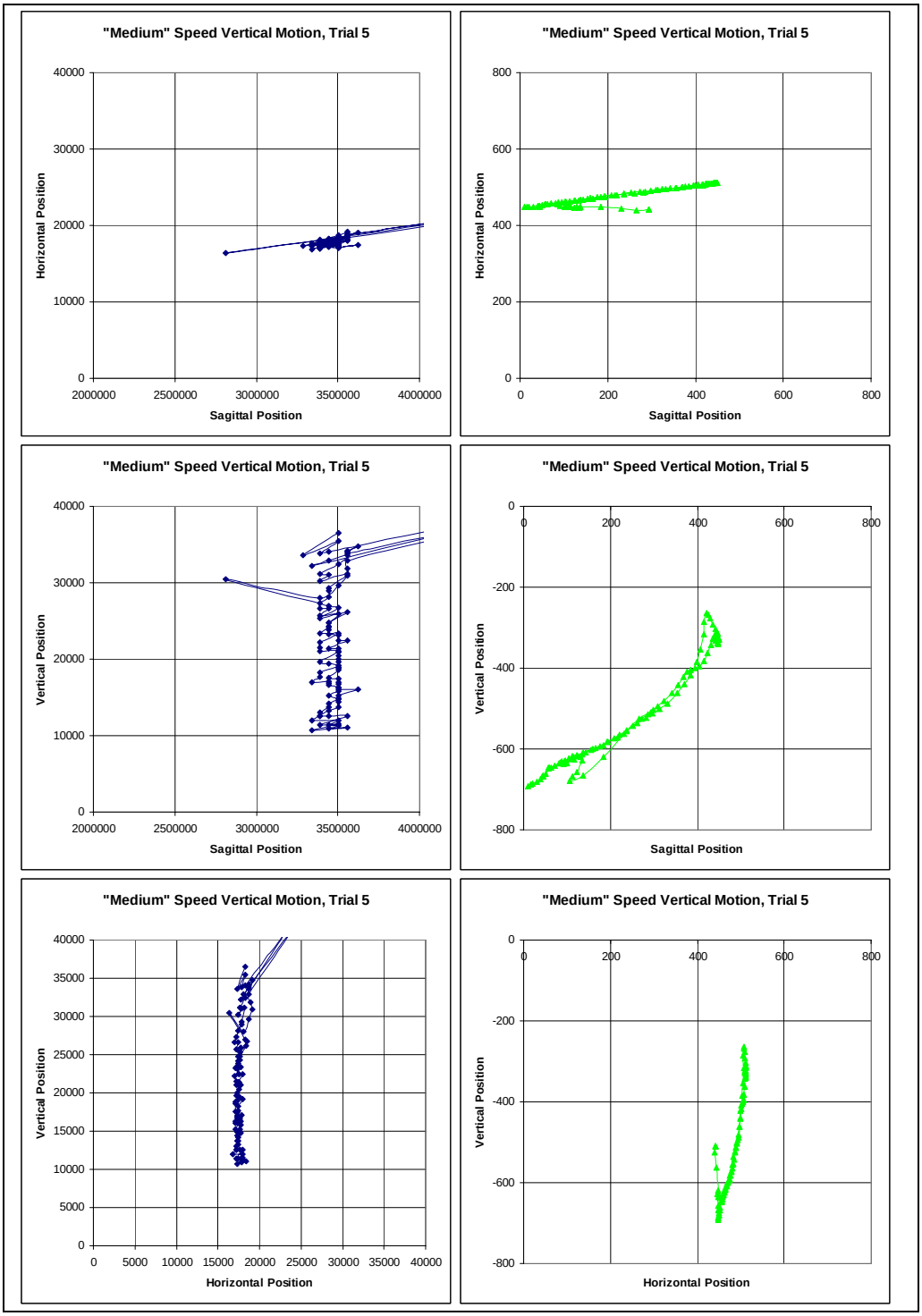
"Medium" speed vertical motion, trial 2; human motion is shown on the left, that of ISAC on the right



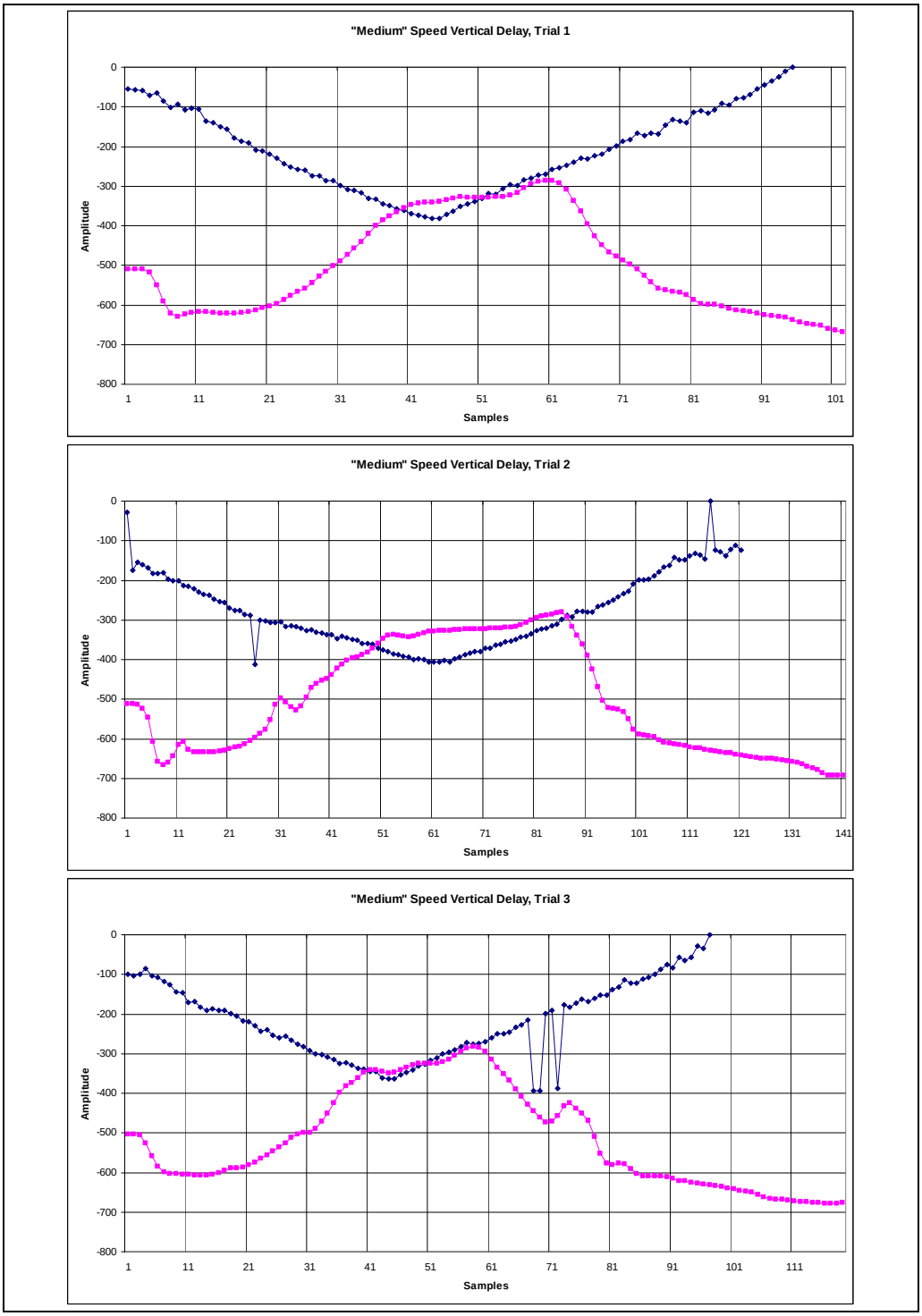
"Medium" speed vertical motion, trial 3; human motion is shown on the left, that of ISAC on the right



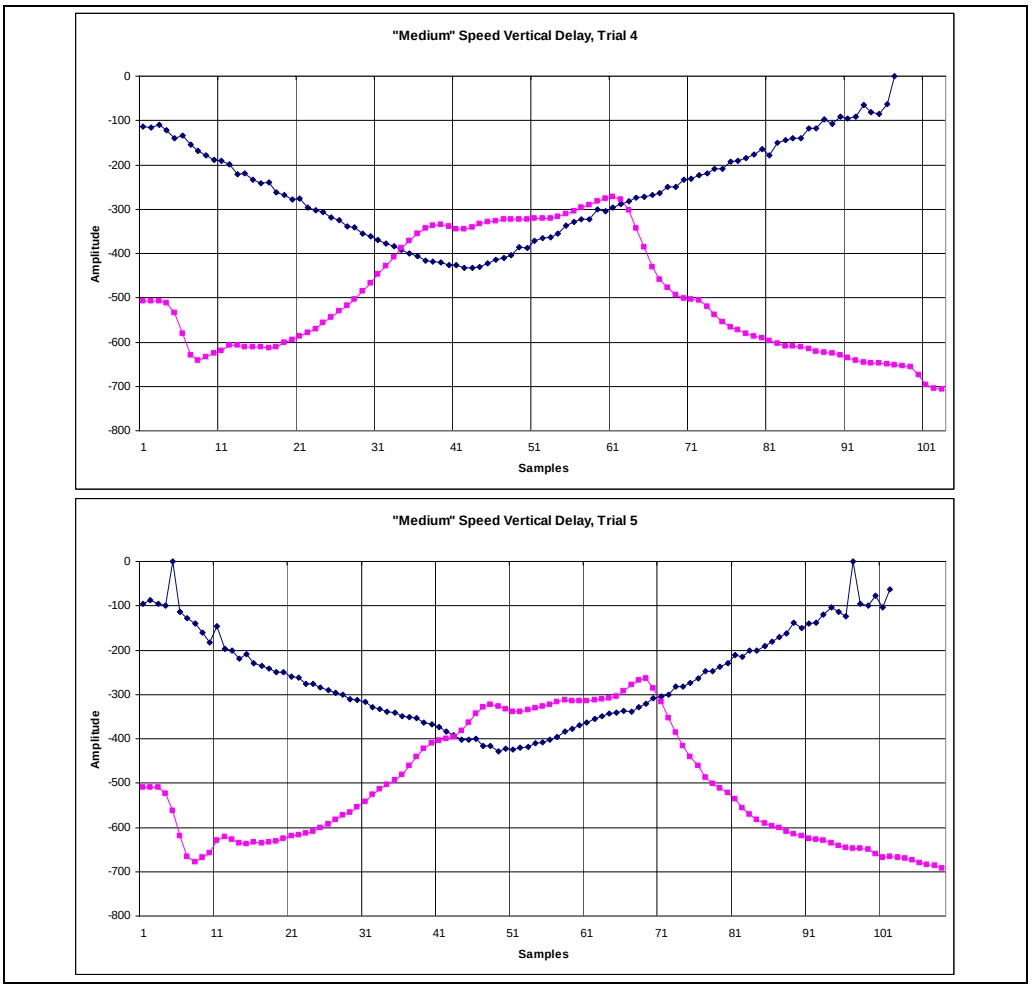
"Medium" speed vertical motion, trial 4; human motion is shown on the left, that of ISAC on the right



"Medium" speed vertical motion, trial 5; human motion is shown on the left, that of ISAC on the right



“Medium” speed vertical delay, trials 1-3; human motion is shown on the on top, that of ISAC on the on bottom



"Medium" speed vertical delay, trials 4-5; human motion is shown on the on top, that of ISAC on the on bottom

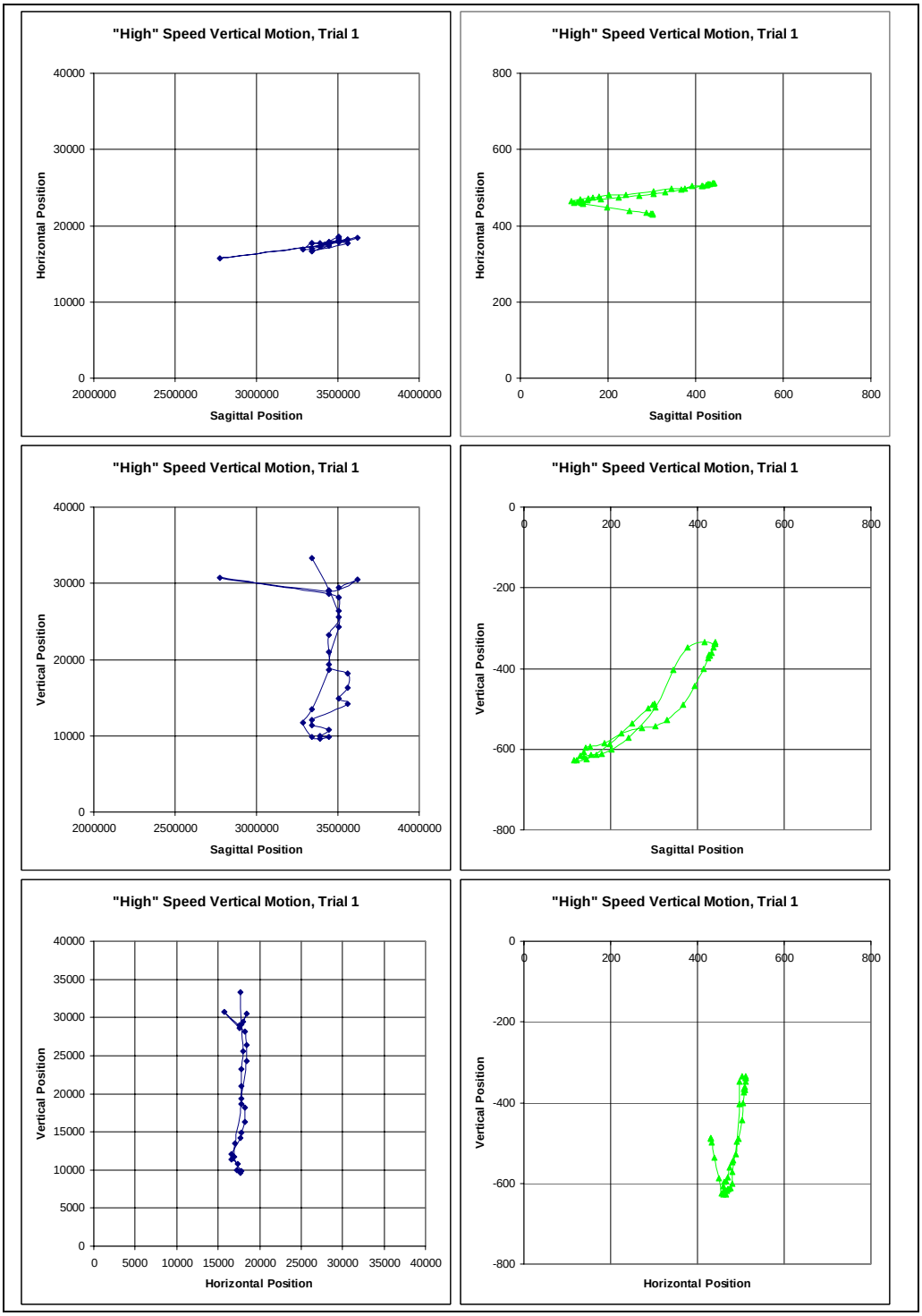
Vertical motion at "High" speed

Human

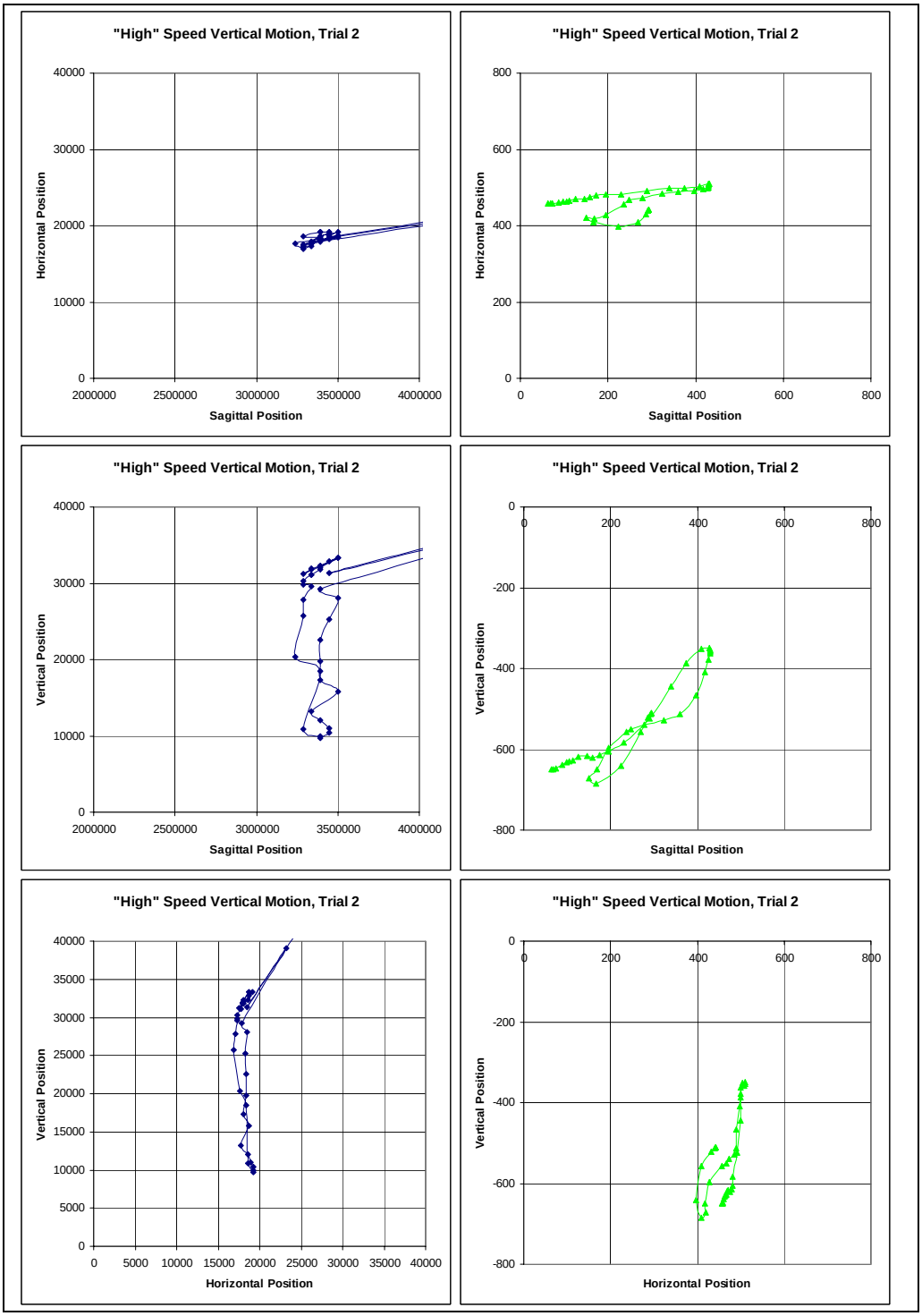
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
3502869	18016.39	29491.8	5211586	24414.63	40975.61	3502869	18245.9	32360.66	3502869	17327.87	33278.69	3561250	18666.67	33133.33
3621610	18389.83	30491.53	3446371	18516.13	31387.1	3502869	18245.9	32475.41	3621610	17915.25	34288.14	3561250	18433.33	33250
3446371	17612.9	29016.13	4969186	23279.07	39069.77	3446371	17838.71	32064.52	3561250	17733.33	33600	3561250	18433.33	33250
2775000	15727.27	30727.27	3391667	17888.89	29222.22	3502869	18016.39	32590.16	3502869	17327.87	33049.18	3561250	18433.33	33133.33
3446371	17612.9	28677.42	3502869	18475.41	28114.75	3502869	18016.39	32819.67	3502869	17327.87	33278.69	3502869	18016.39	32819.67
3502869	18245.9	28114.75	3446371	18290.32	25290.32	3502869	18016.39	32819.67	3446371	17161.29	32854.84	3561250	18433.33	33250
3502869	18475.41	26393.44	3391667	18333.33	22555.55	3391667	17222.22	31777.78	3446371	16935.48	32629.03	3446371	17838.71	32290.32
3502869	18475.41	24327.87	3391667	18333.33	19777.78	3446371	17387.1	31161.29	3338672	16406.25	30843.75	3502869	18245.9	32819.67
3446371	17838.71	19306.45	3391667	18111.11	17333.33	3446371	17161.29	27435.48	3446371	16483.87	17838.71	1405757	11513.16	11743.42
3561250	18200	18200	3502869	18704.92	15836.07	3446371	17387.1	22806.45	3502869	17098.36	15147.54	1405757	11513.16	11789.47
3561250	18200	16333.33	3338672	17718.75	13234.38	3561250	19600	8983.333	3561250	17500	13300	3446371	17838.71	32516.13
3502869	17786.88	14918.03	3391667	18555.55	12000	3561250	19833.33	8283.333	3502869	18245.9	10327.87	3391667	17222.22	31111.11
3561250	17733.33	14233.33	3446371	18967.74	10951.61	3446371	19193.55	7903.226	3621610	18864.41	9847.458	3446371	17161.29	29129.03
3338672	16625	12031.25	3446371	19193.55	10387.1	3561250	20300	8283.333	3561250	18900	8750	3446371	16709.68	13548.39
3338672	16625	11375	3391667	19222.22	9666.667	3561250	20300	8633.333	3621610	18627.12	7593.22	3391667	16777.78	11888.89
3446371	17387.1	10838.71	3391667	19222.22	10000	3502869	19622.95	20426.23	3446371	18064.52	18741.94	3502869	17327.87	11475.41
3391667	17222.22	10000	3287308	18630.77	10876.92	3446371	19193.55	23822.58	3338672	17281.25	20671.88	3446371	17387.1	10274.19
3446371	17838.71	9822.581	3391667	18333.33	18444.45	2927055	16972.6	31068.49	3446371	17387.1	26193.55	3338672	16843.75	7437.5
3391667	17666.67	9666.667	3237500	17606.06	20363.64	2927055	16780.82	32890.41	3287308	16476.92	26923.08	3391667	17444.45	7888.889
3338672	17718.75	9843.75	3287308	16907.69	25738.46	3391667	18111.11	30888.89	3338672	16625	29093.75	3561250	18433.33	8866.667
3287308	16907.69	11738.46	3287308	17123.08	27892.31	3561250	18666.67	33016.67	3391667	17000	31111.11	3502869	18475.41	9639.345
3338672	17062.5	13453.13	3338672	17281.25	29640.63	3561250	18900	33250	3446371	17387.1	32290.32	3391667	18111.11	21777.78
3446371	17838.71	18629.03	3287308	17338.46	29830.77	3391667	17666.67	31666.67	3446371	17387.1	33532.26	2739423	15794.87	27102.56
3446371	17838.71	21000	3287308	17338.46	30261.54	4856250	22272.73	38659.09	3446371	17387.1	34548.39	3391667	17666.67	25888.89
3446371	17838.71	23258.06	3338672	17718.75	31062.5	3502869	18245.9	32475.41	3446371	17612.9	35112.9	3391667	17666.67	27666.67
3502869	18016.39	25590.16	3338672	17718.75	31062.5	3446371	18064.52	32290.32	3391667	17444.45	35333.33	3446371	17387.1	29693.55
3446371	17838.71	29129.03	3391667	18111.11	31777.78	3446371	17838.71	32290.32	3561250	18666.67	38266.67	3391667	17000	30555.55
3338672	17718.75	33359.38	3391667	18111.11	32000				3446371	18290.32	37483.87	3446371	16935.48	32403.23
			3502869	18704.92	33278.69				3502869	18245.9	38327.87	3446371	17387.1	33419.36
			3391667	18111.11	32222.22							3502869	18016.39	34655.74
			3287308	17553.85	31230.77							3561250	17966.67	36866.67
			3338672	17937.5	31828.13							3446371	17161.29	36580.64
			3338672	17937.5	31937.5							3446371	16935.48	37370.97
			3391667	18555.55	32222.22							3338672	16406.25	36531.25
			3502869	19163.93	33278.69							3446371	17161.29	37822.58
			3446371	18741.94	32854.84							3502869	17327.87	38442.62
												3502869	17327.87	38557.38
												3446371	16935.48	38274.2

ISAC

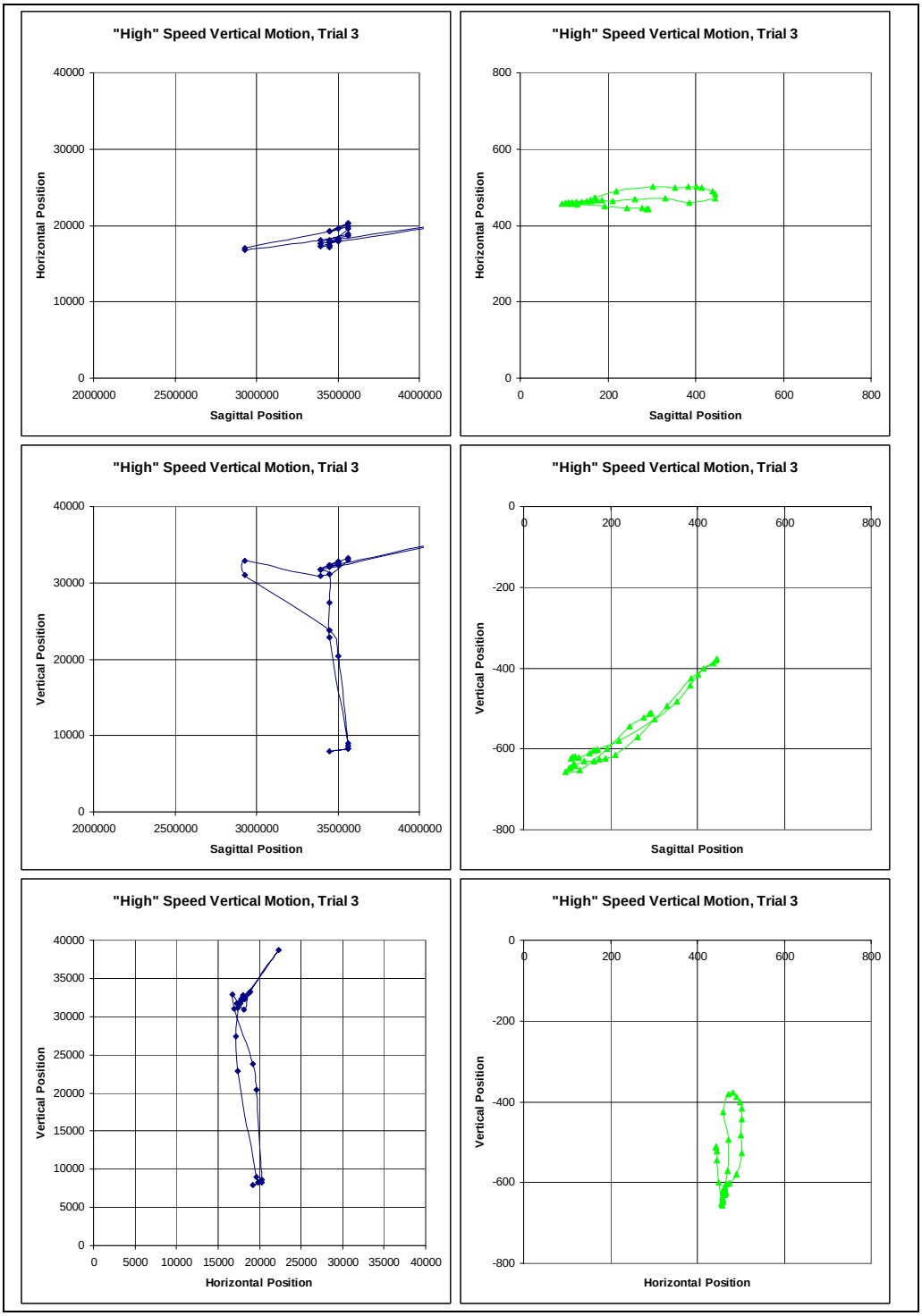
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
301.5735	430.4973	-487.0343	293.3151	441.9273	-511.4786	293.1862	442.6716	-510.6915	291.2748	444.9711	-513.2219	291.9433	444.5112	-510.6573
301.3414	430.9681	-486.977	293.4414	441.9255	-511.3644	293.1222	442.6724	-510.6803	291.3381	444.9709	-513.1306	292.0603	444.339	-510.6573
297.9313	431.822	-489.3165	291.4132	440.2555	-512.7689	288.8659	443.0026	-512.7743	288.5339	444.9808	-514.4347	291.0478	444.3448	-511.3311
287.2955	433.8152	-498.657	286.5095	430.9047	-521.9694	277.7238	444.6382	-522.5057	271.8217	446.1068	-526.0161	280.7013	445.2286	-517.8151
249.4404	438.5894	-535.2854	268.9442	409.5218	-555.9821	243.3912	444.9498	-544.9263	236.872	444.1907	-552.6198	254.6588	445.8999	-537.2486
198.7909	449.0704	-586.5255	223.3692	397.2929	-639.8581	193.293	449.682	-599.9392	188.3704	450.7258	-608.5457	202.1782	447.5952	-592.0442
143.7292	456.4671	-624.7538	167.3728	409.5652	-684.6244	128.9759	455.7529	-651.8849	136.0872	454.3499	-656.4199	137.5519	452.8587	-652.2573
121.4494	460.7486	-627.1407	150.6717	420.0688	-671.2462	95.6419	457.6897	-657.157	99.85558	455.3811	-666.5273	80.10761	455.1224	-675.1649
129.6856	461.5443	-617.1652	169.5631	418.1604	-650.1583	103.7772	459.0832	-648.8791	96.20794	458.531	-657.2842	71.51878	458.244	-666.5312
138.7355	460.0476	-607.067	195.784	427.2187	-596.3606	115.8255	458.0363	-636.2929	103.508	458.9273	-650.7871	86.67956	459.1103	-658.8361
142.1575	461.9471	-596.085	235.8135	456.5794	-556.4166	119.3653	459.2592	-619.0869	112.5009	457.7694	-631.7465	102.6575	459.0059	-642.115
153.0498	465.9834	-593.8371	247.52	467.2603	-549.5479	112.6097	460.0334	-619.5202	111.5316	459.2207	-619.0442	142.971	469.7955	-629.9764
184.5617	469.6727	-585.3097	278.1367	473.4384	-539.3623	109.7118	459.7578	-623.2793	97.05528	457.6559	-624.3642	208.4389	477.8467	-604.5491
224.0911	473.7422	-560.1544	322.9154	484.9546	-528.3686	127.1681	462.4899	-620.9232	82.88081	456.3525	-634.4667	247.4099	470.4547	-569.8836
271.3546	479.7137	-548.2312	360.0029	488.9683	-511.7824	151.7664	464.3644	-610.3995	79.33677	457.1031	-635.2542	239.7028	472.5564	-569.9245
303.7668	482.5946	-542.4567	397.076	490.0859	-465.7953	163.109	465.1633	-604.0565	94.80528	462.0652	-639.2862	204.3358	472.9204	-603.4559
330.4619	488.8008	-527.0273	417.2132	496.813	-409.4952	160.4711	466.9921	-604.4091	149.4476	477.3812	-652.1191	148.5827	464.0397	-632.168
367.2409	494.7231	-488.9616	426.5138	498.7221	-378.9035	169.9226	473.0551	-602.0792	221.8914	488.3007	-630.1125	105.1208	460.855	-646.6014
392.8515	503.2938	-442.274	429.1221	500.2106	-362.3995	220.178	490.8568	-580.2862	304.8373	499.0531	-564.7967	86.08624	462.9502	-649.5626
415.1358	505.0131	-401.137	429.4531	504.1548	-356.6127	302.0006	501.9768	-527.5011	368.0051	503.8568	-489.3867	115.0676	467.3556	-647.2877
425.061	506.4695	-373.6731	429.9279	507.4149	-357.4999	352.849	499.8017	-481.9681	399.1972	507.7929	-425.6601	207.6915	481.3186	-619.4022
427.475	507.8957	-365.0255	430.7568	509.5512	-354.4819	384.4115	502.0227	-443.9357	408.2514	505.375	-373.3178	311.5659	491.9153	-545.8168
427.643	508.741	-368.1323	428.6505	509.1588	-349.6764	401.4653	502.617	-415.5747	416.2733	506.8591	-342.0624	379.6401	492.2022	-464.6663
429.2367	509.2424	-368.0349	409.6112	502.8549	-350.8803	413.637	498.9204	-400.234	421.7699	508.2677	-329.7326	413.592	495.4268	-403.7527
432.7225	509.9244	-360.7005	374.3484	498.9983	-385.7656	437.5778	489.9155	-388.0278	422.5655	508.6526	-329.5689	420.6446	505.9505	-370.0754
438.0595	510.8748	-347.6624	339.9424	499.3207	-444.9017	444.8448	482.2003	-377.054	427.7385	509.5774	-339.0123	422.9198	508.3415	-362.7415
441.1345	511.3158	-338.1259	289.2237	490.7403	-522.8208	443.7017	471.1866	-382.3494	445.85	512.9438	-357.2522	424.8892	509.8044	-367.703
441.9002	511.4256	-335.2617	230.9806	482.0083	-583.9888	385.5115	460.0744	-425.6373	446.2463	514.2416	-375.9061	427.8037	509.2209	-367.4549
439.9822	510.7749	-339.2976	194.6299	481.1736	-606.1828	330.3144	471.1311	-493.0449	444.57	513.9934	-383.5627	432.0589	508.9071	-358.5254
417.1307	503.6881	-335.1837	173.8618	478.8033	-614.2977	262.058	469.7389	-570.587	446.9104	514.1487	-375.7567	439.9543	509.6455	-343.3811
376.3021	496.7682	-347.5274	159.2876	474.166	-620.1538	210.8411	463.9872	-615.1638	440.3177	512.9868	-356.398	445.8573	509.4302	-330.7363
344.7772	496.8976	-403.682	145.6379	471.2237	-617.2938	187.7878	465.9719	-624.7392	414.2735	504.4526	-320.9683	447.4649	509.462	-325.9437
303.0664	491.2478	-496.4788	125.753	469.5092	-618.2596	174.5903	466.2709	-625.5477	380.3921	498.2766	-310.5876	447.3076	509.7254	-326.4157
240.9447	480.7201	-571.2179	112.9443	466.5432	-626.788	161.2756	464.4887	-630.082	347.7686	495.2171	-345.9763	443.6032	509.6851	-329.63
202.3953	481.4245	-599.6379	105.5007	463.852	-629.4864	140.1408	461.9603	-629.3777	313.9599	491.979	-462.0765	403.2078	494.3895	-348.1552
178.6812	476.5633	-612.1485	97.88931	462.569	-631.7254	118.6119	459.5416	-641.0323	254.6105	480.1703	-562.1334	351.2441	488.2535	-408.8333
166.0655	472.8816	-613.7119	87.92638	461.2198	-638.937	108.849	457.5191	-646.7837	195.0209	474.189	-635.6518	303.2187	493.9368	-495.4989
155.3207	470.6183	-614.4248	73.31301	459.1153	-647.0439				139.5932	469.0825	-657.8082	226.9496	480.699	-582.138
137.251	468.5651	-618.9323	68.2598	458.4614	-648.8472				123.1898	468.0528	-654.7583	174.734	475.0333	-629.4248
115.6038	465.3809	-627.258	63.09312	457.3164	-650.5661				128.5745	468.1051	-648.85	143.7502	469.1134	-647.2249
									132.7403	466.3762	-630.2312	137.5144	468.1026	-642.8822
									127.9345	465.9263	-622.6939	137.4696	468.1628	-637.3992
									108.1811	463.9467	-636.201	128.7042	465.3468	-635.8129
									81.99789	459.6965	-653.337	111.7335	462.9553	-642.731
									51.62278	455.2108	-692.3199	86.6361	460.5415	-656.4997
									2.430503	447.9712	-733.9905	58.07212	456.3755	-682.7112
									-14.32867	446.0376	-731.7585	35.50887	453.1394	-701.2146
												24.19045	451.5161	-704.0519
												25.2753	451.6717	-703.5626



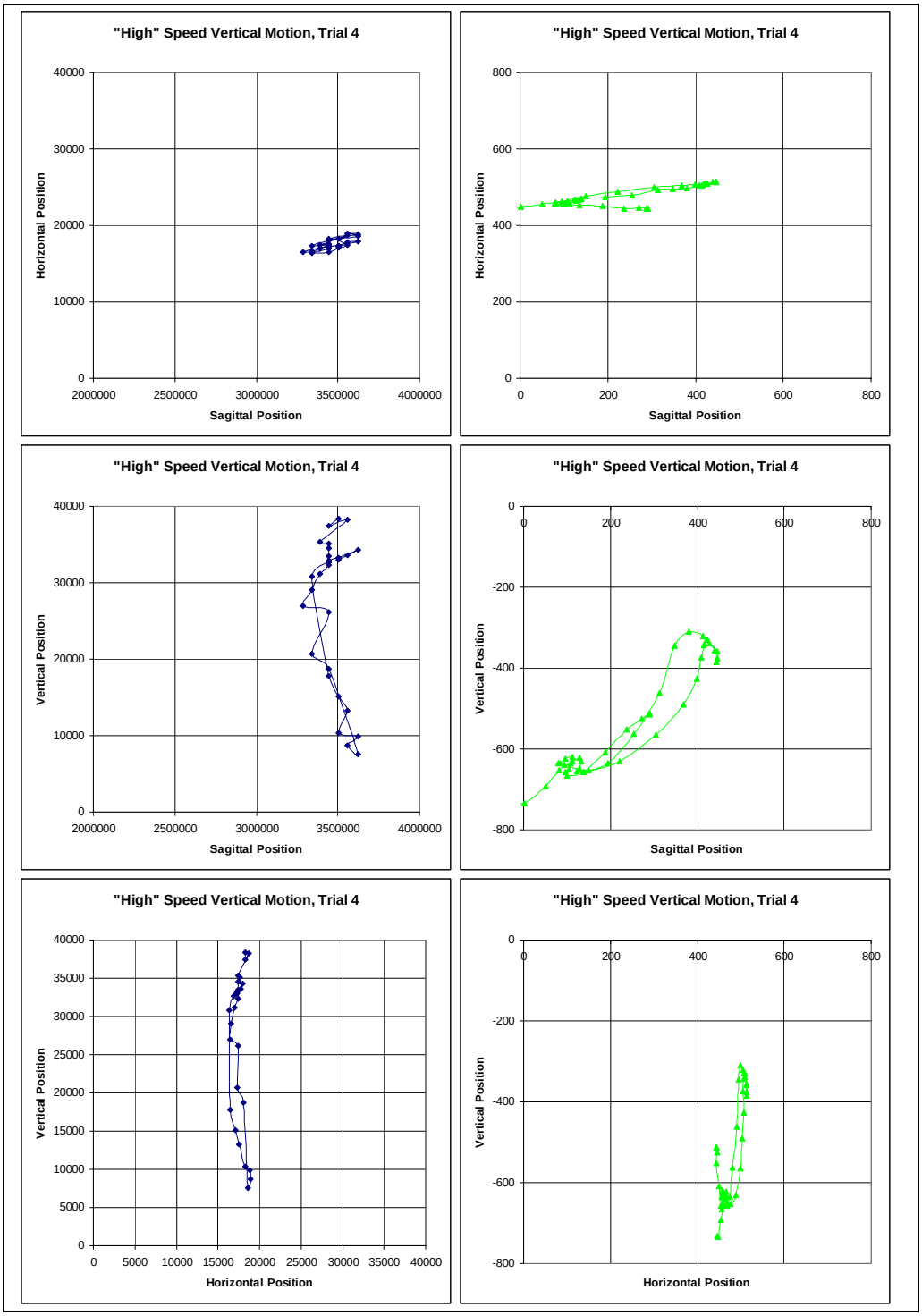
"High" speed vertical motion, trial 1; human motion is shown on the left, that of ISAC on the right



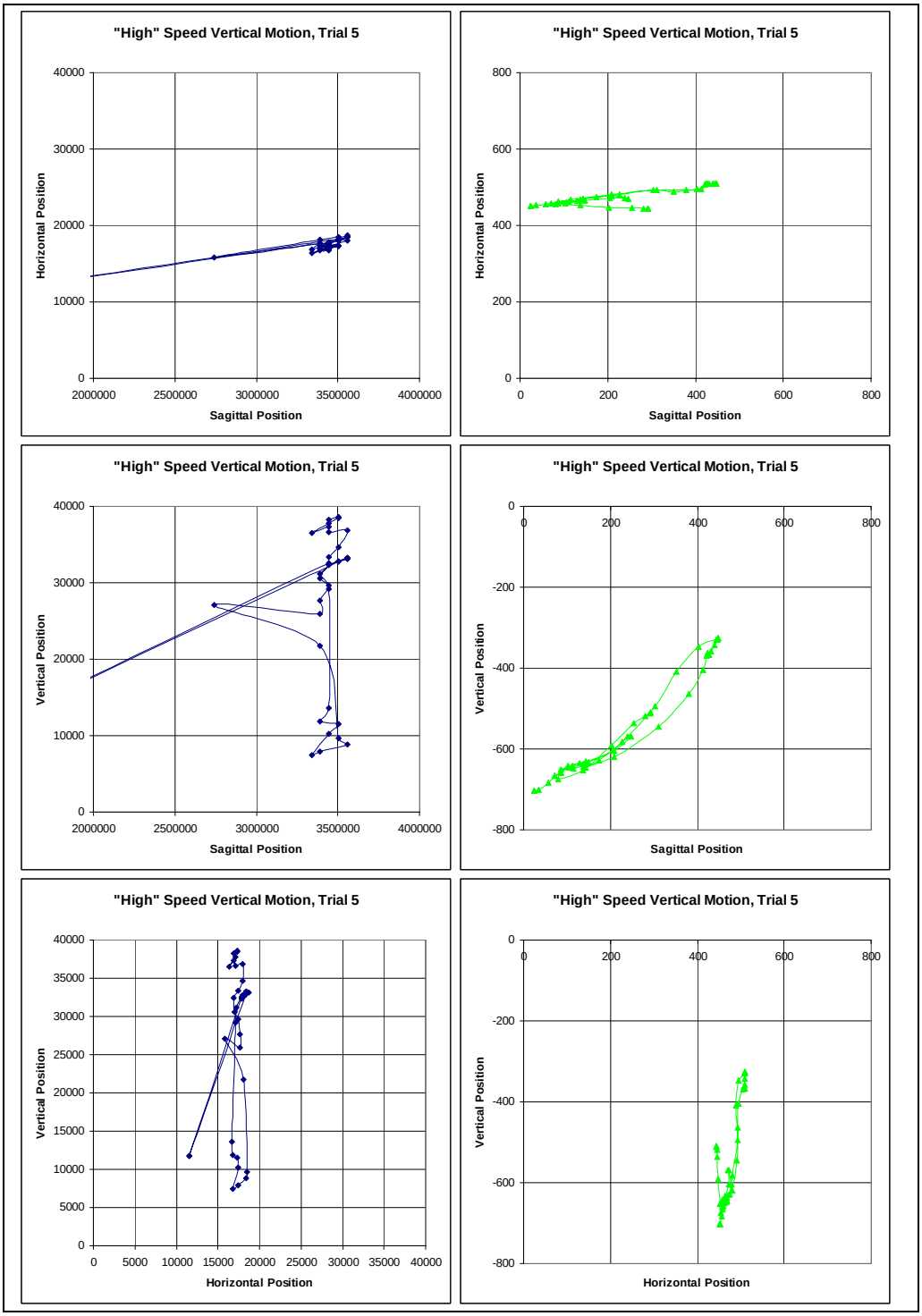
"High" speed vertical motion, trial 2; human motion is shown on the left, that of ISAC on the right



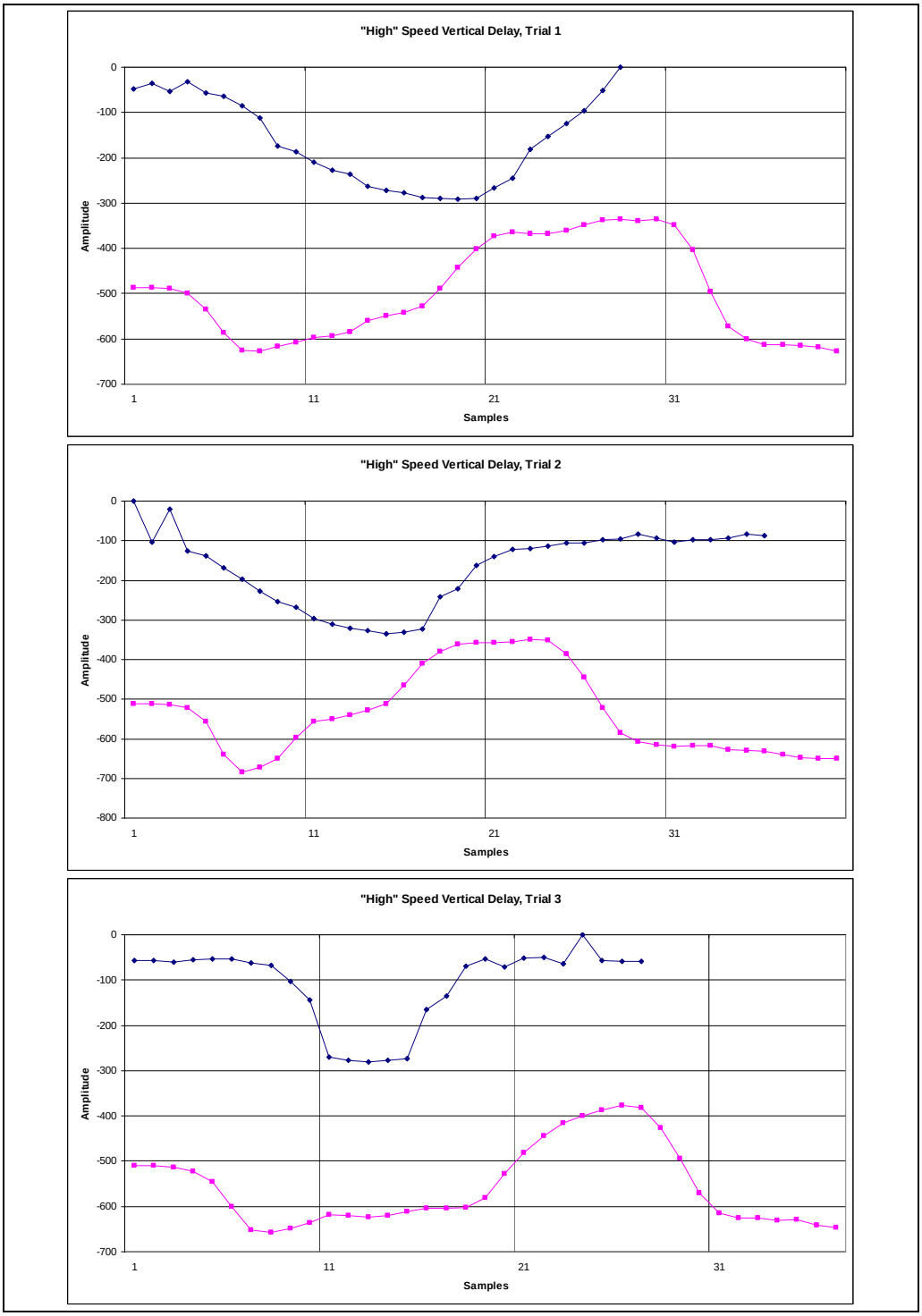
"High" speed vertical motion, trial 3; human motion is shown on the left, that of ISAC on the right



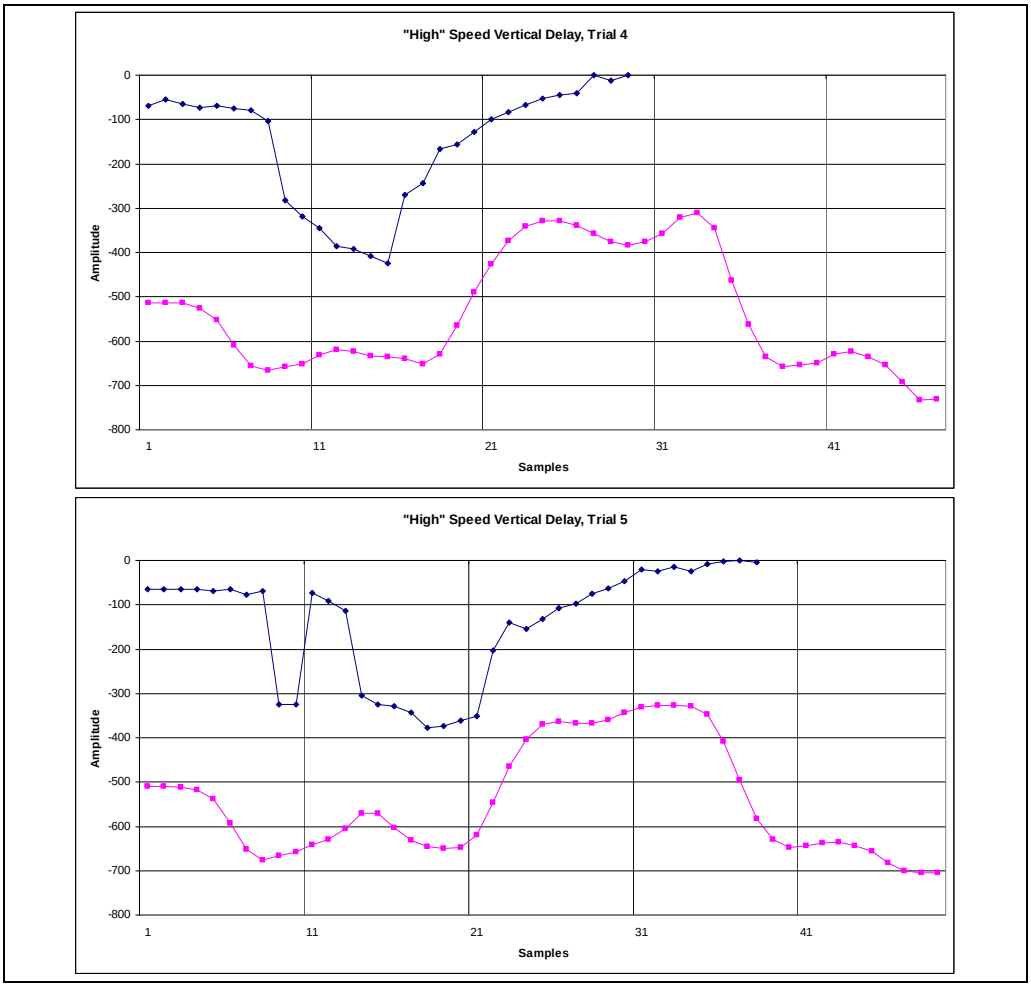
"High" speed vertical motion, trial 4; human motion is shown on the left, that of ISAC on the right



"High" speed vertical motion, trial 5; human motion is shown on the left, that of ISAC on the right



"High" speed vertical delay, trials 1-3; human motion is shown on the on top, that of ISAC on the on bottom



"High" speed vertical delay, trials 4-5; human motion is shown on the on top, that of ISAC on the on bottom

Horizontal motion at "Low" speed

Human

Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
3446371	17838.71	10500	3561250	15166.67	13183.33	3561250	12133.33	14233.33	3502869	14803.28	12278.69	3621610	15067.8	17677.97
3287308	16692.31	10015.38	3561250	14933.33	13066.67	3502869	12049.18	14000	3561250	14933.33	12600	3684052	15206.9	17982.76
3391667	17444.45	10444.44	1559672	10474.45	8226.277	3621610	12457.63	14474.58	3561250	14933.33	12600	3815625	15750	18625
3391667	17444.45	10555.56	3561250	14933.33	13183.33	3502869	12278.69	13885.25	3561250	14933.33	12600	3684052	15206.9	17862.07
3446371	17612.9	10500	3561250	14700	13066.67	3502869	12278.69	13770.49	3561250	14933.33	12483.33	3621610	15067.8	17677.97
3446371	17612.9	10387.1	3561250	14700	13183.33	3561250	12600	13883.33	3561250	15166.67	12483.33	3684052	15206.9	17982.76
3446371	17612.9	10387.1	3621610	15067.8	13288.14	3502869	12737.71	13655.74	3502869	15262.29	12278.69	3748684	15596.49	18175.44
3338672	17281.25	10062.5	3684052	15448.28	13517.24	3502869	12737.71	13885.25	3502869	15262.29	12163.93	3684052	15206.9	17862.07
3391667	17888.89	10333.33	3561250	15166.67	12833.33	3502869	13196.72	13770.49	3502869	15950.82	11934.43	3621610	15305.08	17440.68
3391667	17888.89	10222.22	1473621	10379.31	8158.621	3446371	13322.58	13435.48	3502869	15721.31	11934.43	3684052	15689.66	17862.07
3338672	18156.25	9953.125	1494231	10622.38	8223.776	3502869	13885.25	13655.74	3446371	15580.65	11854.84	3621610	15779.66	17322.03
3391667	18777.78	10222.22	1504754	10647.89	8330.986	3561250	14233.33	13883.33	3446371	16032.26	11629.03	3684052	16172.41	17620.69
3446371	19419.36	10274.19	1494231	10622.38	8223.776	3338672	13562.5	13125	3502869	16409.84	11819.67	3621610	16016.95	17322.03
3391667	19444.45	10222.22	3391667	15222.22	12222.22	3502869	14573.77	13770.49	3446371	16258.06	11741.94	3621610	16016.95	17203.39
3391667	19666.67	10222.22	1537230	11028.78	8460.432	3391667	14111.11	13333.33	3391667	16111.11	11444.44	3684052	16413.79	17379.31
3391667	20111.11	10333.33	3561250	16800	12366.67	3446371	14677.42	13661.29	3391667	16555.55	11444.44	3621610	16491.53	17084.75
3338672	19687.5	10062.5	3561250	17033.33	12600	3338672	14656.25	13234.38	3446371	17161.29	11629.03	3621610	16728.81	17084.75
3391667	20555.55	10111.11	3502869	16868.85	12278.69	3446371	15580.65	13548.39	3391667	17222.22	11444.44	3502869	16639.34	16409.84
3391667	20555.55	10333.33	3502869	16868.85	12393.44	3338672	15093.75	13343.75	3391667	17666.67	11444.44	3621610	17203.39	16966.1
3391667	21222.22	10111.11	3391667	16777.78	12111.11	3391667	15666.67	13444.44	3338672	17937.5	11156.25	3561250	17033.33	16566.67
3338672	21000	9734.375	3391667	16777.78	12111.11	3502869	16409.84	13655.74	3338672	18375	11156.25	3684052	17862.07	17137.93
3287308	20784.62	9800	3338672	16843.75	11812.5	3446371	16709.68	13548.39	3338672	18156.25	10937.5	3502869	17327.87	16524.59
3237500	20575.76	9651.516	3391667	17222.22	11888.89	3338672	16187.5	13015.63	3338672	18593.75	11046.88	3446371	17387.1	16145.16
3287308	21000	9800	3502869	17786.88	12278.69	3338672	16625	13234.38	3391667	19444.45	11222.22	3621610	18627.12	16847.46
3338672	21218.75	9843.75	3338672	17281.25	11484.38	3391667	17222.22	13555.56	3287308	19276.92	10876.92	3391667	17444.45	15888.89
3391667	22111.11	10000	3446371	18064.52	11967.74	3287308	16907.69	13030.77	3287308	19492.31	10769.23	3391667	17444.45	15777.78
3338672	21875	9843.75	3502869	18245.9	12163.93	3338672	17937.5	13234.38	3338672	20125	10937.5	3446371	18064.52	16032.26
3287308	21861.54	9584.615	3391667	18111.11	11888.89	3237500	18030.3	13045.45	3391667	20555.55	11111.11	3502869	18475.41	16295.08
3237500	21636.36	9545.454	3446371	18741.94	11854.84	3189179	18283.58	12641.79	3287308	20138.46	10661.54	3502869	18934.43	16295.08
3142280	21205.88	9264.706	3502869	19163.93	12049.18	3189179	18701.49	12746.27	3287308	20569.23	10553.85	3502869	18934.43	16295.08
3237500	22272.73	9651.516	3391667	18555.55	11666.67	3189179	18910.45	13059.7	3237500	20363.64	10393.94	3391667	18555.55	16000
3142280	21823.53	9161.765	3338672	19031.25	11593.75	3237500	19727.27	13151.52	3287308	20784.62	10769.23	3446371	18967.74	16258.06
3142280	22029.41	9264.706	3391667	19444.45	11777.78	3237500	20363.64	13151.52	3287308	21000	10661.54	3446371	19193.55	16145.16
3189179	22671.64	9507.463	3287308	19276.92	11415.38	3189179	20582.09	12641.79	3237500	21000	10500	3391667	19000	15777.78
3237500	23333.33	9545.454	3391667	20111.11	11888.89	3189179	20582.09	12746.27	3189179	20791.04	10343.28	3446371	19645.16	16032.26
3189179	23089.55	9611.94	3338672	19906.25	11921.88	3142280	20382.35	12455.88	3237500	21424.24	10500	3391667	19666.67	15777.78
3237500	23545.46	9757.576	3338672	20125	11703.13	3189179	21417.91	12641.79	3189179	21208.96	10343.28	3391667	19666.67	15666.67
3189179	23507.46	9611.94	3287308	20138.46	11523.08	3142280	21411.77	12147.06	3189179	21417.91	10343.28	3391667	20111.11	15777.78
3189179	23716.42	9716.418	3446371	21000	12080.65	3189179	22253.73	12328.36	3142280	21205.88	10191.18	3391667	20111.11	15666.67
3189179	23716.42	9611.94	3287308	20353.85	11415.38	3052500	21400	11800	3189179	21626.87	10343.28	3391667	20333.33	15555.56
3142280	23470.59	9779.412	3391667	21222.22	11777.78	3096739	22217.39	11768.12	3237500	22060.61	10500	3338672	19906.25	15312.5
3189179	23925.37	9716.418	3287308	20784.62	11415.38	3142280	22647.06	12044.12	3189179	21835.82	10238.81	3287308	19923.08	14969.23
3142280	23882.35	9470.588	3391667	21666.67	11888.89	3009507	21788.73	11633.8	3189179	22462.69	10343.28	3502869	21229.51	16065.57
3237500	24818.18	9969.697	3338672	21437.5	11812.5	3052500	22200	11700	2115594	17396.04	11366.34	3287308	20353.85	14861.54
3052500	23600	9200	3338672	21656.25	11703.13	3096739	22826.09	12173.91	3237500	23121.21	10287.88	3287308	20569.23	15076.92
3189179	24970.15	9716.418	3338672	21656.25	11593.75	3052500	22800	11800	2348077	20076.92	12538.46	3287308	20569.23	14969.23
3142280	24911.77	9573.529	3189179	20791.04	11179.1	3142280	23676.47	12044.12	3096739	24449.28	10043.48	3338672	21000	15312.5
3096739	24449.28	9434.782	3287308	21861.54	11523.08	3096739	23434.78	11869.57	2374167	20377.78	12833.33	3338672	21437.5	15312.5
3142280	24911.77	9573.529	3237500	21848.48	11454.55	3052500	23200	11800	2374167	20377.78	12755.56	3338672	21437.5	15312.5
3096739	24855.07	9434.782	3287308	22292.31	11415.38	3096739	23637.68	11971.01	3096739	24855.07	9942.029	3391667	22111.11	15666.67
3096739	25057.97	9536.231	3237500	22272.73	11348.48	3096739	23840.58	11971.01	3142280	25323.53	10191.18	3338672	21875	15312.5
3237500	26303.03	9863.637	3237500	22272.73	11454.55	3142280	25529.41	12044.12	3096739	24855.07	9942.029	3338672	21875	15421.88
3096739	25463.77	9434.782	3338672	22968.75	11812.5	2967708	24694.45	11472.22	3052500	24800	9800	3338672	22312.5	15093.75
3189179	26223.88	9820.896	3338672	22968.75	11812.5	2967708	24500	11375	3096739	25666.67	9942.029	3287308	22292.31	15076.92
3096739	25666.67	9333.333	3287308	22723.08	11630.77	3237500	27363.64	12303.03	3096739	26478.26	10144.93	3287308	22292.31	14861.54
3189179	26432.84	9820.896	3237500	22696.97	11454.55	3009507	24943.66	11633.8	3096739	26884.06	10043.48	3287308	22723.08	14969.23
3189179	26641.79	10029.85	3287308	23153.85	11630.77	3052500	25600	11800	3189179	27268.66	10343.28	3237500	22484.85	14742.42
3142280	26558.82	9573.529	3189179	22671.64	11388.06	3052500	25800	11800	3096739	26478.26	10043.48	3287308	22938.46	14861.54
3142280	26352.94	9573.529	3237500	22909.09	11560.61	3052500	26000	11800	3052500	26000	10000	3237500	22909.09	14530.3
3052500	25600	9600	3237500	23333.33	11454.55	2967708	25083.33	11472.22	3009507	25140.85	9859.155	3287308	23369.23	14861.54
3142280	26352.94	9779.412	3142280	23058.82	11220.59	3052500	25400	11600	3052500	25200	9800	3237500	23333.33	14530.3
3237500	27363.64	10181.82	3237500	23969.7	11560.61	3009507	24943.66	11732.39	3052500	25000	10100	3287308	24015.38	15076.92
3189179	26432.84	9925.373	3237500	23969.7	11560.61	3009507	24352.11	11633.8	3052500	24600	10100	3287308	24230.77	14753.85
3142280	26147.06	9882.353	3287308	24661.54	11738.46	3009507	24154.93	11732.39	3096739	24652.17	10043.48	3189179	23716.42	14417.91
3189179	26223.88	10029.85	3237500	24393.94	11454.55	2967708	23333.33	11472.22	3052500	24200	10100	3287308	24876.92	14753.85
3096739	25260.87	9637.682	3142280	24294.12	11014.71	3009507	23760.56	11633.8	3009507	22971.83	9859.155	3189179	24552.24	14313.43
3189179	25805.97	10238.81	3096739	23840.58	11057.97	2927055	22726.03	11123.29	3096739	23434.78	10246.38	3189179	24761.19	14313.43
3096739	24855.07	9942.029	3096739	24246.38	11159.42	3096739	23840.58	11869.57	3009507	22577.46	10253.52	3189179	25179.1	14522.39
3096739	24652.17	9840.58	3142280	24911.77	11117.65	3009507	22971.83	11633.8	3189179	23716.42	10447.76	3142280	24911.77	14205.88
3189179	25179.1	10029.85	3096739	24855.07	10855.07	2967708	22166.67	11569.44	3052500	22400	10100	3142280	25117.65	14205.88
3096739	24246.38	9840.58	3237500	26303.03	11454.55	3096739	23231.88	11869.57	3189179	23507.46	10761.19	3142280	25529.41	14205.88
3096739	24043.48	9840.58	1765909	17413.22	15388.43	3009507	22183.1	11732.39	3142280	23058.82	10500	3096739	25463.77	14000
3189179	24343.28	10134.33	3052500	24800	11100	2967708	21583.33	11375	3096739	22623.19	10449.28	3189179	26432.84	14417.91
1795588	16647.06	11882.35	3142280	26147.06	11220.59	3009507	21788.73	11732.39	3142280	22852.94	10602.94	3142280	26352.94	14308.82
3287308	24661.54	10338.46	3189179	26641.79	11388.06	3052500	22000	11800	3096739	22014.49	10347.83	3189179	26850.75	14626.87
3287308	24230.77	10338.46	3052500	25600	11000	3052500	21400	11800	3096739	22014.49	10347.83	3142280	26764.71	14411.76
3287308	23800	10338.46	3009507	24943.66	10943.66	3009507	20605.63	11633.8	3096739	21811.59	10449.28	3142280	27176.47	14411.76
3237500	23333.33	10181.82	3096739	25463.77	11362.32	3052500	20400	11700	3142280	21411.77	10500	3096739	26681.16	14304.35
3237500	23121.21	10393.94	3052500	24800	11400	3096739	18362.32	12072.46	3189179	21208.96	10656.72	3052500	26000	13900
3237500	22909.09	10287.88	3052500	24400	11400	3142280	18529.41	12147.06	3189179	21000	10656.72	3096739	26072.46	14405.8
3237500	22696.97	10393.94	3096739	24652.17	11565.22	3142280	18323.53	12250	3189179	20791.04	10761.19	3096739	26072.46	14202.9
3338672	23187.5	10609.38	3009507	23760.56	11042.25	3189179	18283.58	12223.88	3189179	20791.04	10865.67	3096739	25463.77	14405.8
3237500	22484.85	10075.76	3142280	24500	11735.29	3237500	18242.42	12515.15	3189179	20582.09	10865.67	3189179	26014.93	14731.34
3287308	22507.69	10553.85	3052500	23600	11300	3189179	18074.63	12328.36	3237500	20575.76	11136.36	3142280	25323.53	14617.65
3287308	22507.69	10553.85	3142280	24294.12	11735.29	3237500	18030.3	12515.15	3189179	20164.18	11074.63	3142280	25117.65	14514.71
3287308	22076.92	10661.54	3009507	22774.65	11338.03	3237500	18030.3	12621.21	3142280	19764.71	10911.76	3096739	24652.17	14304.35
3189179	21417.91	10134.33	3096739	23434.78	11666.67	3189179	17238.81	12641.79	3237500	20151.52	11136.36	3052500	24000	14100

Human

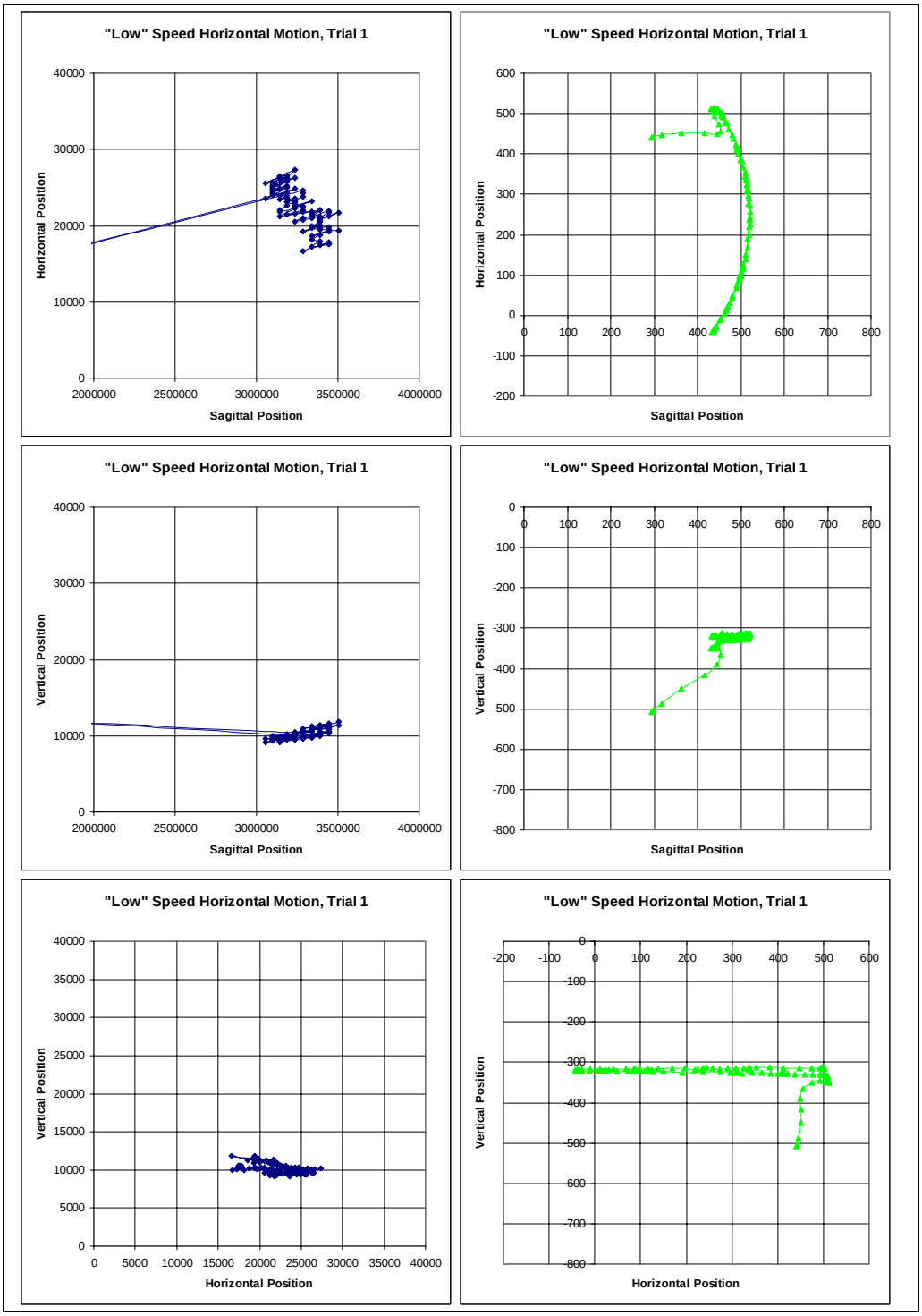
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
3338672	21875	10828.13	3815625	26750	12000	3287308	17769.23	12815.38	3287308	20138.46	11307.69	3052500	23800	14300
3391667	22111.11	10888.89	3096739	22420.29	11565.22	3338672	17937.5	12906.25	1874342	14245.61	10192.98	3009507	22971.83	13901.41
3338672	21437.5	10828.13	3189179	22671.64	12014.93	3287308	17338.46	12923.08	3142280	18529.41	11117.65	3052500	23000	14400
3338672	21437.5	10828.13	3142280	22441.18	12044.12	3338672	17281.25	12906.25	3338672	19687.5	11812.5	3142280	23264.71	14617.65
3338672	21218.75	10937.5	3142280	22029.41	12044.12	3287308	16907.69	12815.38	3287308	19061.54	11630.77	3052500	22400	14000
3446371	21903.23	10951.61	3189179	22253.73	12223.88	3338672	17062.5	13125	3237500	18242.42	11666.67	3052500	22000	14200
3446371	21677.42	10951.61	3142280	21617.65	12044.12	3338672	16843.75	13234.38	3237500	18242.42	11772.73	3096739	22217.39	14507.25
3391667	21000	11000	3142280	21617.65	12044.12	3391667	17000	13444.44	3287308	18415.38	11953.85	3189179	22253.73	14731.34
3502869	21688.53	11360.66	3142280	21411.77	12147.06	3287308	16261.54	13246.15	3287308	18200	12061.54	3189179	22253.73	14731.34
3446371	21225.81	11064.52	3237500	21848.48	12196.97	3338672	16843.75	13343.75	3287308	18200	12169.23	3189179	21835.82	14835.82
3391667	20777.78	11222.22	3237500	21848.48	12303.03				3237500	17818.18	11878.79	3237500	22060.61	14848.48
3391667	20555.55	11111.11	3287308	21861.54	12600							3237500	21848.48	15060.61
3338672	19906.25	11046.88	3237500	21000	12515.15							3142280	21000	14514.71
3391667	20111.11	11111.11	3237500	20575.76	12409.09							3237500	21424.24	15166.67
3391667	19888.89	11222.22	1907813	15125	10937.5							3287308	21215.38	15292.31
3391667	19888.89	11333.33	3287308	20353.85	12815.38							3189179	20373.13	14940.3
3287308	19276.92	10876.92	3391667	21000	13111.11							3338672	21000	15750
3391667	19666.67	11333.33	3338672	20562.5	12906.25							3237500	20151.52	15272.73
3446371	19870.97	11516.13	3391667	20555.55	13222.22							3338672	20562.5	15859.38
3446371	19419.36	11629.03	3391667	20555.55	13000							3338672	20343.75	15750
3338672	18593.75	11265.63	3446371	20774.19	13322.58							3287308	18846.15	15615.38
3446371	19193.55	11516.13	3391667	20111.11	13000							3237500	18242.42	15484.85
3502869	19393.44	11819.67	3446371	20096.77	13435.48							3237500	18030.3	15484.85
			3287308	18846.15	12600							3391667	18555.55	16333.33
			3446371	19645.16	13322.58							3287308	17984.62	15830.77
			3391667	19444.45	13222.22							3338672	17937.5	16078.13
			3446371	19193.55	13322.58							3338672	17500	16187.5
			3446371	19193.55	13209.68							3391667	17444.45	16333.33
			3391667	19000	13111.11							3338672	16843.75	16187.5
			3391667	18555.55	13111.11							3502869	17557.38	17098.36
			3561250	18900	13766.67							3446371	17387.1	16709.68
			3502869	18704.92	13655.74							3502869	17557.38	17098.36
			3502869	18245.9	13540.98							3446371	16709.68	17161.29
			3391667	17666.67	13222.22							3561250	17266.67	17733.33
			3502869	17786.88	13770.49							3561250	17266.67	17500
			3561250	18200	14000									
			3502869	17327.87	13770.49									
			3502869	17327.87	13885.25									
			3561250	17500	14116.67									
			3621610	17677.97	14355.93									
			3502869	16868.85	14000									
			1594590	11388.06	9246.269									

ISAC

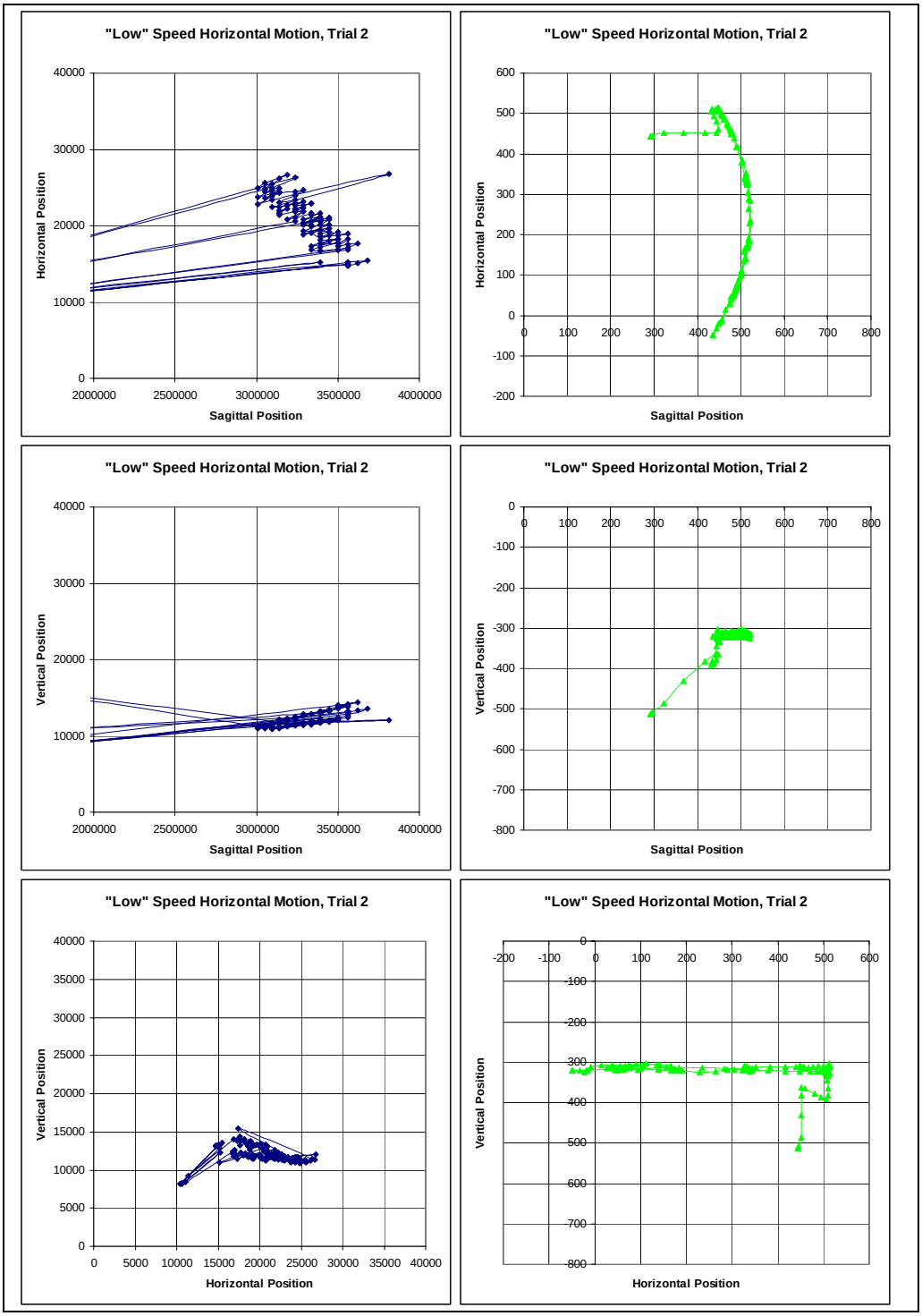
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
294.0538	441.1735	-507.3311	292.5485	443.3639	-512.196	294.3461	440.8828	-508.8801	293.785	442.2068	-509.0385	293.5092	443.0115	-509.5272
293.9906	441.1745	-507.3541	292.6638	443.1915	-512.0935	294.5847	440.707	-508.6974	293.6463	442.037	-509.0964	293.5466	442.9538	-509.4588
298.1601	443.3132	-503.8371	296.5796	445.2431	-509.028	297.4905	442.7967	-505.7819	297.3148	444.1345	-506.2361	294.439	443.5769	-508.3901
316.9782	446.1869	-486.4938	323.6904	451.6273	-486.1589	314.5168	446.0618	-490.41	314.3077	447.051	-490.0793	305.7148	447.2632	-497.537
363.5575	450.6598	-449.8083	369.0072	452.1182	-430.5188	355.0185	449.5677	-454.1959	410.6851	448.7481	-416.2402	326.0435	447.4755	-476.2508
416.8141	450.5986	-416.5959	417.8027	451.4352	-382.9942	406.6309	450.1652	-416.1798	425.444	443.4123	-388.6619	343.8304	453.7531	-453.5015
445.5829	450.2115	-390.8049	443.1907	452.4656	-362.4009	434.489	453.1991	-387.3999	431.5539	452.5526	-374.33	354.8803	458.6104	-446.6906
454.2706	455.5061	-366.1859	448.1	460.1908	-364.2355	444.0648	458.029	-362.962	433.7074	471.6794	-368.5612	351.8777	464.6098	-453.3075
448.7015	474.1907	-349.8027	443.2242	480.982	-376.9532	437.9383	476.0146	-351.9951	432.4438	494.5408	-367.5725	349.4253	474.2833	-468.872
440.1176	492.631	-346.4447	439.0471	494.2566	-387.6968	432.1039	493.2413	-350.0929	431.395	504.5014	-368.1542	346.9	480.6509	-482.8689
434.2158	504.137	-348.605	432.6146	506.0387	-391.8588	428.6861	509.1639	-353.0949	434.7718	503.1059	-367.8001	347.7035	488.4323	-482.3345
430.5649	512.1048	-350.8325	434.3461	510.5278	-383.2586	429.6902	514.2837	-359.3843	437.8883	503.0413	-364.2547	354.5756	491.2055	-468.095
436.3245	512.8662	-349.1752	441.6081	510.4419	-365.2742	435.8349	513.6349	-363.9041	444.5182	500.8868	-354.4746	366.2156	492.6398	-450.5599
441.4754	512.6851	-344.6287	443.716	509.1347	-344.5703	438.9958	513.0742	-361.3738	448.8168	500.9327	-341.5839	368.8071	492.3319	-437.4425
446.2085	507.6795	-340.4926	445.625	503.9269	-327.6961	444.6868	509.3632	-352.0698	449.6435	502.8257	-332.9073	370.6831	490.1251	-427.9744
448.0995	506.9844	-336.3024	444.0627	505.2295	-316.2616	447.342	509.6351	-340.3435	449.0407	503.7911	-331.8043	372.4187	493.2459	-421.9194
446.547	510.4749	-334.4819	442.1044	508.3485	-316.0133	445.0365	510.2634	-331.7684	448.2455	504.8253	-333.4687	369.8496	496.9556	-420.0899
447.0289	508.6427	-335.3841	442.0963	508.6292	-319.8955	441.0145	509.7003	-323.8103	447.5905	505.8759	-333.6881	370.0877	500.5669	-420.9101
447.5687	507.4827	-335.8858	443.3019	508.7949	-322.4899	442.3218	509.8833	-318.4694	447.6913	507.1199	-330.8706	372.1474	500.9397	-417.497
448.0168	507.1631	-335.008	443.5893	508.8344	-321.9309	442.2746	509.8767	-318.8115	447.5613	507.1974	-329.8619	377.6655	501.7245	-411.4059
448.067	507.1697	-332.6427	444.7981	509.0005	-320.7522	443.2578	509.0714	-320.4642	447.5722	507.1989	-330.0509	381.0879	502.0472	-405.5887
449.995	504.3845	-331.4187	447.5858	508.054	-323.8956	444.0252	508.9886	-321.5289	447.6158	507.2046	-329.674	384.1458	502.233	-399.9171
451.7484	501.5666	-330.9483	449.1417	507.7877	-328.8174	444.8192	509.0979	-321.3775	447.9267	507.2459	-328.672	385.1613	502.4592	-398.0335
455.1791	495.2302	-330.8836	449.558	509.1775	-333.5449	445.7713	509.2289	-323.4492	448.1013	507.2691	-328.1395	385.4164	502.4953	-400.4068
456.7311	491.6067	-330.8766	449.2919	509.618	-333.8615	446.0911	509.3677	-324.9575	447.5765	508.2425	-327.9198	386.7381	502.9316	-403.473
463.8631	477.1052	-330.4607	449.5622	509.6551	-332.726	446.5265	509.9971	-325.9697	446.2722	510.6264	-327.6693	390.8259	503.8482	-401.4731
472.2935	459.3527	-330.3033	449.7285	509.678	-332.1585	446.204	510.8068	-325.531	444.7571	513.0699	-327.7001	392.1329	503.8663	-397.1263
481.7756	437.3228	-329.8336	449.7762	508.8256	-331.7404	445.8909	511.2375	-325.7191	444.2444	513.9452	-327.7317	398.5436	504.6083	-391.9637
487.6788	422.5491	-328.7996	449.7174	508.8176	-331.423	446.0723	511.1682	-326.192	444.2045	514.0344	-327.7633	401.615	504.8722	-382.5696
490.3189	415.6205	-327.9839	448.3779	509.4924	-329.1988	446.3072	511.2015	-326.2887	445.4019	512.308	-327.1673	404.0957	505.0503	-378.1083
491.6232	412.4228	-327.3886	447.3775	510.6864	-328.0284	446.3602	511.209	-326.1004	447.0491	510.0698	-326.132	404.1464	505.0575	-378.4529
492.9025	408.838	-326.7916	446.5793	511.4303	-326.733	446.4131	511.2165	-325.9121	452.8637	500.3714	-325.3786	403.6701	504.9902	-379.9651
491.8688	409.8594	-327.1295	446.4239	511.4083	-326.5111	446.4261	511.2183	-325.9437	455.5386	494.8896	-325.3445	403.8763	505.0193	-379.6015
494.6124	401.0489	-327.5053	446.4239	511.4083	-326.5111	446.5054	511.2296	-325.6612	460.6445	485.067	-325.8466	404.8042	505.1504	-377.4892
499.3747	385.5035	-326.9415	446.4923	511.418	-325.7249	446.5581	511.237	-325.4728	467.9139	470.563	-325.7525	404.9094	505.1652	-376.8335
504.6563	366.2226	-325.7816	446.4019	511.4051	-325.3466	446.6501	511.2501	-325.2218	475.9114	453.5061	-325.0936	404.5889	505.12	-377.8215
509.409	343.4317	-326.3774	446.3885	511.4032	-325.4723	446.6891	511.2556	-325.3165	483.202	435.978	-324.9995	406.5786	505.8365	-375.361
512.8205	323.1552	-326.8798	446.4415	511.4107	-325.1267	446.768	511.2668	-325.0339	489.043	420.5016	-324.9369	409.2079	506.2982	-368.5679
514.594	314.04	-325.1242	446.4552	511.5078	-324.6553	446.8335	511.276	-324.877	494.4431	404.2743	-325.345	411.3947	505.905	-363.495
515.5297	308.8014	-323.7742	446.429	511.5041	-324.7495	447.1463	510.8443	-324.6885	498.3217	391.6894	-325.4081	411.7396	505.0733	-362.3471
515.1902	310.656	-324.1193	447.2615	510.0993	-324.7495	448.3132	508.9125	-324.5942	501.1394	382.1487	-324.9374	412.3534	504.3662	-362.7625
515.1792	307.629	-325.6566	448.556	508.1846	-324.1842	450.2416	505.8402	-324.1229	502.4007	377.3213	-325.0315	414.6234	502.9992	-361.1257
516.5995	296.6134	-324.8098	449.6385	506.8074	-323.2418	455.2989	496.9483	-323.7454	504.8767	367.1401	-325.314	419.8391	495.9767	-358.731
518.2699	276.1506	-324.2131	454.8602	497.5639	-323.053	472.5161	462.5608	-325.2241	510.0056	342.9217	-325.2819	423.1131	493.6077	-357.0688
518.9211	236.9597	-324.6206	457.2147	492.9789	-323.4928	481.3462	443.9211	-323.9682	512.8303	327.9013	-324.5287	429.147	490.3351	-351.5426
516.2602	190.7601	-324.8423	461.2418	485.4004	-323.5243	489.0703	424.0874	-323.7479	514.7074	316.934	-323.5867	433.4019	482.9756	-347.7406
510.9198	149.6715	-322.1414	468.3002	471.0547	-323.2723	497.8245	396.8599	-324.8489	515.3094	313.3583	-323.1156	437.3668	477.3119	-351.0869
506.0265	123.9026	-319.4049	477.7083	448.6312	-322.0447	506.5062	364.4856	-325.1639	515.0588	314.7926	-323.2726	443.8601	470.2095	-353.8238
504.2739	116.8554	-319.4049	490.342	415.7133	-321.8249	514.3935	325.8712	-323.1496	517.0391	296.8726	-323.6495	449.8279	460.8733	-352.1572
504.8976	122.2733	-322.2355	501.7888	378.7371	-321.1663	518.3317	295.8839	-322.646	518.8353	196.0186	-317.765	457.5539	445.1014	-350.5991
505.4751	126.0548	-323.5864	509.9841	344.6577	-319.6912	520.2362	272.6533	-321.8901	518.6566	204.748	-321.2295	466.7924	422.6636	-350.7323
505.5278	123.7996	-321.3239	512.6526	330.849	-318.9375	520.1324	271.5864	-322.1106	519.1117	210.5465	-321.607	477.1646	394.591	-348.3252
504.7763	119.1924	-319.1852	513.5508	325.6895	-318.6548	519.3778	274.4614	-323.9985	519.6592	211.8561	-319.3724	486.1577	362.1071	-344.6689
503.0817	113.9696	-320.4751	513.4118	327.1789	-318.2777	519.4421	268.4229	-324.501	519.6887	211.3142	-318.6172	492.6797	331.47	-342.4428
501.2363	108.6051	-322.1096	511.0105	338.8537	-319.6913	520.1162	250.0592	-323.3059	521.0646	238.3925	-319.3076	496.2188	308.8845	-342.0241
497.7795	96.5247	-321.9839	510.7741	337.6527	-322.1075	519.0061	216.5904	-322.8645	521.3063	257.7209	-318.2668	497.4808	297.0999	-342.6007
490.8219	74.55052	-321.3239	513.5692	323.0638	-321.3864	515.1851	175.3326	-322.8335	519.7612	276.3279	-319.3416	497.5533	295.5634	-342.8709
481.355	48.71761	-320.2549	517.0028	304.6559	-317.6152	505.208	122.1617	-321.9844	517.4653	285.6707	-322.7042	497.9173	294.4039	-342.5516
473.6568	29.9183	-318.9644	518.3736	289.1405	-317.9931	505.6081	121.045	-318.5544	518.9523	266.6951	-323.5228	499.5047	287.889	-339.1825
470.071	22.36298	-319.1219	518.9383	264.6899	-322.2028	505.3947	121.2028	-319.6249	519.1396	222.065	-320.4757	501.0544	276.23	-336.7106
469.1125	21.3112	-320.8523	519.5982	228.6079	-323.777	505.2636	121.0649	-320.0026	511.5347	152.1988	-318.6531	502.4214	258.1483	-334.7818
468.0303	19.83207	-322.1726	518.3664	191.975	-319.9054	502.8362	111.1849	-319.216	489.8345	67.79572	-319.4643	503.0485	232.068	-332.4245
465.7909	14.66413	-321.1353	516.8918	174.6596	-317.3191	498.7134	97.22324	-320.0342	462.3831	1.267775	-317.6819	501.8849	202.3764	-330.9779
460.9165	4.665319	-320.2547	515.9484	170.5979	-318.5501	490.9871	73.64151	-320.3174	443.8775	-31.37946	-316.3525	499.861	176.0702	-330.253
451.2489	-12.70276	-321.0095	515.6491	171.5475	-320.5358	479.4676	42.84528	-319.4041	438.9588	-38.59562	-315.8203	497.3555	153.3274	-328.4262
443.2309	-26.90327	-320.5372	517.5113	185.753	-320.2525	469.4644	17.35856	-319.8367	447.6861	-19.40857	-318.9967	495.035	139.611	-326.8732
438.8781	-34.15455	-319.7188	517.2267	187.8579	-320.4743	464.1637	8.43874	-317.1655	447.5426	-17.32843	-322.7996	493.2213	131.2214	-326.7216
438.4869	-34.0883	-321.4182	516.4847	178.7272	-320.1586	465.0145	10.64786	-314.3079	445.1161	-26.12143	-316.5016	491.0767	121.6258	-327.466
438.9167	-33.87948	-320.5056	516.4751	173.1471	-317.573	464.3571	12.28441	-316.1755	444.9351	-24.74796	-315.5689	486.8597	105.1221	-327.0129
438.8744	-34.64539	-318.9941	516.8029	175.5893	-316.7863	462.5169	7.751617	-317.8039	444.3152	-25.66254	-316.0406	481.3216	84.79103	-326.7607
438.6537	-34.99024	-318.9941	516.0348	176.257	-318.9324	460.0737	2.278404	-316.2934	444.1495	-26.23895	-315.3791	475.8744	65.30335	-327.5148
438.1945	-35.48892	-319.4666	514.5984	167.1069	-319.5619	459.2473	0.307336	-314.9722	443.4151	-27.64394	-314.7495	472.7255	50.77408	-328.9507
437.8524	-36.32355	-318.805	509.6612	140.1797	-318.902	459.2535	0.747406	-315.9794	441.2768	-30.99371	-314.9385	472.5947	44.12969	-329.4225
436.2793	-38.87654	-318.5214	498.4961	94.71449	-317.9561	459.3741	1.196163	-317.0796	437.8571	-36.16398	-315.5681	472.3902	42.7319	-328.4634
434.0554	-42.16936	-318.7105	483.9619	49.99521	-315.772	459.9024	1.743447	-316.3865	435.5411	-40.14502	-314.7798	468.2348	31.7784	-327.1342
433.5444	-43.10566	-318.3323	476.0922	30.36059	-313.4024	461.1419	3.655278	-315.3791	436.4309	-39.09879	-314.1491	461.244	15.76583	-324.1458
434.0124	-42.44476	-318.2378	474.598	27.09162	-313.4024	463.2278	7.787998	-315.6939	437.596	-36.95091	-314.8744	456.984	7.347497	-323.6155
434.5925	-41.44326	-318.5214	480.3699	43.54555	-317.1354	468.533	18.1256	-316.2894	438.1781	-36.29606	-316.2554	452.8802	-0.182954	-322.6471
435.5923	-39.88351	-318.616	482.4244	50.18342	-319.9706	477.2536	36.43645	-315.3439	440.3218	-33.41451	-315.3719	447.5574	-8.667405	-321.8
437.3127	-37.37335	-318.3323	482.7376	49.34148	-317.5131	486.6371	59.39403	-315.376	446.2937	-24.0422	-314.9613	442.8591	-17.157	-320.3376
443.212	-28.49556	-317.3866	484.8291	51.93269	-312.97	493.5981	79.65777	-317.0136	453.7863	-10.88696	-315.121	438.9954	-22.78079	-318.3128
454.0907	-10.35421	-316.7876	487.4025	60.23288	-312.9502	499.6718	96.92968	-317.9521	459.4448	-0.232995	-316.4752	436.3747	-25.36034	-315.7961
465.4398	11.16881	-317.9845	488.311	65.21411	-317.0466	505.2791	115.1504	-317.097	465.1002	9.820151	-316.3137	435.013	-27.14231	-312.6989
480.0018	41.35383	-317.5078	487.3446	62.7561	-318.1469	510.5122	137.6477	-316.9402	473.3146	25.82601	-314.7038	432.9778	-29.42393	-310.8469
490.3632	67.43886	-316.2794	486.5159	58.07231	-314.4595	516.4613	171.284	-317.1297	485.0387	52.65513	-314.1353	428.2276	-37.15195	-309.64
496.7387	89.78522	-318.618	486.0416	57.87128	-311.6396	520.2746	203.7051	-316.7501	494.8977	80.53252	-315.5876	426.3637	-40.23296	-309.144
496.9504	92.1935	-321.0094	491.9033	78.25757	-309.9462	521.9443	233.9785	-316.3711	502.5409	106.5066	-316.8496	426.2781	-40.78144	-308.3995

ISAC

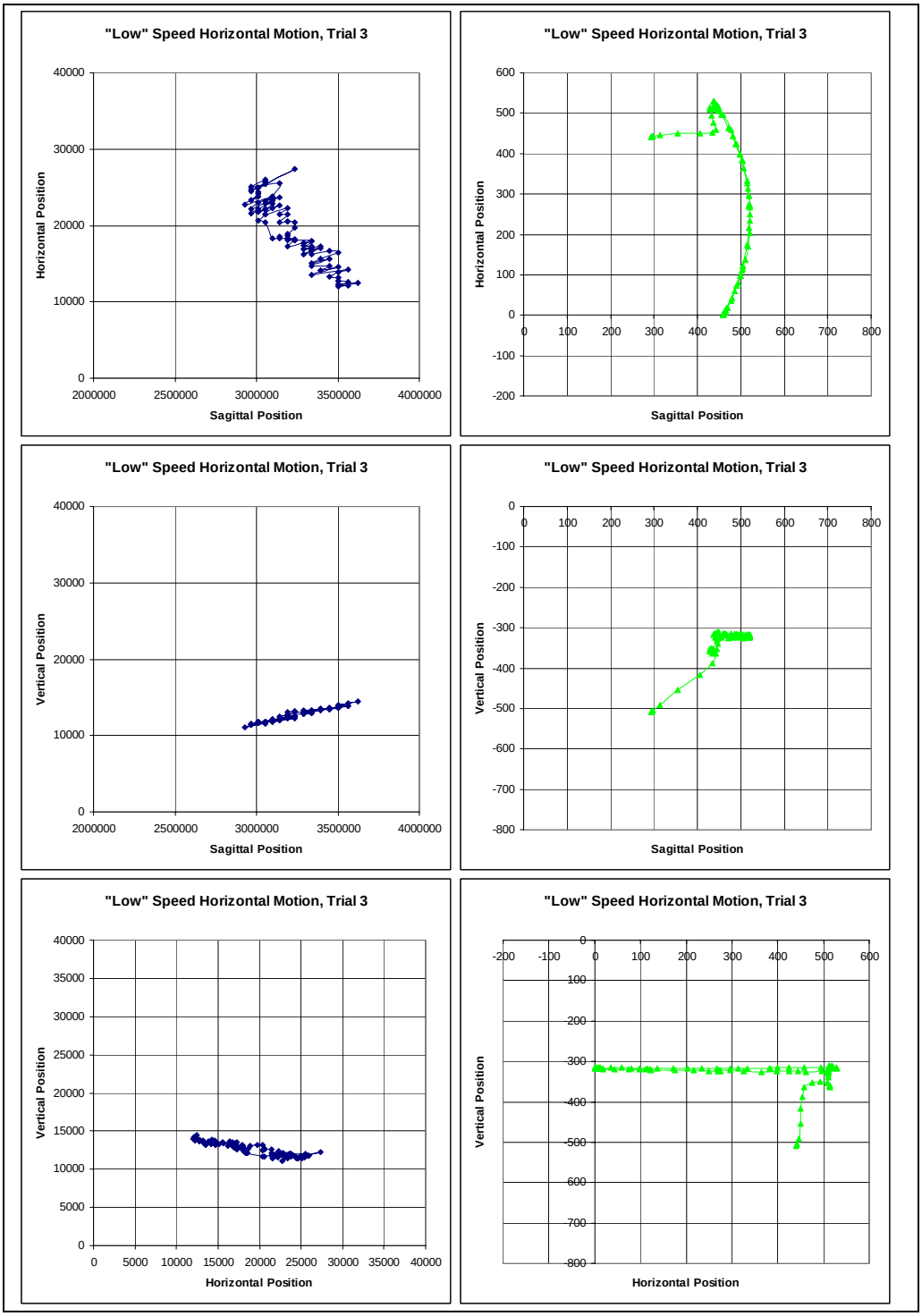
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
495.8189	84.48971	-318.1721	501.0349	106.6072	-306.7182	521.4955	266.458	-316.9716	508.1146	127.061	-315.4929	426.558	-40.52979	-308.1819
497.6379	87.54293	-313.7874	509.1075	141.5845	-305.7599	519.0197	296.662	-318.2349	511.0327	140.3895	-315.1471	426.6073	-40.60514	-308.3662
500.5732	97.66131	-313.9791	510.5324	166.9482	-308.8514	517.2304	313.1311	-317.9823	512.5154	147.9434	-315.399	427.0902	-40.15383	-307.7149
504.9783	115.6147	-316.6923	508.9149	157.578	-310.9663	514.6667	332.8054	-316.31	514.2549	156.9672	-315.1775	427.4176	-40.14357	-307.4634
510.4383	138.7319	-316.6603	507.2526	135.3117	-307.1469	503.5206	383.0782	-316.8462	517.2501	176.2053	-314.9567	428.6763	-38.16384	-307.7734
516.2917	168.7359	-314.6091	502.2075	112.0261	-303.5109	503.6282	382.6824	-316.8462	519.5342	195.968	-314.9567	431.6069	-33.67649	-308.1756
519.4632	197.9349	-315.6839	494.6802	90.16363	-305.3527	499.0814	399.5875	-315.5512	520.904	270.9068	-316.4411	435.6464	-27.56073	-308.576
520.2703	218.1817	-318.4273	478.0463	44.42834	-314.0882	491.1241	423.9945	-314.698	521.2537	270.1767	-315.5563	442.1126	-18.18804	-308.9696
520.7779	224.4252	-317.7336	455.6736	-14.70529	-319.8743	477.1633	458.8253	-314.3187	521.2394	277.2163	-313.8832	451.2443	-3.981374	-309.7019
521.875	236.1823	-314.5788	436.5851	-47.31503	-320.4018	458.5623	495.5879	-315.7729	520.4876	291.1466	-312.7456	463.4199	16.10712	-312.1705
522.1693	243.9926	-313.2855	435.1293	-49.78809	-320.0819	445.7129	517.3173	-316.8783	518.3519	311.451	-312.808	475.9731	43.59453	-314.0102
521.6578	257.1777	-314.8648	444.4555	-32.84487	-321.3039	437.156	529.6889	-316.7596	513.3641	342.5046	-313.2821	487.6331	73.50218	-313.9164
520.7739	272.5618	-316.4731	449.1776	-21.56536	-324.6684	438.0852	528.8009	-315.5299	505.2535	378.8338	-312.9977	498.9689	108.7079	-312.7914
519.84	290.1593	-315.306	451.0779	-19.03605	-320.5244	443.9906	519.8255	-314.6782	496.7974	407.2079	-313.4405	507.4608	144.9708	-312.1379
518.0996	309.4557	-314.1383	457.0425	-9.198405	-312.6927	448.456	513.8975	-312.7468	487.2087	433.5229	-314.4831	513.4671	185.2208	-311.8621
515.4259	326.2028	-315.4016	465.5832	13.75143	-306.962	446.3092	517.6015	-311.7387	480.0948	450.8211	-314.862	515.1927	218.7512	-311.2538
513.5074	335.7867	-316.8522	476.1205	37.75718	-306.1043	443.863	519.9009	-312.5738	475.3704	461.9962	-314.104	515.8688	240.7432	-310.3534
513.4042	337.1667	-316.2846	482.8694	55.16293	-308.4885	443.6315	519.8637	-313.8341	475.3828	462.1862	-313.7249	515.1276	252.8972	-308.7197
513.9798	336.263	-314.5807	487.9964	66.79307	-309.4116	446.0829	516.8953	-312.3162	475.595	461.7191	-313.7249	514.0725	259.4371	-306.6615
514.0518	336.6351	-314.0124	491.0496	74.59967	-308.5305	449.3766	512.1148	-310.8313	475.6374	461.6257	-313.7249	512.299	275.388	-305.3543
514.2206	335.2384	-314.5797	490.2357	72.6676	-309.5342	449.3893	512.3081	-310.3573	472.9319	467.4967	-313.7249	510.8609	288.6338	-306.4722
513.4777	339.6703	-314.3903	488.4876	63.24002	-311.5064	446.7159	515.7463	-312.6315	467.8205	478.1265	-313.7561	507.6818	313.4113	-306.6335
510.8429	354.3109	-313.348	491.7289	67.98865	-312.9555	446.4986	516.0965	-314.1091	460.8734	491.6307	-313.7561	501.793	336.9965	-308.8685
504.0055	383.016	-312.8428	494.3376	75.85344	-314.8171				453.98	504.123	-313.7249	495.2313	365.2409	-310.2208
494.7151	412.9683	-313.6323	502.4007	101.3027	-315.4785				449.5056	511.7502	-313.9145	490.0435	381.2824	-311.08
481.275	447.8928	-314.5797	512.4714	139.5263	-313.9233				448.2955	513.874	-313.7249	483.9534	398.4415	-312.8519
468.767	475.3394	-314.7692	520.017	184.2429	-312.7512				448.2955	513.874	-313.7249	477.1374	413.3348	-315.4321
460.1467	492.1309	-314.8639	523.2879	235.0461	-313.892				448.821	512.9926	-313.7249	470.0657	428.5826	-319.2515
454.5217	502.185	-315.2426	521.6142	284.5999	-315.412				448.9783	512.728	-313.7249	469.3337	430.5198	-319.1606
454.7961	501.8364	-314.9586	516.9135	324.0806	-315.6683							463.6923	444.2703	-319.1606
457.5538	497.4806	-313.7271	512.4707	345.1098	-315.9319							455.3595	460.6915	-322.6753
458.4821	496.4373	-312.3999	513.4948	339.3759	-315.8067							442.5921	482.6552	-326.5843
457.053	498.76	-312.9688	516.1027	329.4521	-312.1401							427.7492	503.3869	-332.0593
			516.8871	327.0193	-310.2736							417.7035	515.425	-337.9785
			515.9677	332.6035	-310.2425							415.5195	514.715	-345.1701
			511.8344	351.8078	-312.5209							417.3082	512.2035	-343.7821
			504.7348	382.444	-312.5483							416.5802	512.0022	-343.8415
			494.5786	416.4946	-310.8732							408.2216	510.7245	-350.5752
			485.8255	439.0517	-311.9204							404.3485	510.2199	-361.9604
			477.6621	456.8461	-314.6706							401.2647	509.9214	-370.5108
			475.9761	460.2141	-315.4599							400.0752	508.4419	-376.8906
			478.3721	456.4825	-312.4585							396.8942	506.5933	-383.1147
			481.888	449.5087	-310.1818									
			482.1836	448.7508	-310.2768									
			478.0883	457.1404	-312.1752									
			473.3973	466.5684	-313.9447									
			468.9757	476.6823	-312.3325									
			463.1674	488.5109	-309.6544									
			455.7923	501.0959	-309.9499									
			450.118	510.1138	-311.8451									
			447.808	513.9939	-311.971									
			448.1265	514.2324	-310.2654									
			448.5854	513.8206	-309.381									
			448.4633	513.5153	-310.5187									
			448.3224	513.4948	-311.0873									
			448.5335	513.5256	-310.2343									
			447.9514	513.4407	-309.2429									
			447.8539	513.4265	-308.3042									
			447.6827	513.4015	-307.9323									
			447.5567	512.8098	-308.1312									
			447.0369	512.5438	-306.6446									
			445.933	512.4798	-303.3018									



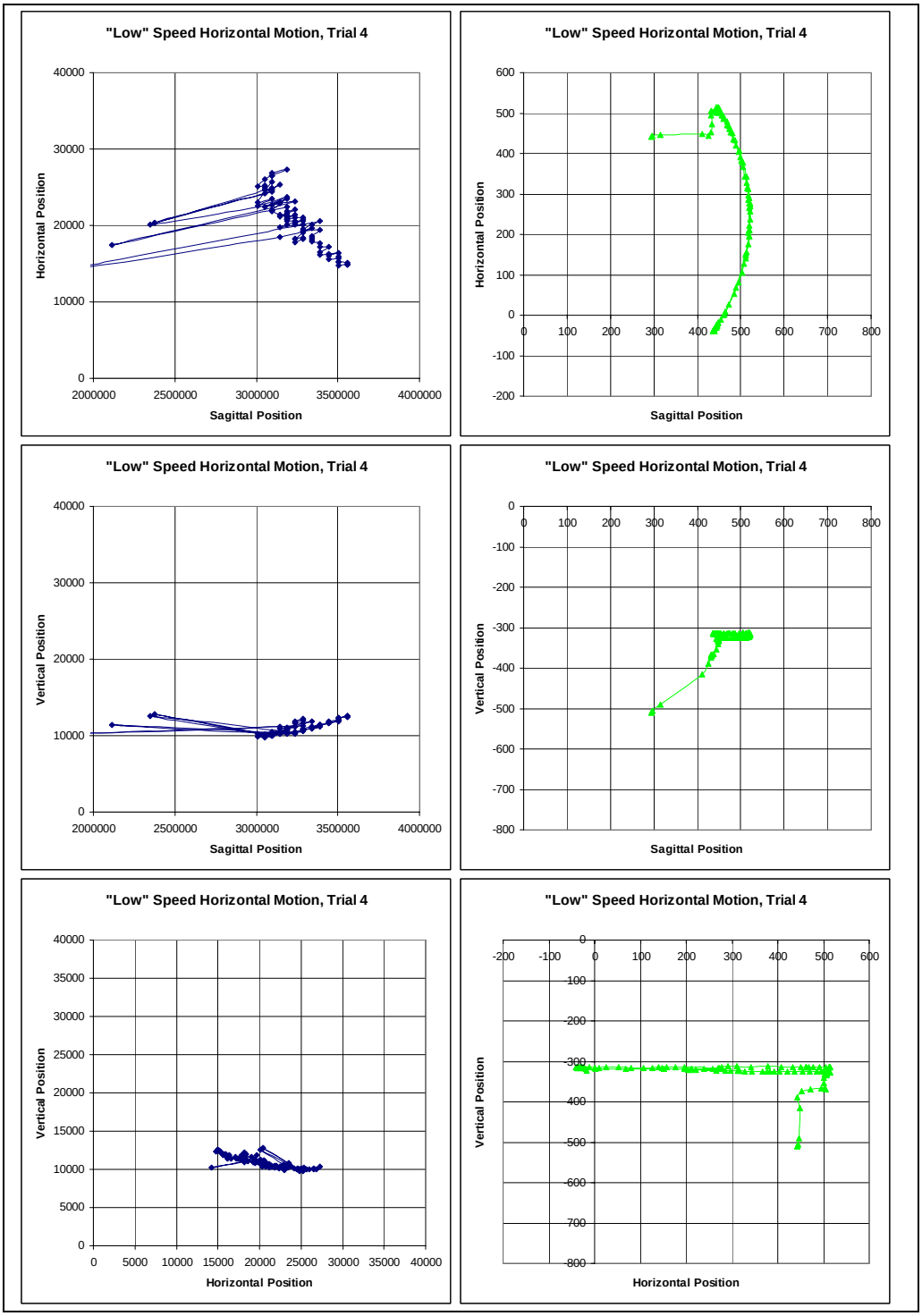
"Low" speed horizontal motion, trial 1; human motion is shown on the left, that of ISAC on the right



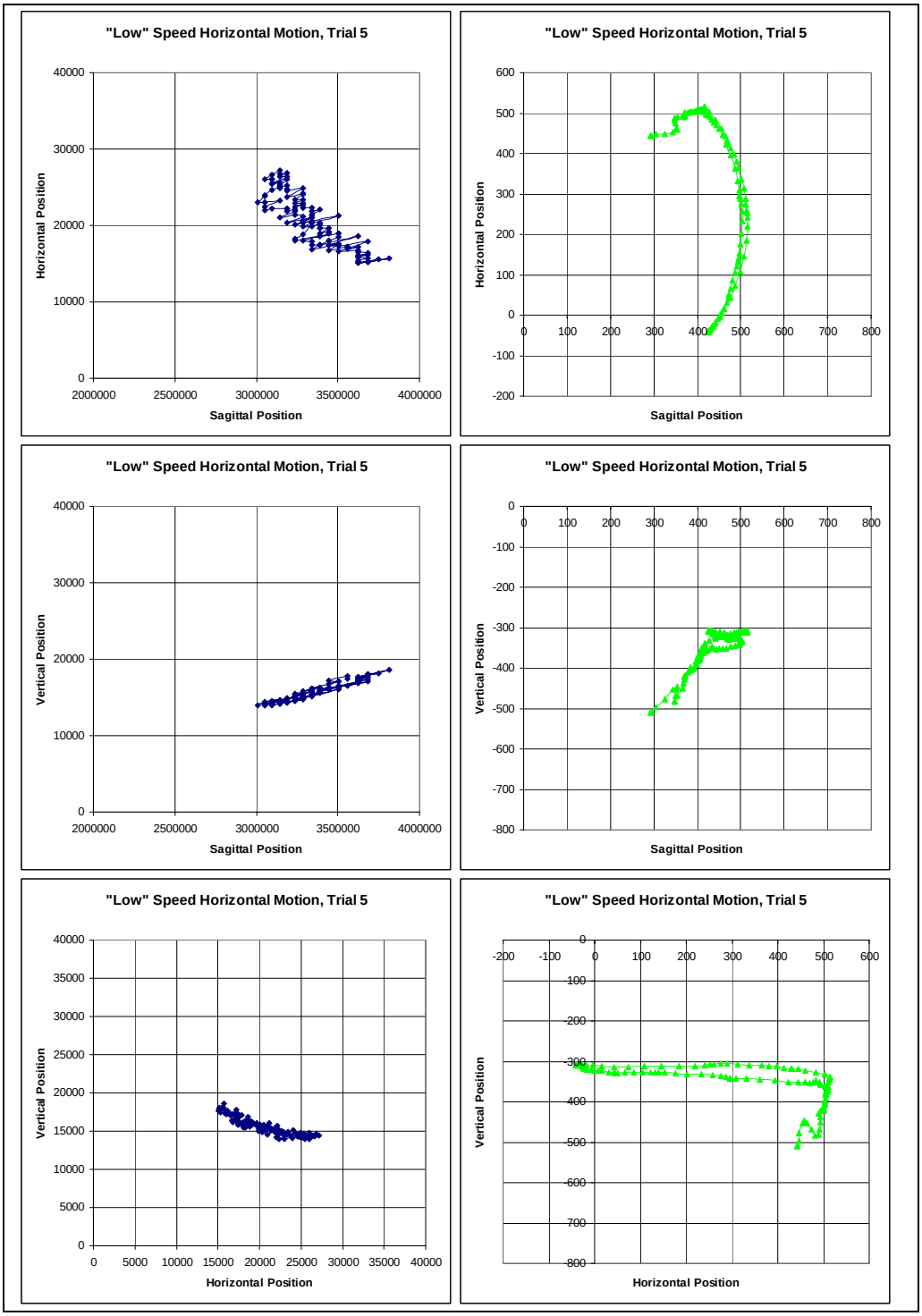
"Low" speed horizontal motion, trial 2; human motion is shown on the left, that of ISAC on the right



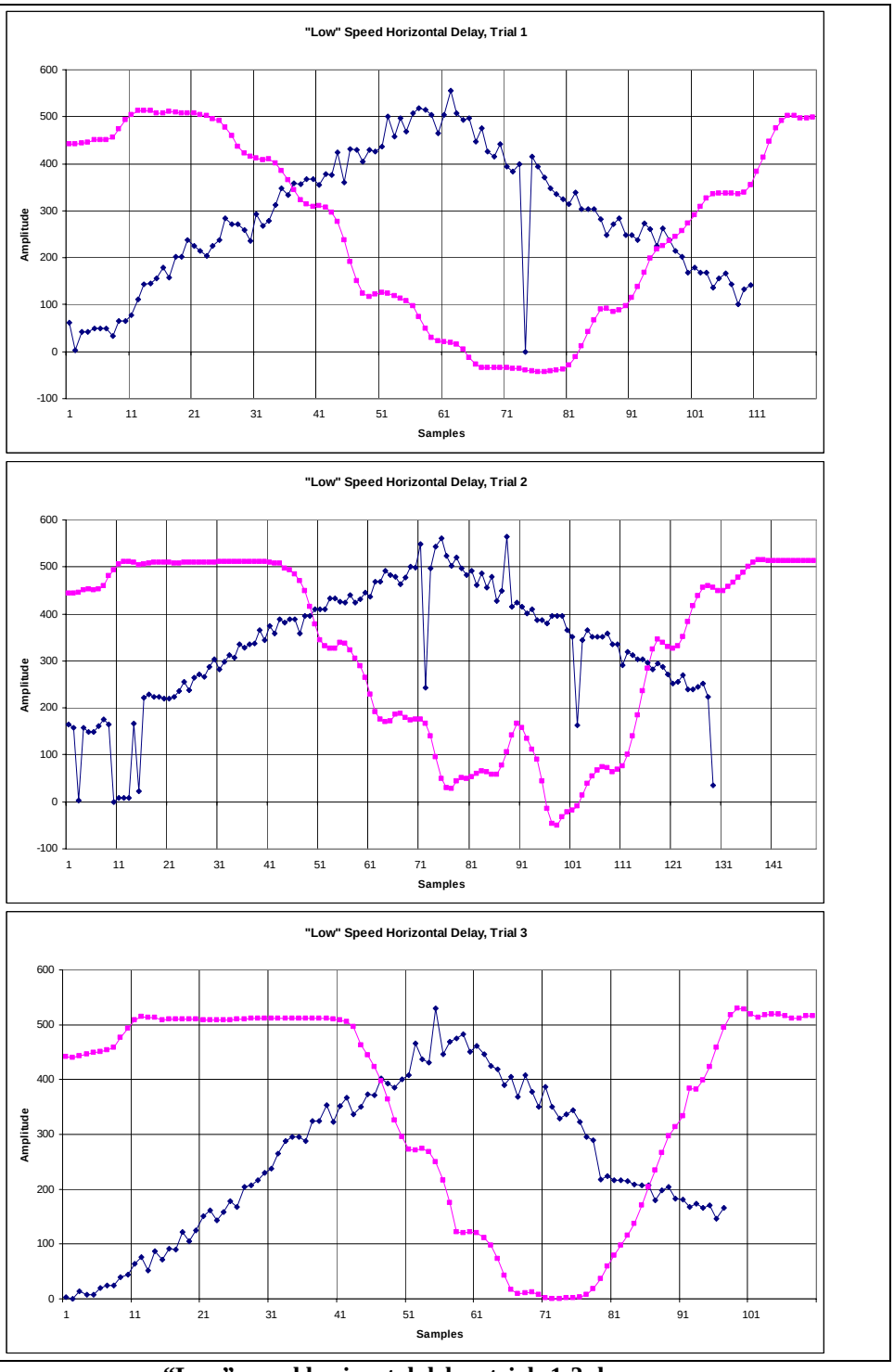
"Low" speed horizontal motion, trial 3; human motion is shown on the left, that of ISAC on the right



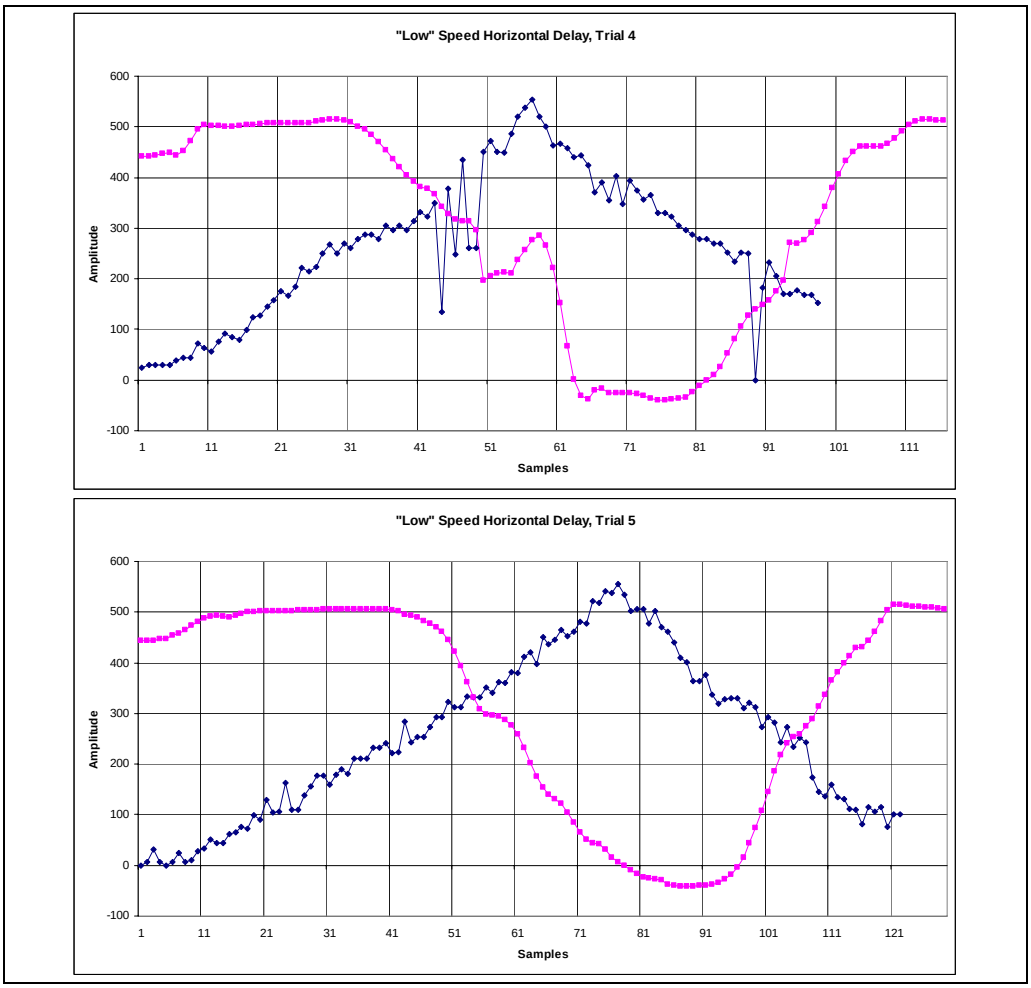
"Low" speed horizontal motion, trial 4; human motion is shown on the left, that of ISAC on the right



"Low" speed horizontal motion, trial 5; human motion is shown on the left, that of ISAC on the right



"Low" speed horizontal delay, trials 1-3; human motion is shown on the on bottom initially, that of ISAC on the on top initially



"Low" speed horizontal delay, trials 4-5; human motion is shown on the on bottom initially, that of ISAC on the on top initially

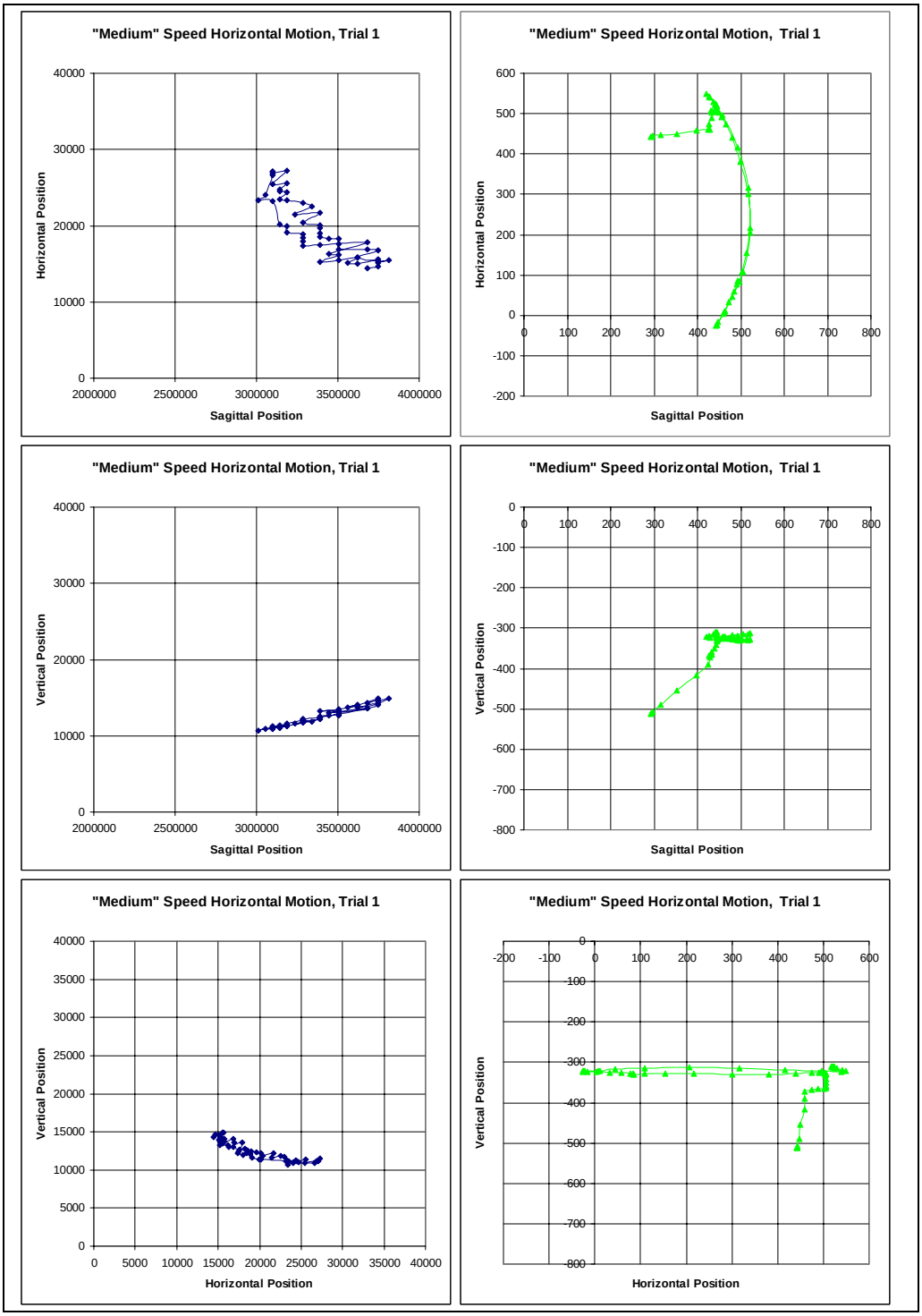
Horizontal motion at "Medium" speed

Human

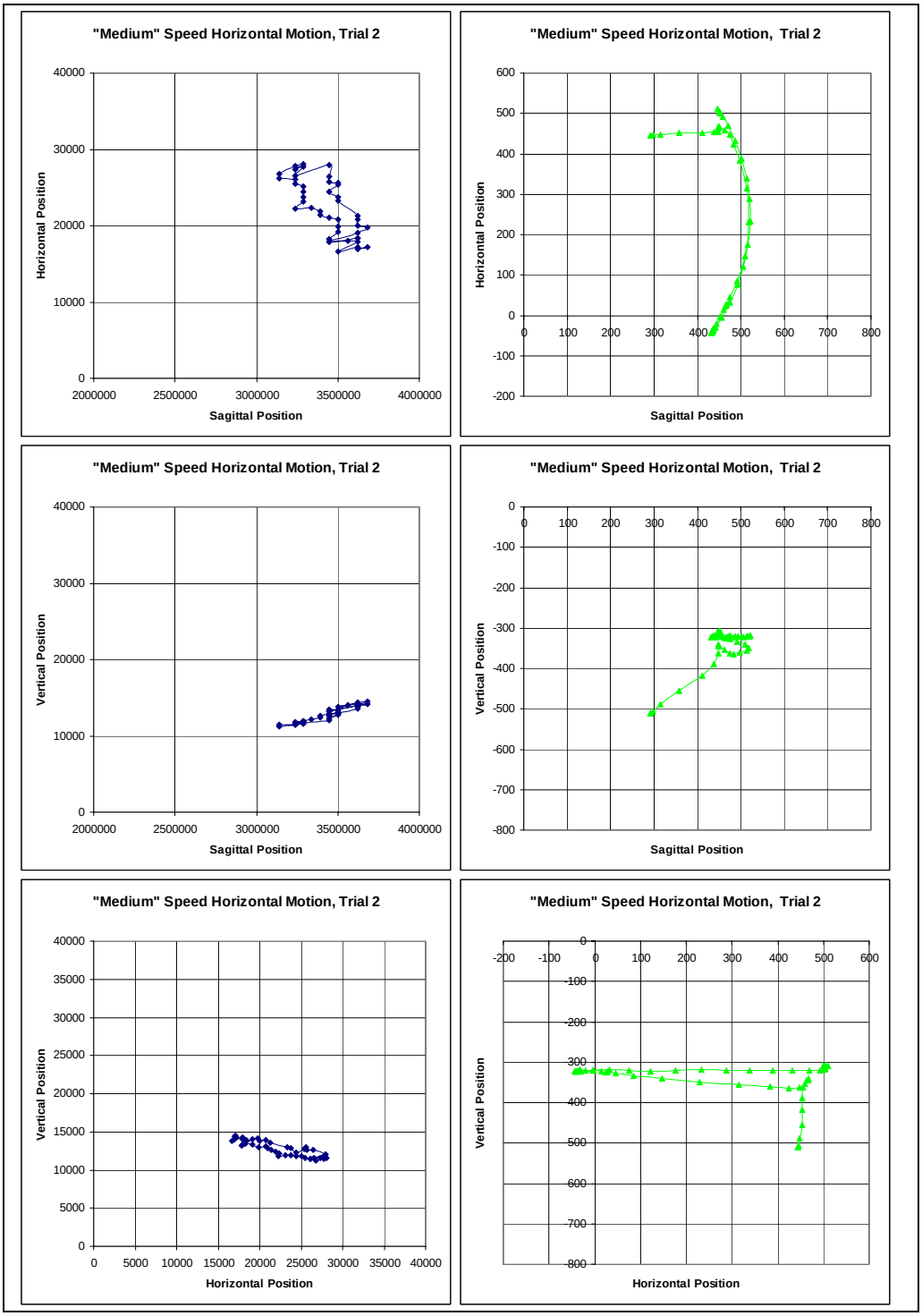
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
3684052	14482.76	14362.07	3621610	18389.83	13762.71	3621610	16728.81	13644.07	3446371	14000	13435.48	3391667	15888.89	14222.22
3748684	14614.04	14614.04	3621610	18389.83	13881.36	3684052	16896.55	13637.93	3502869	14114.75	13885.25	3391667	16333.33	14111.11
3748684	15105.26	14491.23	3446371	17838.71	13209.68	3561250	16566.67	12950	3502869	14344.26	13885.25	3338672	16187.5	13781.25
3815625	15500	14875	3621610	19101.7	14000	3446371	16032.26	12532.26	3502869	14573.77	13770.49	3287308	16046.15	13676.92
3748684	15350.88	14368.42	3684052	19793.1	14120.69	3684052	18827.59	12913.79	3561250	15166.67	13883.33	1526250	11600	10850
3621610	15779.66	13762.71	3621610	20050.85	13762.71	3561250	19133.33	12250	3561250	15866.67	13766.67	3446371	18290.32	13774.19
3748684	16824.56	14122.81	3621610	20762.71	13881.36	3502869	20540.98	12163.93	3391667	15888.89	12777.78	3338672	19031.25	13015.63
3684052	16896.55	13637.93	3621610	21237.29	13525.42	3502869	21918.03	11819.67	3391667	17222.22	12777.78	3189179	20791.04	12537.31
3502869	16868.85	12967.21	3502869	23295.08	12967.21	2094853	17156.86	11598.04	3338672	17500	12468.75	3096739	22014.49	11768.12
3502869	17557.38	12622.95	3502869	23754.1	12852.46	3287308	23800	10769.23	3338672	18375	12359.38	3189179	23925.37	11910.45
3502869	18245.9	12737.71	3446371	24387.1	12306.45	3237500	25666.67	10606.06	3142280	19147.06	11529.41	3052500	23600	11600
3446371	18290.32	12645.16	3502869	25360.66	12737.71	3189179	25805.97	10238.81	3189179	20164.18	11805.97	3009507	24352.11	11535.21
3391667	18555.55	12555.56	3502869	25590.16	12967.21	3142280	26558.82	10294.12	3096739	20594.2	11362.32	2927055	24068.49	11027.4
3391667	19000	12444.44	3446371	25741.94	12645.16	3189179	27059.7	10238.81	3009507	21197.18	11239.44	2927055	24643.84	10931.51
3391667	19666.67	12333.33	3446371	26419.36	12645.16	3189179	26641.79	10134.33	2967708	23527.78	10986.11	2849000	25293.33	10546.67
3391667	20111.11	12222.22	3446371	28000	12080.65	3142280	25735.29	9985.294	3009507	24549.29	11338.03	2887500	26486.49	10594.59
3287308	20353.85	11846.15	3237500	26515.15	11560.61	3096739	24449.28	9840.58	2887500	25540.54	10783.78	3009507	28492.96	11140.84
3391667	21666.67	12222.22	3287308	27676.92	11738.46	3142280	23882.35	10088.24	2887500	27810.81	10878.38	3684052	34758.62	18706.9
3237500	21424.24	11560.61	3237500	27363.64	11560.61	3096739	22623.19	9637.682	2849000	27533.33	10920	3684052	34517.24	19189.65
3338672	22531.25	11812.5	3287308	28107.69	11630.77	3237500	22696.97	10500	2811513	27078.95	10868.42	2775000	26818.18	10454.55
3287308	22938.46	11738.46	3237500	27787.88	11454.55	3237500	21848.48	10393.94	2775000	26090.91	10818.18	2811513	23026.32	10223.68
3189179	23298.51	11283.58	3142280	26764.71	11220.59	3189179	20164.18	10238.81	2775000	25181.82	10545.45	2887500	22324.32	10500
3142280	23470.59	11117.65	3142280	26147.06	11426.47	3287308	20138.46	10338.46	2704747	22240.51	10278.48	2927055	20424.66	10643.84
3189179	24343.28	11283.58	3237500	26090.91	11454.55	3287308	19492.31	10446.15	2811513	21921.05	10315.79	2967708	19833.33	10791.67
3142280	24500	11117.65	3237500	25454.54	11560.61	3287308	18415.38	10553.85	2887500	20243.24	10500	3009507	19225.35	10845.07
3142280	24705.88	11014.71	3287308	25092.31	11846.15	3446371	18290.32	11177.42	2849000	19133.33	10266.67	3052500	18400	11200
3189179	25597.02	11388.06	3287308	24446.15	11846.15	3502869	17786.88	11590.16	3009507	18042.25	10845.07	3142280	18117.65	11529.41
3096739	25463.77	10956.52	3287308	23800	11953.85	3446371	16935.48	11629.03	3052500	17000	11100	3189179	17656.72	11597.01
3189179	27268.66	11492.54	3287308	23153.85	11953.85	3621610	17203.39	11983.05	3189179	15985.07	11701.49	3237500	17181.82	11878.79
3096739	26884.06	11159.42	3237500	22272.73	11772.73	3621610	16491.53	12220.34	3287308	15830.77	12276.92	3287308	16692.31	12276.92
3096739	27086.96	11159.42	3338672	22312.5	12140.63	3684052	16172.41	12551.72	3237500	14848.48	12090.91	3287308	15830.77	12600
3096739	27086.96	11260.87	3391667	21888.89	12444.44	3684052	15689.66	12672.41	3391667	14777.78	12666.67	3391667	14555.56	13111.11
3096739	26681.16	10956.52	3391667	21444.45	12666.67	3748684	15105.26	13017.54	3502869	14573.77	13426.23	3391667	13666.67	13666.67
3052500	24000	10900	3446371	21000	12870.97				3502869	13196.72	13655.74	3561250	14000	14233.33
3009507	23366.2	10647.89	3502869	20770.49	13081.97									
3096739	23231.88	11159.42	3502869	19852.46	12967.21									
3142280	20176.47	11323.53	3502869	19163.93	13311.48									
3189179	19955.22	11388.06	3446371	18290.32	13435.48									
3189179	19119.4	11597.01	3446371	17838.71	13209.68									
3287308	18846.15	12061.54	3561250	17966.67	14000									
3287308	18415.38	12169.23	3621610	17915.25	14237.29									
3287308	17984.62	11953.85	3502869	16639.34	13770.49									
3287308	17338.46	12169.23	3621610	17203.39	14237.29									
3391667	17444.45	12444.44	3621610	16966.1	14000									
3684052	17862.07	13637.93	3684052	17137.93	14482.76									
3446371	16258.06	12983.87	3621610	16966.1	14355.93									
3502869	16180.33	13196.72												
3391667	15222.22	13222.22												
3502869	15491.8	13540.98												
3621610	15779.66	14118.64												
3561250	15166.67	13766.67												
3621610	15067.8	14000												
3748684	15596.49	14859.65												

ISAC

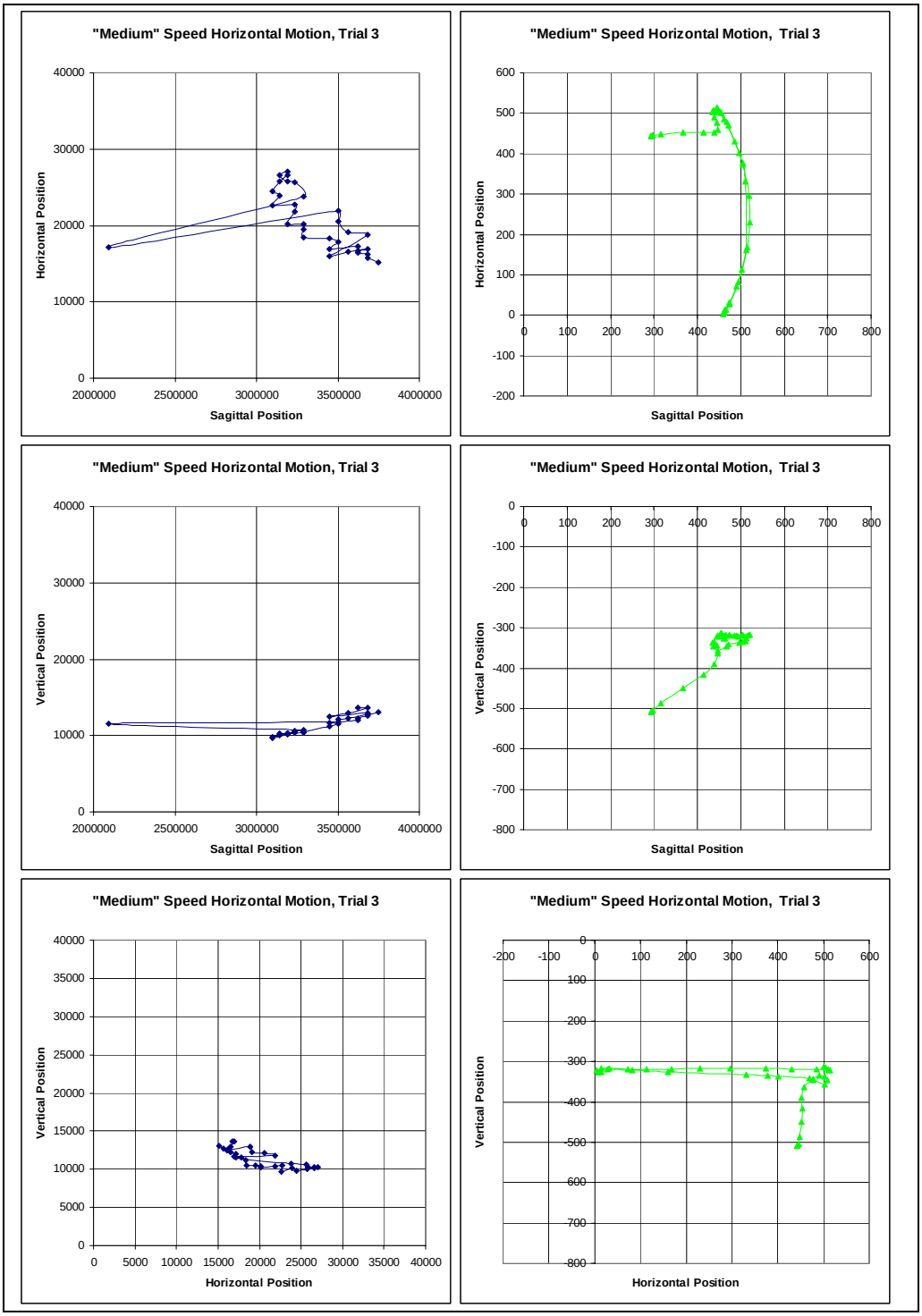
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
293.2066	442.5571	-511.193	291.9809	444.3395	-509.7233	292.9825	443.1884	-508.2301	295.1318	440.2962	-507.0786	293.759	441.9211	-507.8431
293.3069	442.4987	-511.0333	291.9803	444.3395	-509.6891	293.1067	443.1872	-508.0816	295.1309	440.2962	-507.0445	293.7582	441.9211	-507.8089
297.3588	444.3088	-507.4182	297.5569	446.585	-505.3053	297.9124	445.532	-504.368	301.8856	443.5155	-501.8766	293.8489	442.435	-507.4425
314.7944	446.6808	-490.002	314.8136	447.735	-488.3445	315.6008	447.5527	-487.2243	332.3957	448.2251	-474.4652	303.9487	447.975	-500.9334
351.6062	449.742	-455.3446	357.3288	452.7056	-454.1891	366.7289	451.7171	-450.0184	382.9317	445.3231	-420.3081	350.4182	455.6475	-469.2169
398.5088	458.6312	-416.5161	410.5182	452.8034	-418.1853	414.4434	453.1967	-417.4761	415.1795	449.4331	-383.6898	404.8973	451.7485	-420.7689
423.7281	459.7627	-389.1244	438.1626	454.4752	-388.1269	439.4991	452.8467	-389.1566	433.0526	451.633	-368.7918	436.4079	448.318	-379.4843
429.8961	460.0474	-373.1476	447.6703	455.1022	-361.9169	447.7586	458.7523	-362.8544	436.9098	463.6829	-372.7565	445.1749	456.1119	-343.5136
427.7483	473.8477	-366.8406	448.9238	462.8168	-345.8237	444.8736	476.9352	-343.5155	434.9005	482.0296	-384.7319	436.3668	476.0861	-328.1563
431.9834	488.2923	-365.7736	448.0021	467.7249	-340.8661	439.7855	490.7703	-335.2527	437.6593	494.6552	-393.7705	433.1722	494.1684	-330.5692
432.2232	503.1462	-364.6704	450.2563	467.652	-343.6839	435.6699	503.3126	-336.7218	435.1715	505.1829	-390.6014	431.9188	507.8744	-343.0446
431.0455	506.0107	-363.0892	461.8294	458.691	-353.7665	437.5979	507.546	-346.5576	439.0933	508.2166	-377.3103	435.931	510.6626	-362.2816
433.9756	506.4063	-359.2478	474.2988	447.1189	-362.1489	446.2287	503.063	-356.7186	444.7441	508.1432	-359.4537	442.5893	510.2978	-378.4553
438.6346	507.0353	-351.0864	483.9597	423.4772	-363.61	466.9078	478.1596	-345.577	446.2006	505.1295	-339.1803	445.4642	507.674	-379.4778
442.4837	506.6168	-341.0564	497.5037	383.3783	-361.0086	471.5157	469.801	-341.6723	445.1856	501.0606	-321.7789	452.0458	504.1674	-367.5116
446.3807	504.5874	-332.8559	513.1565	315.2062	-355.6654	496.3878	401.7371	-336.8558	445.8673	502.3611	-309.1962	459.7364	494.1852	-350.3201
447.4227	505.0036	-328.708	518.1717	229.879	-348.405	503.8091	377.4093	-333.9829	446.4138	501.396	-307.7734	459.275	475.1251	-330.2789
446.0757	506.9057	-328.6413	509.6263	147.4776	-340.8441	510.069	330.9754	-332.4173	450.337	493.5954	-311.7069	467.0738	443.3418	-312.2388
445.8722	506.9731	-329.5482	490.5981	84.14917	-332.719	512.4786	160.544	-326.702	457.752	481.5478	-314.1812	493.2227	369.2992	-301.8532
447.5695	505.0224	-328.428	474.9491	44.91485	-326.4934	494.6959	82.08616	-320.9132	469.9686	458.0724	-314.3016	510.1314	273.3145	-306.497
449.3738	502.9815	-326.8361	467.5026	27.59376	-323.5271	472.9928	27.71424	-318.8368	489.0438	412.6953	-318.0358	510.7055	186.2488	-315.6721
456.2104	491.0842	-325.96	466.7348	26.21276	-324.0257	460.1828	3.48062	-322.0159	513.305	331.5679	-327.3094	500.3433	116.175	-322.8793
464.9577	473.8811	-326.8694	465.5533	23.89033	-324.306	461.0785	7.558411	-325.7846	525.332	210.4347	-341.4764	496.6656	73.02925	-334.643
479.5595	440.36	-328.3396	461.1735	14.12461	-322.9041	463.5289	13.53728	-327.2567	509.7648	108.8749	-348.1434	491.5064	63.28421	-347.8969
499.5876	381.5945	-329.223	453.0906	-2.396097	-321.002	462.4181	10.05774	-325.4391	481.1415	28.05927	-346.6998	492.2761	67.87524	-353.5154
517.1816	299.7841	-329.2606	443.7288	-20.07243	-319.6291	466.2341	13.57709	-317.673	462.9186	-9.048209	-340.8402	494.5841	74.36595	-349.0833
520.9776	215.7396	-328.018	437.9102	-31.28609	-319.1913	473.5767	30.97455	-316.4869	450.4278	-16.45138	-330.955	474.2195	36.06815	-324.8687
513.8178	154.5505	-327.955	436.4728	-33.82813	-319.7237	490.1419	71.98932	-318.7784	451.4557	0.854185	-316.4781	463.7021	17.21218	-310.8968
503.6373	108.6534	-328.6525	436.3718	-34.01272	-321.2906	502.9436	113.2442	-319.3751	449.5818	3.942634	-301.5956	448.917	13.08521	-295.3526
495.8795	83.56452	-329.1885	436.2423	-36.14803	-322.7063	514.6481	168.733	-318.4627	445.827	-8.703359	-295.0982	445.94	14.00314	-279.7193
494.9936	82.82885	-328.9919	435.606	-37.55428	-322.6756	520.3571	229.2992	-317.6129	450.4903	-0.470725	-299.863	444.6659	10.67245	-274.7901
493.4953	85.73747	-329.3698	434.615	-39.11471	-322.5813	518.1157	296.8393	-317.8644	457.5658	9.130747	-303.1466	440.8779	2.569994	-279.8163
490.648	77.23123	-328.3936	432.9124	-41.64996	-322.6756	504.7499	373.2661	-318.179	471.6335	22.46605	-310.6191	437.876	-8.100329	-286.0259
484.0425	58.48038	-325.659	432.111	-43.5088	-322.5192	486.2231	431.3745	-319.0914	492.7468	65.2514	-323.4915	437.3654	-17.81785	-290.0367
472.9424	31.93177	-325.1247	432.7387	-42.91894	-322.2363	461.3713	485.4113	-320.5698	512.2576	121.0101	-333.4725	456.6939	-25.02279	-344.2411
459.6869	4.227911	-323.9002	433.6723	-41.42869	-322.425	447.139	510.2725	-322.2031	528.2858	209.0888	-334.3842	468.5489	2.533694	-372.3973
448.2398	-16.84495	-323.87	435.5903	-38.53613	-322.4565	444.8131	514.0294	-322.4857	524.5122	319.8146	-331.3959	496.2317	56.92886	-370.219
442.996	-25.44966	-323.8382	441.1935	-30.38584	-322.205	447.8048	510.3653	-320.0668	497.0434	425.0726	-331.6837	520.7425	124.3309	-346.5375
443.2029	-25.89689	-322.2358	456.6466	-5.80353	-320.5026	453.0235	503.7178	-316.0044	453.9024	501.3488	-330.6942	528.4243	214.8029	-322.9058
443.8644	-25.52366	-320.6948	475.5028	31.26658	-319.3353	455.6299	500.886	-312.3126	419.4857	543.9167	-319.4	509.9963	317.5027	-300.6819
445.5324	-22.68606	-320.8836	492.6148	75.22846	-320.881				417.1652	547.6903	-303.6208	489.5638	407.3136	-282.6081
462.3906	8.194044	-322.2358	505.9867	121.353	-322.0789				425.3271	536.7728	-295.2735	454.6936	482.2435	-280.2401
464.6074	11.41314	-320.4429	516.9757	176.0364	-320.4386				434.916	524.0367	-297.0249	428.285	524.7657	-294.1186
481.2241	45.37201	-316.8829	521.7756	232.706	-319.0817				438.1327	521.447	-306.2217	424.063	536.9902	-307.8972
504.5135	109.0545	-313.841	520.0886	287.6924	-319.8394							429.3132	531.6608	-308.8361
521.6904	206.6036	-312.888	513.1796	338.7662	-320.8806							444.5563	523.6705	-306.2467
518.1059	316.331	-314.6585	501.6546	388.7437	-320.5967							452.61	515.9582	-311.4939
493.5696	415.891	-318.7347	486.7401	432.8723	-319.8099									
456.7725	496.2535	-321.9846	471.2764	468.6522	-320.1886									
428.0647	539.4413	-323.6464	458.2305	492.6066	-320.506									
421.2646	548.2519	-322.0432	452.0666	502.4588	-318.6232									
427.0466	541.7283	-318.1853	449.734	504.8256	-315.3385									
436.4154	529.5528	-314.1616	451.1821	502.1617	-312.0874									
442.4622	522.1655	-309.9366	452.2322	500.3922	-309.4322									
441.6691	522.8965	-309.8492	452.6919	499.7827	-308.088									
440.9934	522.3045	-313.4723	449.4351	504.6928	-308.1595									
440.3094	523.7213	-314.7872	445.4703	510.6086	-308.4493									
443.4142	519.9245	-311.9843												
445.1551	517.7098	-309.5299												
443.7146	518.0587	-310.2746												



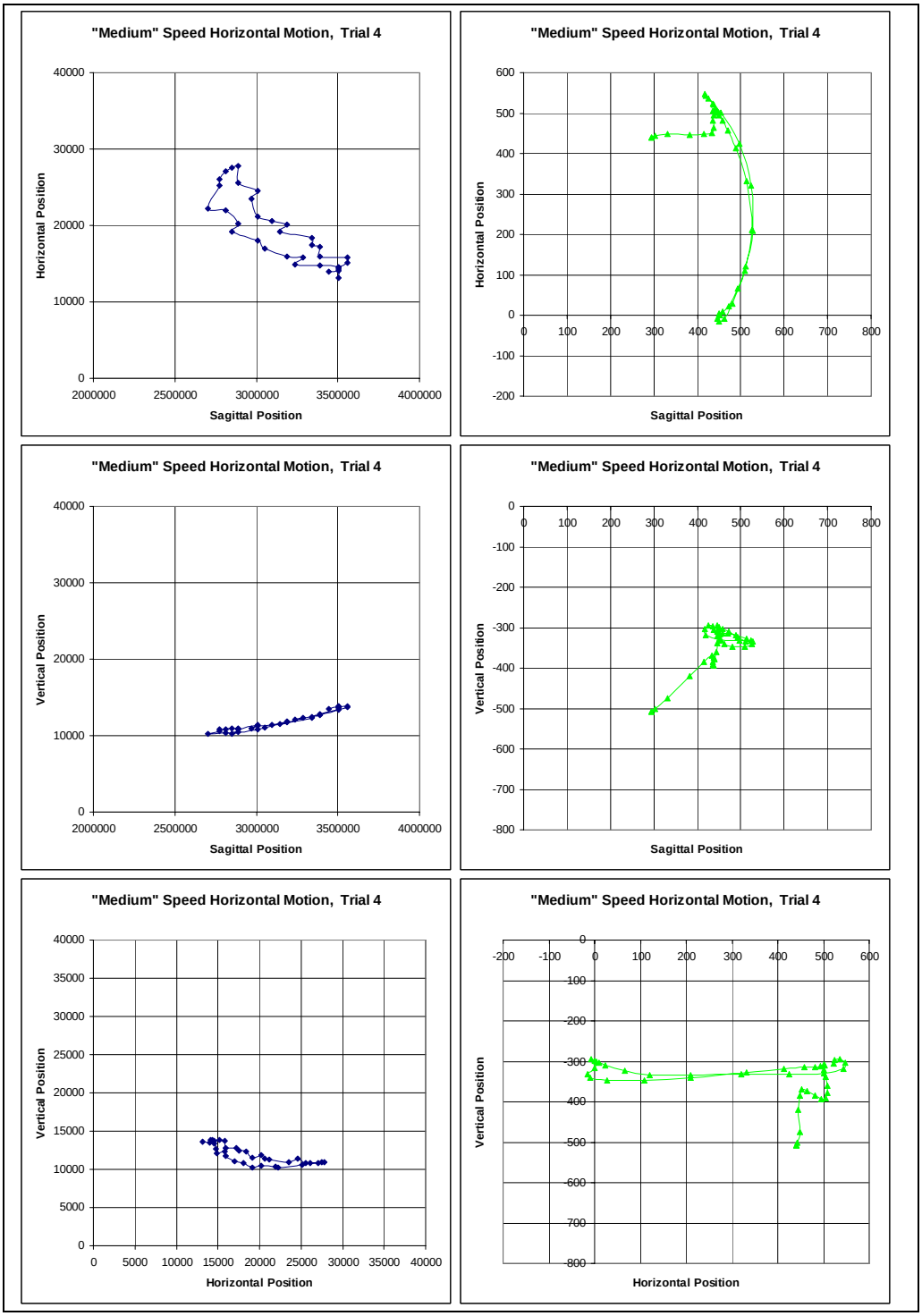
"Medium" speed horizontal motion, trial 1; human motion is shown on the left, that of ISAC on the right



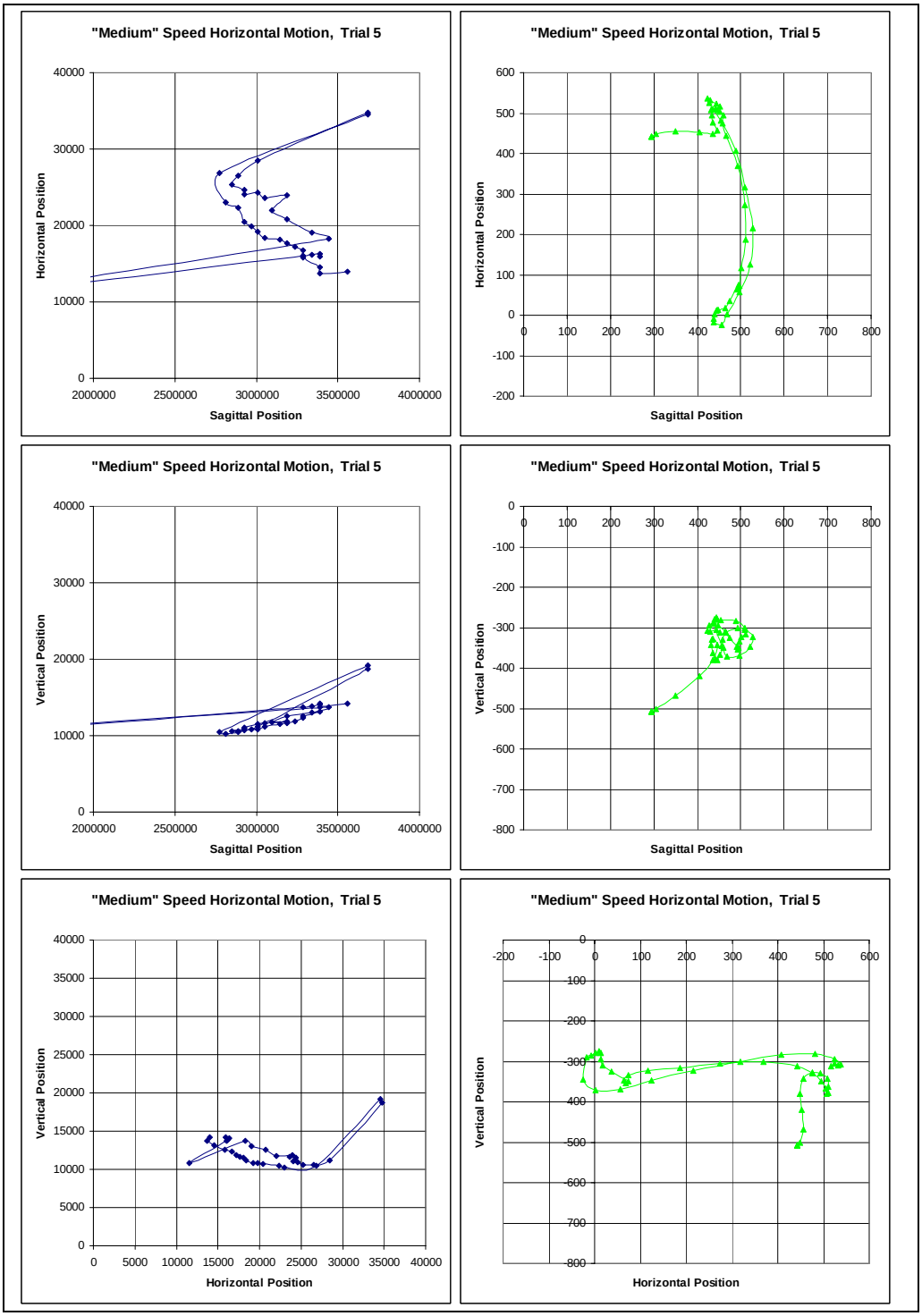
"Medium" speed horizontal motion, trial 2; human motion is shown on the left, that of ISAC on the right



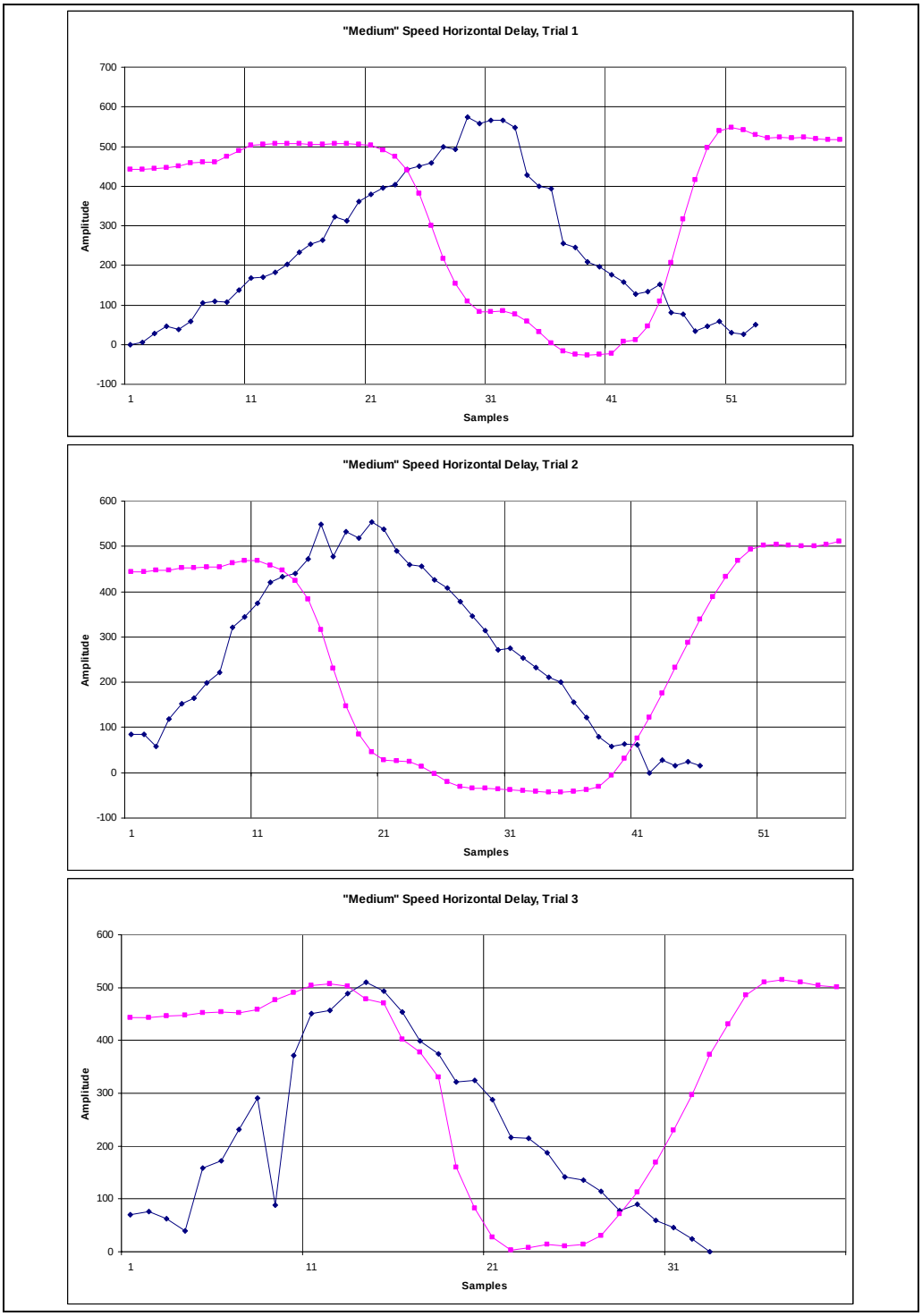
"Medium" speed horizontal motion, trial 3; human motion is shown on the left, that of ISAC on the right



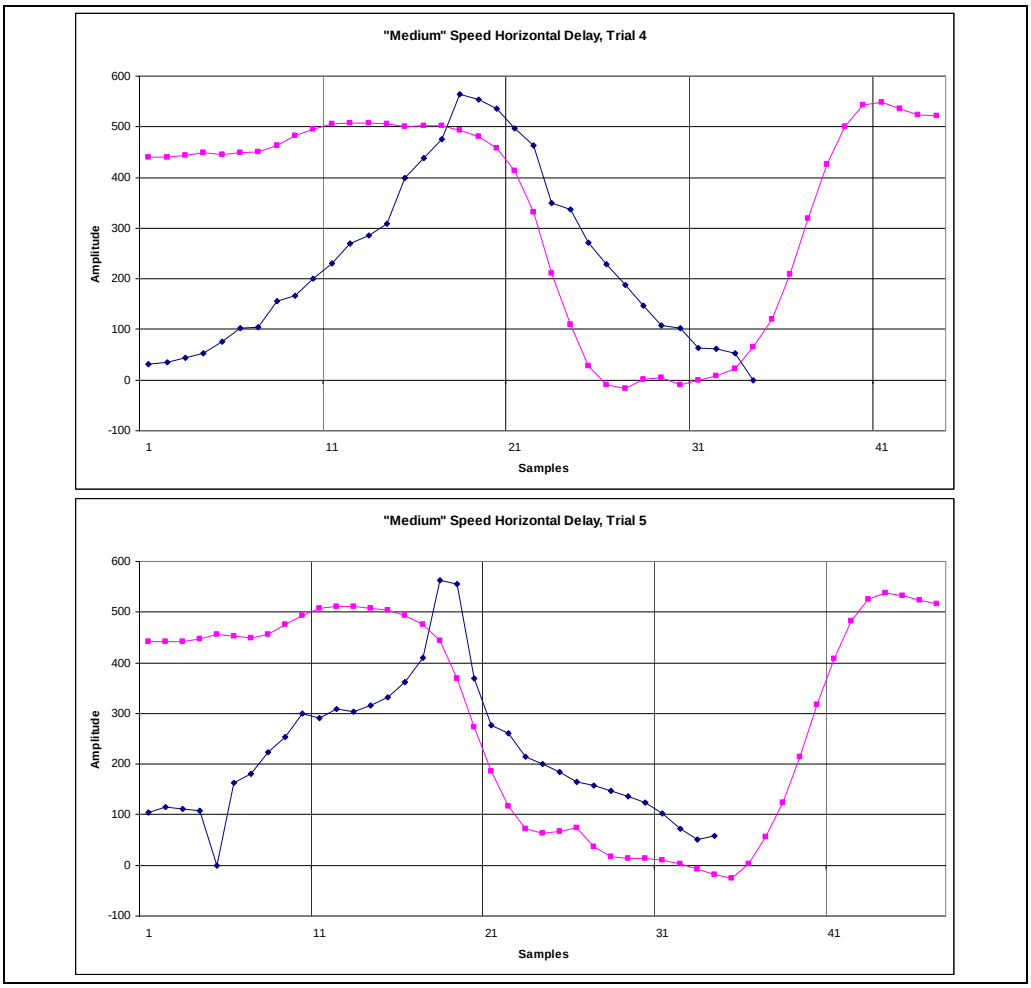
"Medium" speed horizontal motion, trial 4; human motion is shown on the left, that of ISAC on the right



"Medium" speed horizontal motion, trial 5; human motion is shown on the left, that of ISAC on the right



"Medium" speed horizontal delay, trials 1-3; human motion is shown on the on bottom initially, that of ISAC on the on top initially



"Medium" speed horizontal delay, trials 4-5; human motion is shown on the on bottom initially, that of ISAC on the on top initially

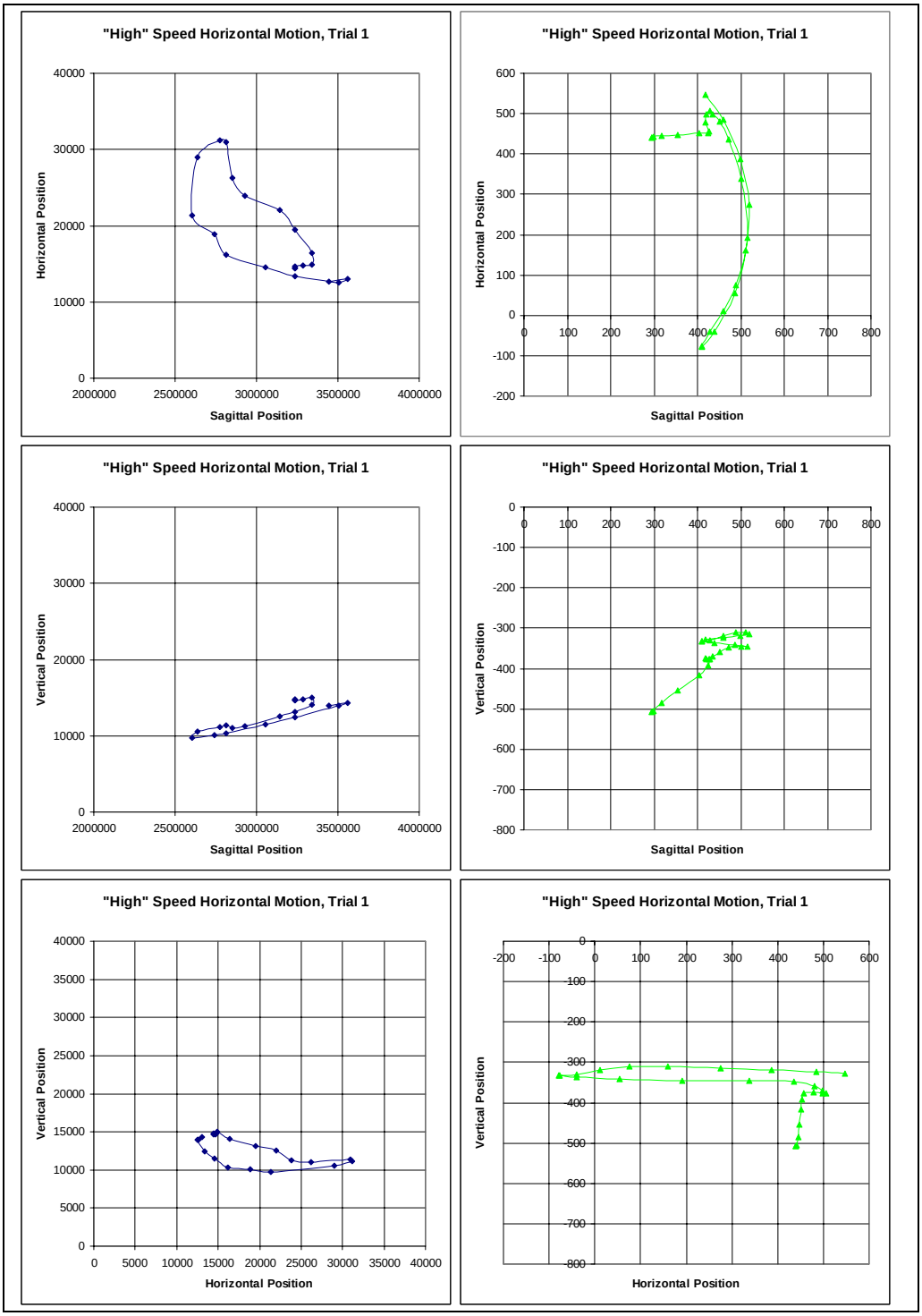
Horizontal motion at "High" speed

Human

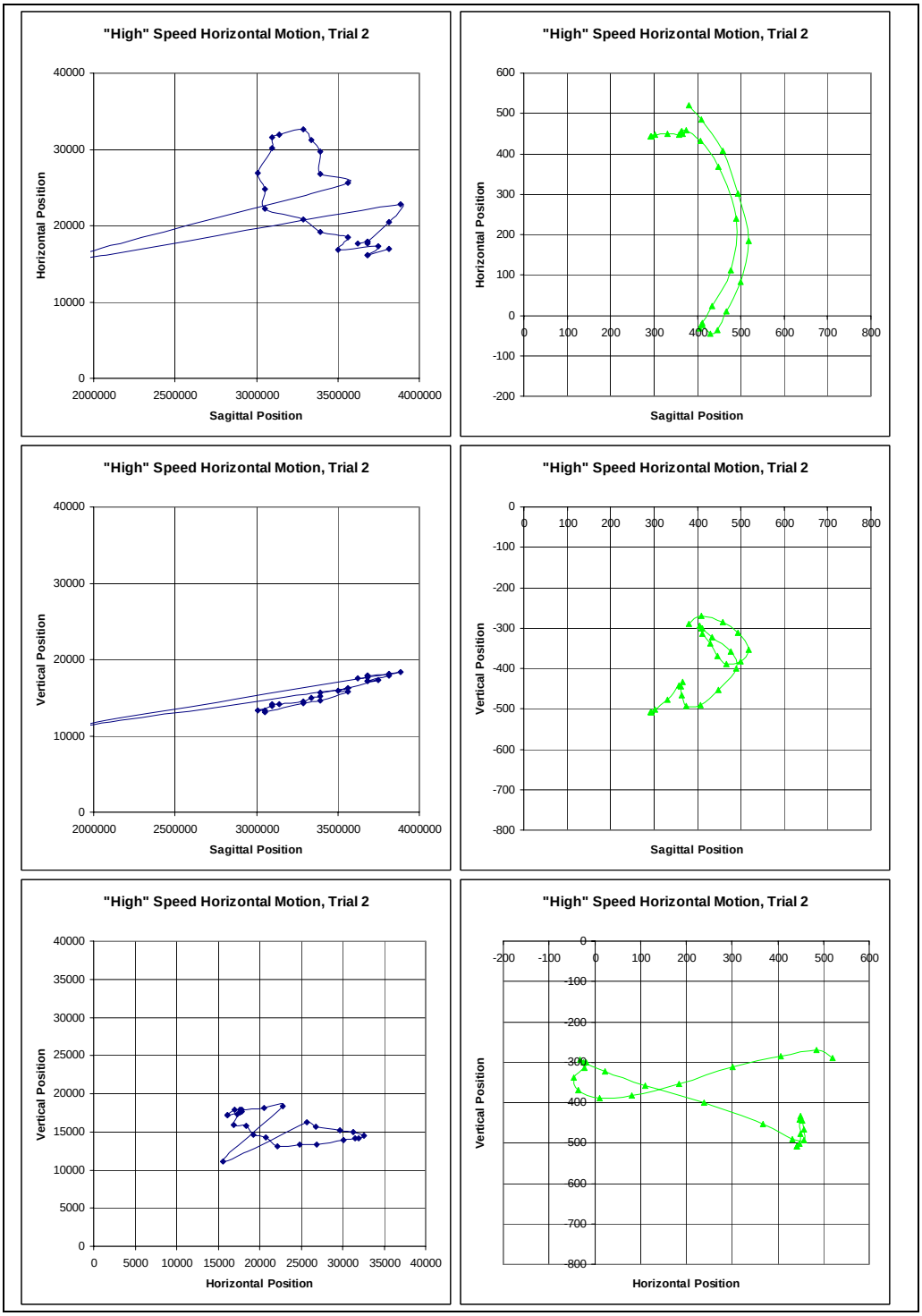
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
3237500	14424.24	14742.42	3621610	17677.97	17559.32	3561250	16800	17850	3815625	17000	18000	3621610	17915.25	16016.95
3237500	14424.24	14636.36	3684052	17862.07	17620.69	3561250	16800	17616.67	3956945	17629.63	18537.04	3684052	18344.83	16051.72
3237500	14636.36	14636.36	3684052	17620.69	17862.07	3684052	17862.07	18224.14	1396569	10751.63	8464.053	3561250	17733.33	15516.67
3287308	14753.85	14753.85	3684052	17862.07	17862.07	3684052	19793.1	17620.69	1443750	11351.35	8750	3621610	17915.25	15779.66
3338672	14875	14984.38	3815625	20500	18125	3621610	22423.73	16728.81	3748684	21000	17070.18	3561250	18200	15283.33
3338672	16406.25	14109.38	3885000	22781.82	18327.27	3502869	25360.66	16180.33	3561250	23800	15516.67	3446371	21225.81	13096.77
3237500	19515.15	13151.52	1842026	15568.97	11103.45	3561250	28933.33	16216.67	3502869	30409.84	14688.52	3502869	24213.12	12852.46
3142280	22029.41	12558.82	3561250	25666.67	16216.67	3391667	32333.33	15555.56	3189179	30194.03	13791.04	3338672	26468.75	11703.13
2927055	23876.71	11219.18	3391667	26777.78	15666.67	3237500	31606.06	15060.61	3096739	30536.23	13492.75	3287308	28538.46	11523.08
2849000	26226.67	11013.33	3391667	29666.67	15222.22	3142280	30676.47	14617.65	3052500	29800	13200	3189179	29567.16	11388.06
2811513	30947.37	11328.95	3338672	31281.25	14984.38	3052500	28600	14200	2967708	27222.22	12833.33	3052500	30000	11300
2775000	31181.82	11090.91	3287308	32630.77	14538.46	3009507	25732.39	13901.41	2967708	24694.45	12638.89	3052500	31400	11500
2637963	28950.62	10543.21	3142280	31911.77	14205.88	3142280	23882.35	14514.71	3052500	22200	13000	2927055	29438.36	10931.51
2605793	21341.46	9731.707	3096739	31550.72	14101.45	3096739	19985.51	14405.8	3237500	20151.52	13681.82	2849000	26413.33	10546.67
2739423	18846.15	10051.28	3096739	30130.44	13898.55	3237500	18030.3	15166.67	3391667	18333.33	14555.56	2927055	24643.84	10835.62
2811513	16210.53	10315.79	3009507	26915.49	13309.86	3391667	16555.55	16333.33	3561250	16800	15633.33	3009507	22774.65	10943.66
3052500	14600	11500	3052500	24800	13300	3621610	16254.24	17677.97	3621610	15779.66	16610.17	3096739	21000	11463.77
3237500	13363.64	12409.09	3052500	22200	13100	3748684	16333.33	18912.28	3621610	15305.08	16966.1	3189179	19537.31	12223.88
3502869	12508.2	14000	3287308	20784.62	14215.38	3815625	16500	19250	3885000	16418.18	18327.27	3391667	18555.55	13444.44
3561250	13066.67	14350	3391667	19222.22	14666.67	3748684	16333.33	18912.28	3815625	16250	18125	3502869	17557.38	14000
3446371	12645.16	14000	3561250	18433.33	15750	3748684	16333.33	18789.47				3446371	15806.45	14677.42
			3502869	16868.85	15950.82							3561250	16100	15283.33
			3748684	17315.79	17315.79							3502869	15950.82	15147.54
			3684052	16172.41	17137.93									
			3684052	16172.41	17137.93									
			3815625	17000	17875									

ISAC

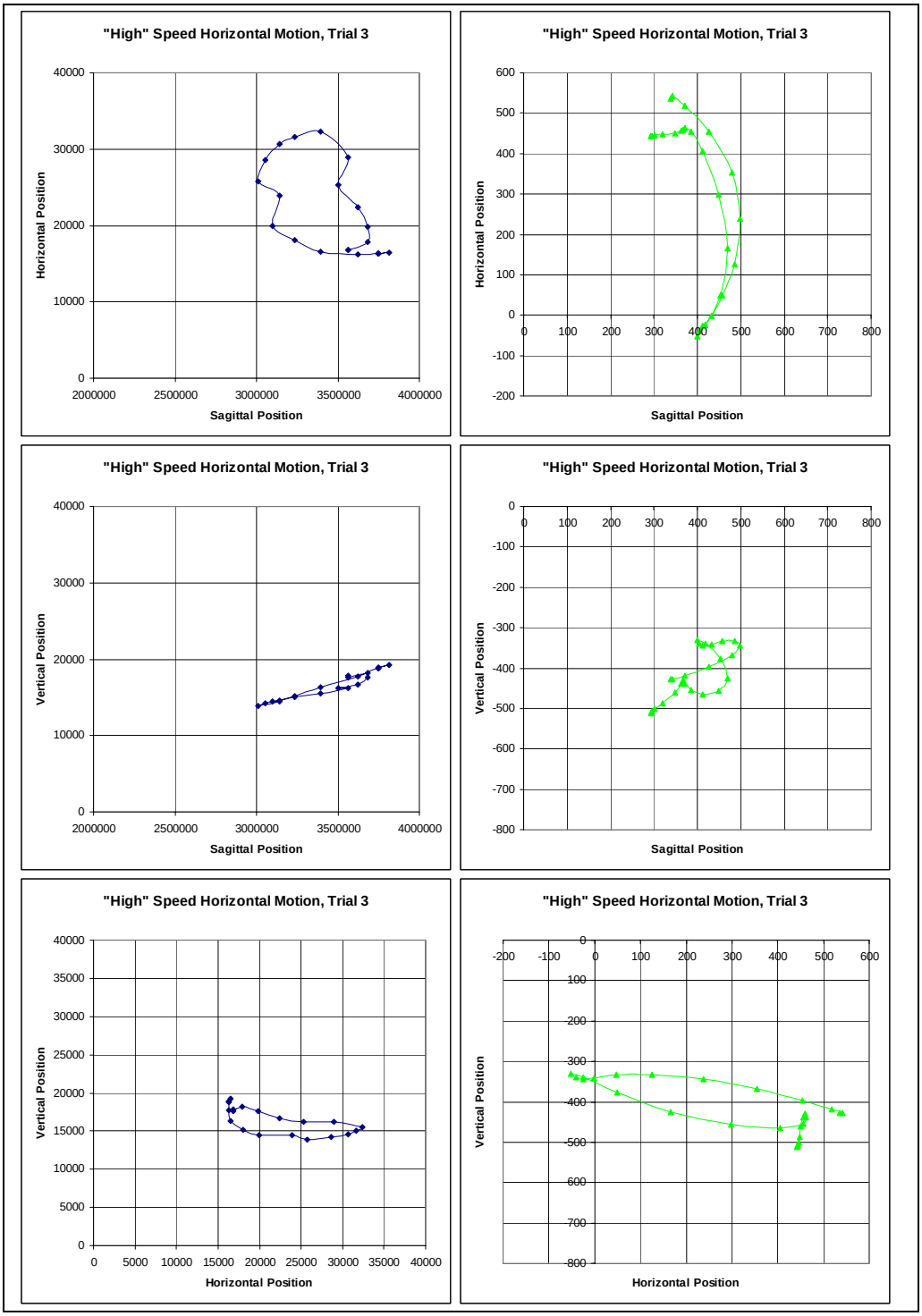
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
295.3118	439.9491	-507.6484	293.2744	442.2134	-509.0515	293.2306	442.7855	-510.3037	292.4727	443.0791	-510.9888	294.1794	440.9999	-507.7412
295.1108	440.0677	-507.7629	293.2082	442.2143	-508.9378	293.3068	442.6703	-510.2354	292.573	443.021	-510.8292	294.3042	440.9978	-507.6269
298.7474	442.0842	-504.6848	293.0115	442.7879	-508.8347	295.5098	444.1459	-507.9863	296.1447	444.6057	-507.5277	294.0254	441.9746	-507.2837
317.2136	444.6306	-486.0894	302.8159	447.7287	-502.0537	302.2416	445.5845	-500.4249	301.2519	444.9949	-500.942	298.4131	445.7069	-503.8168
355.0656	447.6067	-454.0396	331.0538	449.9169	-476.6492	319.3191	448.3268	-486.1262	316.938	449.1234	-489.693	333.4677	452.3927	-481.0703
404.1319	451.6577	-416.5374	358.4872	448.1166	-442.8154	347.9239	450.4601	-459.7353	341.5028	452.6816	-470.1228	385.0177	453.0697	-433.5554
425.3687	452.3748	-391.1619	365.1778	449.381	-433.1847	362.9337	456.4242	-437.6789	375.5076	465.1574	-450.7045	408.8328	451.1572	-395.1466
426.9637	456.9737	-376.7316	361.4379	453.5005	-444.1311	367.4622	459.9378	-430.1631	403.7514	467.3949	-428.3782	414.2598	457.5566	-375.2926
418.5859	478.8992	-374.4301	362.9969	457.2966	-467.2926	371.8215	462.5694	-438.8417	407.428	461.4384	-413.2364	408.9227	465.2732	-378.565
420.8491	498.1297	-376.1312	375.2699	458.3508	-492.412	386.5864	455.4032	-454.2387	415.3399	457.0074	-408.0678	425.7607	475.2793	-396.2065
429.8676	506.4	-375.6504	407.3208	432.3434	-491.2096	412.7333	404.9647	-464.0899	425.4074	438.4248	-404.8747	449.1758	467.0588	-407.5463
435.921	498.7542	-370.3142	449.1126	368.0709	-453.2733	449.5236	298.3188	-456.0416	458.8927	355.2353	-399.0084	467.8574	431.0226	-401.8249
451.2602	480.1306	-358.9497	488.9447	239.9993	-399.6769	469.6443	165.2369	-424.3797	490.9964	228.9969	-381.2212	491.4982	362.6311	-386.6207
472.0306	436.3027	-348.4424	477.1437	110.5564	-357.8498	452.628	49.42985	-376.22	489.0588	91.13836	-348.5786	510.5353	243.7543	-374.2665
502.0518	337.5997	-346.0275	434.943	23.23472	-322.6171	417.6348	-24.28295	-339.3174	454.7458	-2.215742	-322.731	448.7092	-19.8301	-331.5917
514.5821	191.6813	-345.9119	410.0229	-23.80906	-300.4085	399.2498	-51.69074	-330.9894	423.8971	-51.0751	-315.13	409.427	-72.07033	-312.53
487.1317	54.72338	-342.0344	404.3507	-33.02493	-293.8315	404.3361	-40.02541	-338.3654	420.2201	-54.74245	-321.096	404.681	-68.19656	-303.8919
438.6601	-39.67044	-336.1044	410.8844	-18.18173	-301.4177	412.4785	-25.34482	-343.5822	434.9452	-27.69484	-333.8919	419.5077	-42.91034	-303.7301
411.0556	-78.43483	-332.5422	410.4499	-23.33262	-314.0075	432.0166	-1.113045	-340.86	464.1353	26.60021	-340.7981	423.8033	-35.98504	-308.7821
410.4825	-75.86361	-331.6564	430.7316	-45.87291	-337.2062	458.3016	48.16619	-333.4881	492.3237	95.80673	-336.648	430.4142	-33.5738	-311.1861
429.184	-39.77315	-330.2467	445.3126	-35.86296	-368.5099	487.1759	125.4127	-332.906	511.8268	194.68	-325.6576	451.1165	-0.212832	-308.7682
459.9653	10.63243	-319.8774	466.1783	11.03613	-389.2293	497.5914	238.2627	-344.2797	506.9763	299.7605	-315.8733	485.3971	57.7034	-309.4882
488.9898	75.18409	-310.5093	500.1146	81.81605	-381.2254	480.4177	354.1142	-367.909	479.013	401.125	-305.6696	521.5703	169.1344	-322.7414
511.6051	160.3148	-309.2478	518.8143	185.0244	-353.5887	425.8604	454.2383	-396.6376	430.4898	482.1128	-305.0005	523.277	285.2555	-341.3377
518.7674	275.2768	-314.0475	493.7319	302.4177	-312.6819	370.467	519.1705	-418.6487	383.2775	532.7287	-323.1066	498.2406	407.6658	-350.0583
499.5204	387.7601	-319.9739	457.6083	406.7402	-284.6299	341.9811	541.681	-428.5456				460.8728	490.7739	-344.6348
460.3367	483.7214	-323.7031	410.0244	485.6786	-269.8809	338.7833	536.3825	-427.5559						
418.5519	547.4354	-327.057	381.4164	519.53	-289.3592									



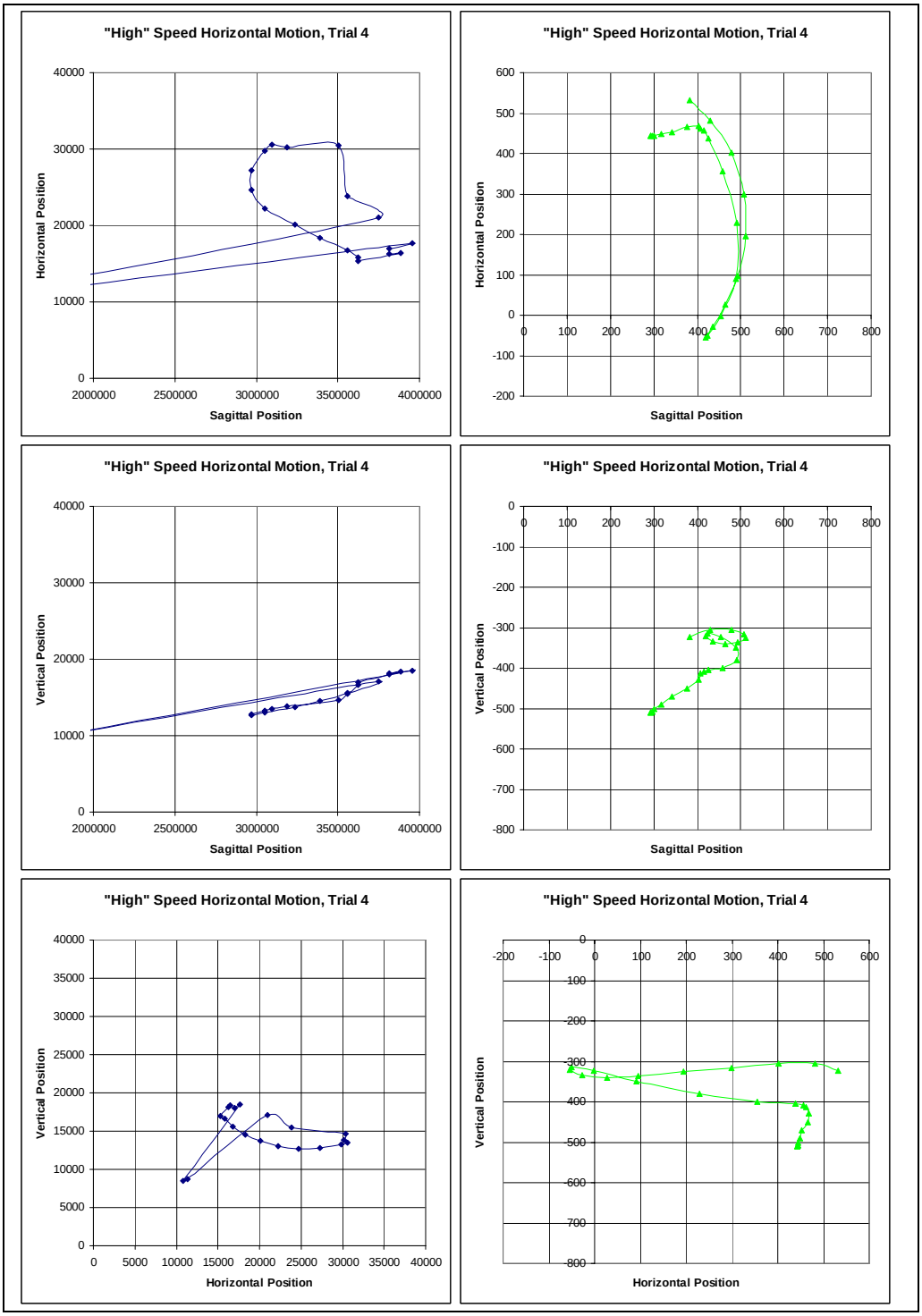
"High" speed horizontal motion, trial 1; human motion is shown on the left, that of ISAC on the right



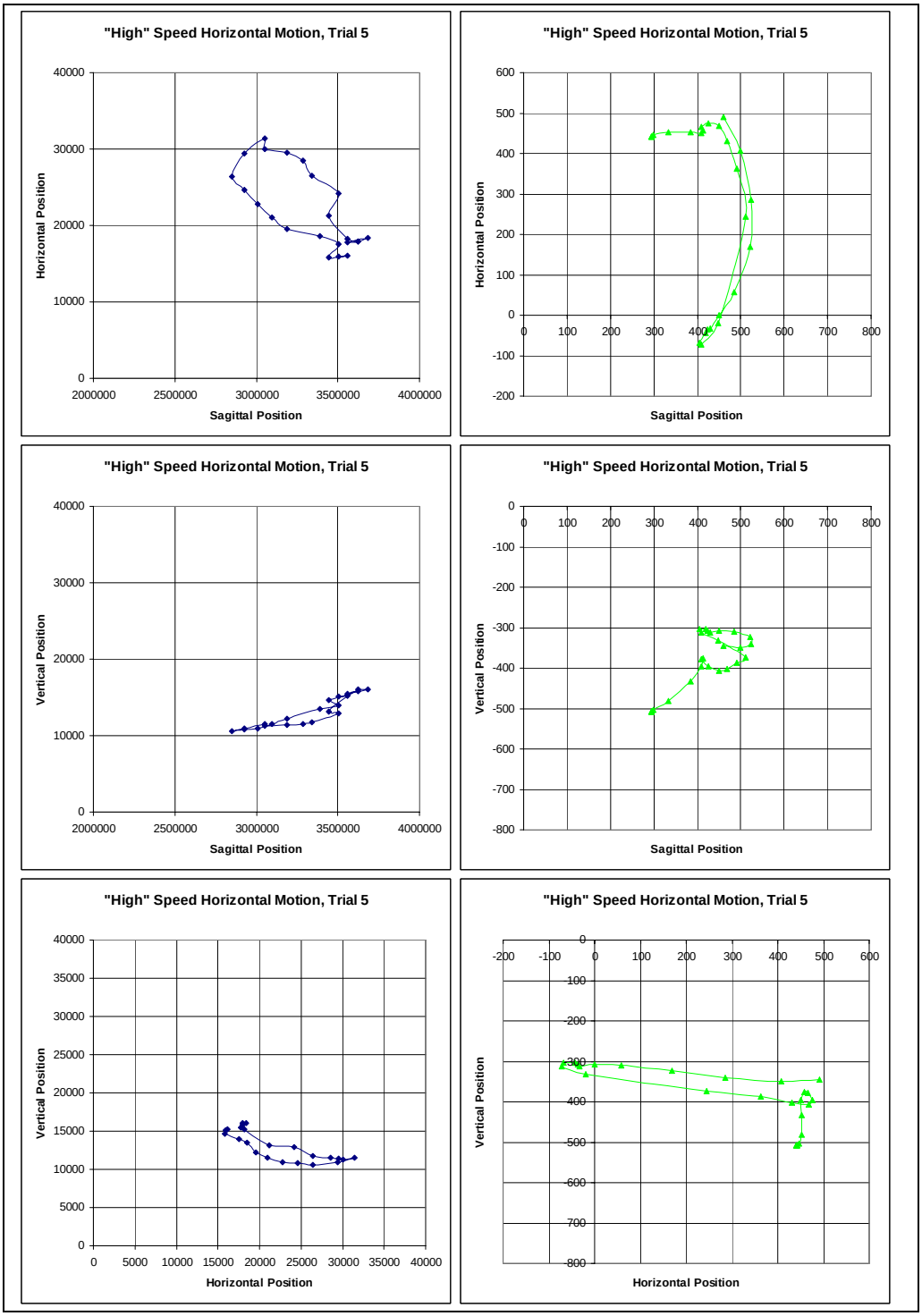
"High" speed horizontal motion, trial 2; human motion is shown on the left, that of ISAC on the right



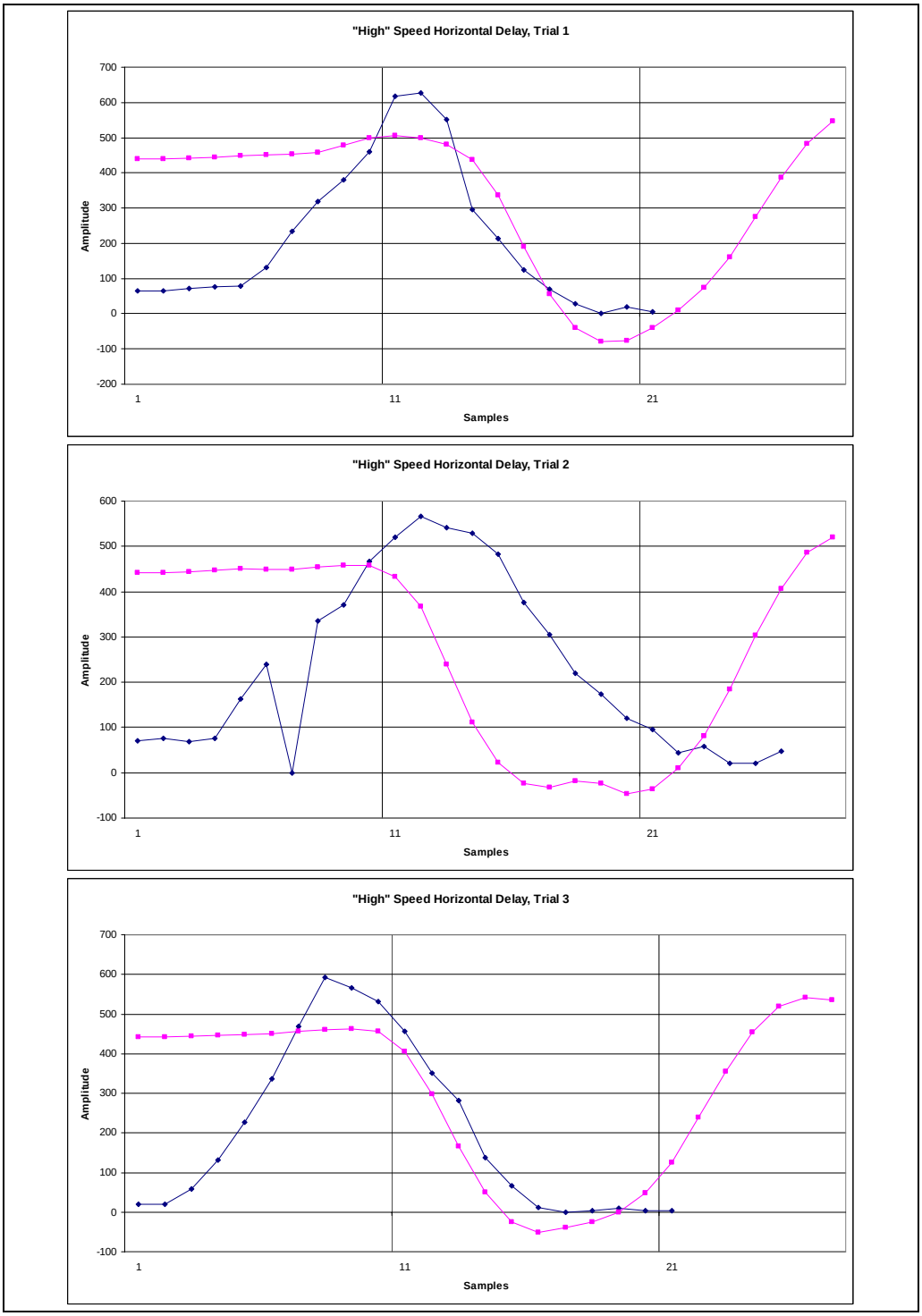
"High" speed horizontal motion, trial 3; human motion is shown on the left, that of ISAC on the right



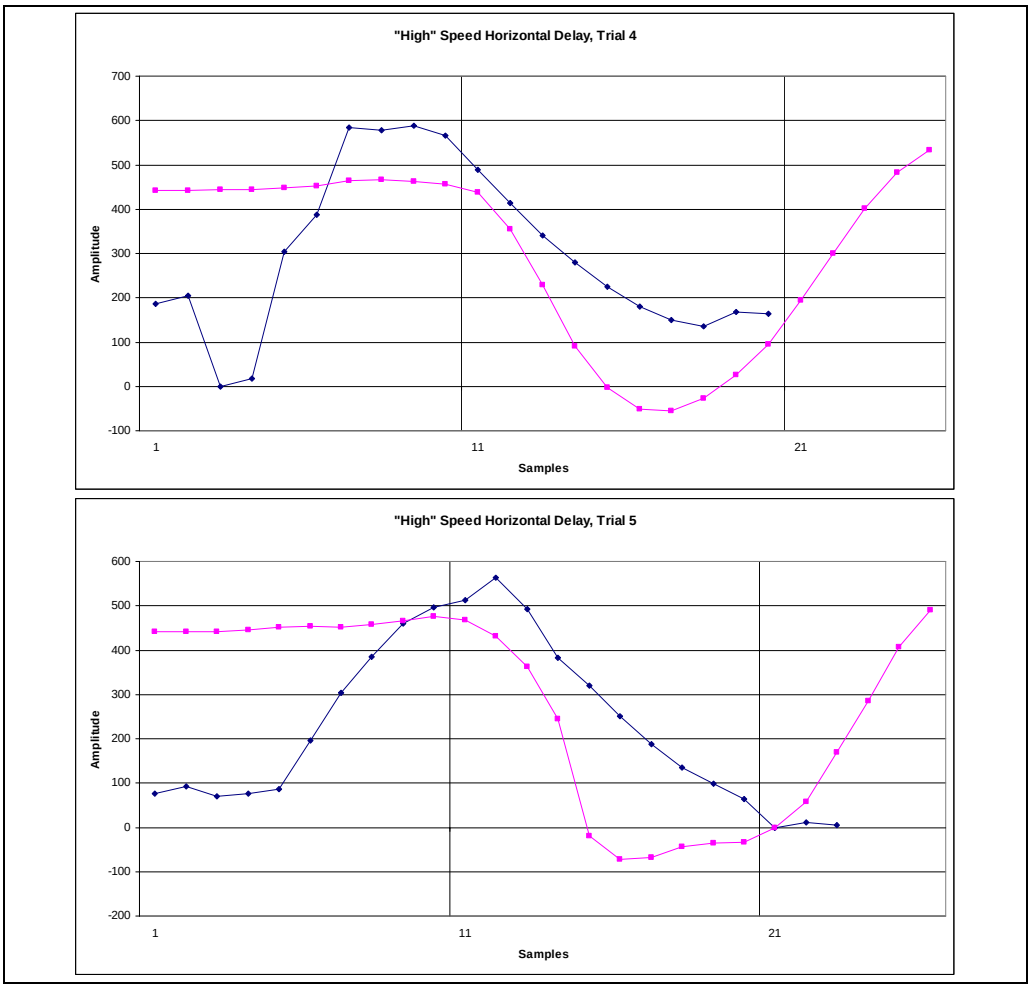
"High" speed horizontal motion, trial 4; human motion is shown on the left, that of ISAC on the right



"High" speed horizontal motion, trial 5; human motion is shown on the left, that of ISAC on the right



"High" speed horizontal delay, trials 1-3; human motion is shown on the on bottom initially, that of ISAC on the on top initially



"High" speed horizontal delay, trials 4-5; human motion is shown on the on bottom initially, that of ISAC on the on top initially

Sagittal motion at "Low" speed

Human

Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
1780625	11316.67	14291.67	1842026	10741.38	14301.72	1978472	11277.78	16592.59	1996963	11710.28	15897.2	1874342	11543.86	14675.44
1751434	11131.15	14057.38	1874342	10807.02	14614.04	1978472	11277.78	16787.04	1978472	11666.67	15685.19	1874342	11543.86	14675.44
1780625	11200	14350	1874342	10807.02	14552.63	1960321	11238.53	16568.81	1978472	11666.67	15685.19	1842026	11344.83	14060.34
1780625	11200	14408.33	1907813	10875	14875	2074515	11757.28	17262.14	1978472	11666.67	15750	1907813	11625	14500
1795588	11235.29	14470.59	1858044	10773.91	14547.83	2054567	11711.54	17096.15	1996963	11710.28	15831.78	1978472	12185.19	15037.04
1810805	11389.83	14652.54	1874342	10807.02	14614.04	2074515	11893.2	16990.29	1978472	11537.04	15685.19	2015802	12283.02	15452.83
1842026	11706.9	14724.14	1907813	10750	14937.5	2115594	12128.71	17188.12	2015802	11886.79	16047.17	2035000	12333.33	15800
1874342	11912.28	14921.05	1907813	10750	15000	2202835	12340.21	17752.58	2035000	11933.33	16200	1996963	12102.8	15373.83
1925000	12297.3	15450.45	1925000	11036.04	15135.13	2273138	12808.51	18542.55	2054567	12115.38	16288.46	1978472	11925.93	15490.74
1942500	12472.73	15527.27	1960321	11366.97	15605.5	2348077	13000	19307.69	2115594	12544.55	16772.28	2035000	12333.33	15800
1996963	12757.01	15831.78	1942500	11327.27	15400	2374167	13533.33	19288.89	2074515	12300.97	16310.68	2035000	12200	15933.33
1996963	12495.33	15700.93	1942500	11327.27	15336.36	2322554	13239.13	18945.65	2094853	12490.2	16470.59	1996963	11971.96	15570.09
2015802	12415.09	15849.06	1960321	11495.41	15348.62	2374167	13377.78	19288.89	2115594	12683.17	16564.36	2074515	12300.97	16242.72
2035000	12200	16066.67	1942500	11327.27	15272.73	2484593	14000	20267.44	2136750	12740	16800	2202835	13350.52	16958.76
2054567	11980.77	16355.77	1942500	11200	15272.73	2400843	13449.44	19426.97	2158333	12656.57	16828.28	2074515	12165.05	16106.8
2094853	11803.92	16470.59	1996963	11317.76	15831.78	2400843	13292.13	19584.27	2180357	12571.43	16857.14	2136750	12740	16520
2158333	12232.32	17252.53	1996963	11317.76	15766.36	2400843	13134.83	19584.27	2158333	12232.32	16757.58	2180357	12714.29	16642.86
2225781	12250	18010.42	1960321	11238.53	15348.62	2428125	12886.36	19727.27	2202835	12051.55	17103.09	2136750	12320	16240
2225781	12104.17	18010.42	1996963	11317.76	15504.67	2484593	12860.46	20023.26	2273138	11617.02	17797.87	2202835	12484.54	16381.44
2273138	12361.7	18393.62	1996963	11317.76	15439.25	2513824	12929.41	20423.53	2428125	12250	18772.73	2180357	12285.71	16142.86
2297581	12569.89	18591.4	2054567	11576.92	15884.62	2484593	12534.88	20186.05	2456035	12632.18	18827.59	2374167	13066.67	17966.67
2348077	12538.46	18846.15	2015802	11226.42	15320.75	2574398	12903.61	20831.33	2428125	12568.18	18534.09	2456035	13114.94	19068.96
2297581	12419.35	18666.67	2136750	11620	15960	2605793	12975.61	21256.1	2404843	12348.31	18325.84	2484593	13674.42	18883.72
2322554	12630.43	18565.22	2115594	11712.87	15801.98	2605793	12975.61	21085.37	2484593	12697.67	18965.12	2428125	12886.36	19090.91
2456035	13275.86	19873.56	2158333	11808.08	16121.21	2574398	12734.94	20831.33	2456035	12471.26	18988.51	2400843	12348.31	19269.66
2484593	13348.84	20511.63	2180357	12000	16571.43	2670938	13125	21612.5	2484593	12697.67	19290.7	2428125	12727.27	19011.36
2543750	13500	20583.33	2158333	11949.5	16121.21	2637963	13049.38	21432.1	2484593	12534.88	19209.3	2513824	12929.41	20176.47
2543750	13500	20750	2225781	12104.17	17135.42	2704747	13025.32	22151.9	2513824	12435.29	19682.35	2513824	13094.12	19847.06
2574398	13746.99	21000	2202835	11907.22	17175.26	2670938	12950	21787.5	2574398	12566.26	20072.29	2456035	12793.1	19310.35
2574398	13746.99	21000	2225781	12104.17	17354.17	2637963	12876.54	21345.68	2513824	12435.29	19600	2574398	13409.64	20240.96
2849000	14466.67	30426.67	2249211	12305.26	17610.53	2739423	12923.08	22525.64	2605793	12804.88	20231.71	2605793	13487.8	20829.27
2574398	13746.99	20915.66	2202835	12051.55	17247.42	2775000	13000	22545.46	2574398	12734.94	19819.28	2704747	13911.39	21620.25
2637963	13913.58	21777.78	2322554	12478.26	18489.13	2739423	13102.56	22346.15	2811513	13447.37	21736.84	2637963	13740.74	21000
2605793	13829.27	21426.83	2297581	12268.82	18139.79	2775000	13000	22909.09	2775000	13363.64	21545.46	2637963	13567.9	21172.84
2637963	13913.58	21864.2	2273138	12212.77	17946.81	2811513	13263.16	23026.32	2637963	13049.38	20308.64	2605793	13487.8	20914.63
2605793	13829.27	21426.83	2322554	12630.43	18336.96	2739423	12923.08	22346.15	2775000	13545.45	21454.54	5775000	38027.03	49378.38
2670938	14350	21962.5	2374167	12911.11	18744.45	2811513	13078.95	23026.32	2775000	13545.45	21545.46	2605793	13829.27	20573.17
2775000	14818.18	22818.18	2322554	12630.43	18260.87	2811513	13078.95	23026.32	2887500	14189.19	22418.92	2574398	13578.31	20493.98
2775000	14818.18	22909.09	2322554	12630.43	18336.96	2849000	12973.33	23240	2811513	13815.79	22013.16	2670938	14000	21437.5
2670938	14525	21700	2348077	12846.15	18538.46	2927055	13328.77	24068.49	2811513	13631.58	22105.26	2704747	14088.61	21531.65
2670938	14350	21962.5	2322554	12630.43	18260.87	2887500	13243.24	23648.65	2811513	13631.58	22105.26	2811513	14552.63	22842.11
2775000	14454.55	23272.73	2400843	12977.53	19191.01	2927055	13136.99	24068.49	2775000	13545.45	21818.18	2775000	14272.73	22818.18
2670938	14175	21962.5	2428125	13045.45	19329.54	2967708	13416.67	24208.33	2739423	13461.54	21448.72	2775000	14454.55	22363.64
2739423	14358.97	23153.85	2374167	12600	18977.78	3009507	13507.04	24549.29	2739423	13461.54	21448.72	2704747	14088.61	21620.25
2811513	14552.63	23947.37	2428125	12886.36	19250	3142280	14000	25941.18	2775000	13545.45	21636.36	2704747	13911.39	21797.47
2775000	14272.73	23454.54	2428125	12886.36	19329.54	3096739	13898.55	25463.77	2739423	13461.54	21269.23	2775000	14090.91	22636.36
2775000	14272.73	23545.46	2428125	12568.18	19488.64	3142280	13588.24	25735.29	2849000	14093.33	22120	2775000	14090.91	22636.36
2775000	14272.73	23454.54	2400843	12191.01	19505.62	3052500	13400	24600	2811513	14000	21644.74	2811513	14368.42	22842.11
2887500	14567.57	24594.59	2484593	12697.67	20267.44	3237500	14000	26409.09	2849000	14093.33	21933.33	5087500	20333.33	43500
2849000	14466.67	24266.67	2400843	12191.01	19348.31	3142280	13382.35	25735.29	2704747	13202.53	21000	2739423	13820.51	22435.9
2811513	14184.21	23763.16	2484593	12697.67	19860.46	3096739	13086.96	25768.12	2811513	13631.58	21828.95	2775000	13909.09	22818.18
3009507	15084.51	25535.21	2513824	12764.71	20176.47	3096739	13289.86	25362.32	2887500	14189.19	22324.32	2775000	13909.09	23000
3096739	15521.74	26275.36	2484593	12697.67	20104.65	3189179	13477.61	26537.31	2887500	14189.19	22229.73	2811513	14000	23118.42
3009507	15281.69	25535.21	2574398	13240.96	20915.66	3338672	14437.5	27562.5	2849000	14280	21933.33	2887500	14189.19	24121.62
2967708	15166.67	24888.89	2543750	13000	20666.67	3338672	14437.5	27671.88	2927055	14671.23	22534.25	2887500	14189.19	24027.03
3009507	15281.69	25436.62	2574398	13072.29	21084.34	3287308	14107.69	27246.15	2887500	14189.19	22324.32	2849000	14093.33	23800
2967708	15555.56	24597.22	2543750	13000	20916.67	3338672	14437.5	27671.88	3052500	14600	23600	2775000	13727.27	22727.27
3142280	15852.94	26558.82	2605793	13487.8	21085.37	3391667	14333.33	28000	3052500	14600	23400	2887500	14189.19	23648.65
3142280	15647.06	26661.77	2543750	13333.33	20583.33	3446371	14677.42	28225.81	3052500	14600	23300	2927055	14287.67	23972.6
3142280	15441.18	26970.59	2605793	13658.54	21085.37	3391667	14555.56	27888.89	3142280	14411.76	24191.18	2927055	14287.67	24164.38
3287308	15830.77	27892.31	2605793	13487.8	21000	3338672	14218.75	27671.88	3052500	14200	23400	2927055	14287.67	23876.71
3287308	15615.38	27784.62	2605793	13658.54	21000	3287308	13892.31	27246.15	3142280	14411.76	23779.41	2927055	14479.45	23972.6
3237500	15272.73	27787.88	2670938	14000	21612.5	3391667	14333.33	28222.22	3096739	14507.25	23434.78	3052500	15000	25000
3237500	15060.61	27045.46	2605793	13829.27	20829.27	3391667	14333.33	28333.33	3189179	14313.43	24447.76	2927055	14671.23	24068.49
3287308	14969.23	27784.62	2670938	14000	21612.5	3391667	14333.33	28111.11	3096739	13898.55	23637.68	3052500	15200	25000
3287308	14969.23	28000	2739423	14358.97	21987.18	3446371	14677.42	28677.42	3237500	14000	24818.18	3052500	15400	25000
3391667	15222.22	29000	2704747	14088.61	21974.68	3446371	14903.23	28677.42	3142280	13794.12	23882.35	3052500	14800	25400
3446371	15354.84	29354.84	2775000	14090.91	22454.54	3502869	15032.79	29147.54	3142280	13794.12	23985.29	3052500	14800	25500
3391667	15000	28777.78	2739423	14179.49	22346.15	3502869	15032.79	29147.54	3237500	14000	24606.06	3142280	15235.29	26250
3391667	15000	28777.78	2775000	14272.73	22545.46	3502869	15032.79	29147.54	2775000	13000	29545.46	3237500	15909.09	27257.58
3391667	15000	28666.67	2811513	14368.42	22934.21	3502869	15032.79	29147.54	3237500	14212.12	24181.82	3237500	15696.97	27045.46
3391667	15000	28777.78	2811513	14368.42	22842.11	3446371	14451.61	28564.52	3237500	14000	24500	3189179	15358.21	26850.75
3338672	14875	28109.38	2811513	14184.21	22750	3391667	14333.33	28222.22	3237500	14424.24	24393.94	3287308	15830.77	27569.23
3237500	14636.36	27363.64	2927055	14287.67	23876.71	3391667	14333.33	28222.22	2927055	13712.33	30972.6	3287308	16046.15	27569.23
3189179	14522.39	27268.66	2927055	14095.89	23589.04	3446371	14677.42	28451.61	3009507	13901.41	32140.85	3237500	15909.09	26833.33
3142280	14411.76	26764.71	3009507	13901.41	24253.52	3391667	14333.33	27888.89	3338672	14437.5	25156.25	3237500	15909.09	26727.27
3052500	13800	26300	3052500	14000	24500	3391667	14333.33	28000	2927055	13328.77	31164.38	3237500	15909.09	27045.46
3142280	14411.76	27279.41	3052500	13800	24600	3237500	13575.76	26727.27	3009507	13901.41	31845.07	3237500	16121.21	26833.33
3052500	14000	26700	3096739	13695.65	24855.07	3287308	13461.54	27569.23	2927055	13328.77	31068.49	3287308	16046.15	27353.85
2967708	13611.11	26055.55	3052500	13400	24700	3237500	13787.88	26833.33	2927055	13328.77	31068.49	3338672	16187.5	27890.63
3009507	13901.41	26521.13	3052500	13200	24700	3189179	13477.61	26432.84	2887500	13243.24	30743.24	3338672	16187.5	27781.25
3009507	13901.41	26718.31	3052500	13200	24500	3189179	13477.61	26328.36	2927055	13136.99	31164.38	3391667	16111.11	28111.11
2849000	13160	25200	2967708	12833.33	23819.45	3142280	13588.24	26044.12	2967708	13222.22	31597.22	3338672	15531.25	27562.5
2927055	13712.33	25986.3	3237500	13575.76	25984.85	3142280	13794.12	26147.06	2967708	13222.22	31597.22	3287308	15400	27030.77
2927055	13712.33	26369.86	3189179	13268.66	25910.45	3189179	14104.48	26328.36	2967708	13222.22	31597.22	3391667	15000	27777.78
2849000	13346.67	25946.67	3237500	13575.76	26090.91	3142280	14000	26044.12	3391667	13888.89	25555.55	3338672	15093.75	27125
2739423	12923.08	24769.23	3189179	12850.75	25597.02	3009507	13309.86	24943.66	3391667	13666.67	25666.67	3446371	15580.65	28225.81

Human

Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
2775000	12818.18	25454.54	3446371	14000	27887.1	2887500	12864.87	24121.62	3502869	14344.26	26508.2	3446371	15580.65	28225.81
2775000	12818.18	25454.54	3446371	13774.19	27548.39	2927055	12945.21	24643.84	3391667	14111.11	25555.55	3287308	14538.46	26600
2637963	12185.19	24283.95	3391667	13444.44	27444.45	2887500	12864.87	24405.41	3391667	13888.89	25777.78	3237500	14424.24	26196.97
2637963	12012.35	24370.37	3391667	13666.67	27444.45	2775000	12454.55	23272.73	2887500	13054.05	30648.65	3237500	14424.24	25984.85
2574398	11554.22	23445.78	3391667	13444.44	27444.45	2927055	12945.21	24739.73	2849000	12973.33	30520	3287308	14538.46	26492.31
2543750	11500	23333.33	3446371	13774.19	28000	2670938	12425	22137.5	3287308	13676.92	25092.31	3189179	14104.48	25805.97
2543750	11333.33	23166.67	3502869	13655.74	28688.53	2739423	12743.59	22435.9	3189179	13268.66	24343.28	3189179	13895.52	25910.45
2400843	10460.67	21786.52	3009507	12126.76	24253.52	2605793	12463.42	21170.73	3189179	13268.66	24343.28	3189179	14104.48	25910.45
2322554	10347.83	20619.56	3009507	12323.94	24253.52	2637963	12530.86	21518.52	3287308	13461.54	25630.77	3142280	13794.12	25632.35
2225781	10791.67	19833.33	2967708	12250	23722.22	2428125	12090.91	19727.27	3142280	13382.35	24191.18	3096739	13695.65	25260.87
2225781	11229.17	19541.67	2967708	12250	24111.11	2374167	11822.22	19522.22	3096739	13289.86	23739.13	3009507	13112.68	24746.48
2273138	11170.21	26585.11	2637963	11320.99	21950.62	2374167	12133.33	18977.78	3142280	13588.24	24191.18	3096739	13695.65	25159.42
2074515	10805.83	19029.13	2637963	11320.99	22037.04	2348077	11923.08	18846.15	3142280	13382.35	24294.12	3096739	13695.65	25159.42
1978472	10240.74	18731.48	2484593	11232.56	20755.81	2428125	12250	19488.64	3142280	13588.24	24705.88	2927055	13328.77	23684.93
1978472	10240.74	18666.67	2456035	11505.75	20436.78	2374167	12133.33	19133.33	3189179	13895.52	25074.63	2927055	13136.99	23876.71
1890929	9973.451	17716.81	2348077	11307.69	19769.23	2348077	12076.92	18923.08	3096739	13492.75	24246.38	2849000	12973.33	23146.67
1907813	10000	17875	2348077	11153.85	20230.77	2322554	12021.74	18565.22	3096739	13695.65	24043.48	2927055	13328.77	23972.6
1890929	9973.451	17716.81	2180357	11000	18571.43	2348077	12076.92	19153.85	3052500	13400	24200	2887500	13243.24	23554.05
1858044	10043.48	17408.7	2180357	11142.86	18571.43	2322554	12021.74	18793.48	3009507	13112.68	23957.75	2887500	13054.05	23648.65
1874342	10070.18	17622.81	2158333	11101.01	18313.13	2322554	12021.74	18793.48	3009507	13507.04	23859.15	2811513	12894.74	23026.32
			2158333	11101.01	18242.42	2273138	11765.96	18170.21	2887500	13054.05	23081.08	2637963	12530.86	21172.84
			2180357	11142.86	18285.71	2348077	12076.92	19000	2887500	13243.24	22891.89	2811513	12710.53	23118.42
			2136750	11060	18130	2202835	11185.57	18257.73	2811513	12894.74	22842.11	2849000	13160	23426.67
						2249211	11568.42	18642.11	2775000	12636.36	22545.46	2775000	12818.18	23000
						2180357	11142.86	18214.29	2739423	12743.59	21807.69	2739423	12743.59	22705.13
						2225781	11375	18593.75	2670938	12775	21175	2670938	12600	21875
						2180357	11285.71	18071.43	2605793	12292.68	21256.1	2637963	12358.02	21864.2
						2180357	11142.86	18000	2605793	12463.42	21170.73	2605793	12121.95	21682.93
						2094853	10843.14	17294.12	2637963	13395.06	21777.78	2704747	12493.67	22506.33
						2115594	11019.8	17603.96	2574398	13240.96	21337.35	2637963	12012.35	22123.46
						2115594	11019.8	17603.96	2543750	13166.67	21000	2456035	11505.75	20275.86
						2136750	11060	17920	2484593	13023.26	20267.44	2605793	12121.95	21853.66
						2136750	11060	18200	2484593	13023.26	20186.05	2574398	12060.24	21506.02
						2094853	10980.39	17843.14	2456035	12793.1	20034.48	2543750	11833.33	21416.67
						2054567	10634.62	17500	2484593	13023.26	20837.21	2513824	11611.76	21164.71
						2015802	10301.89	17301.89	2605793	14341.46	21512.2	2513824	11941.18	21164.71
						2035000	10333.33	17466.67	2400843	13134.83	20213.48	2484593	11558.14	21325.58
						2035000	10333.33	17600	2322554	12630.43	19478.26	2513824	11776.47	21658.82
									2374167	12911.11	20144.45	2428125	11613.64	20761.36
									2297581	12720.43	19419.36	2456035	11505.75	21321.84
									2249211	12305.26	18789.47	2428125	11454.55	21159.09
									2249211	12305.26	18789.47	2428125	11613.64	21397.73
									2180357	11714.29	18357.14	2484593	12046.51	22058.14
									2158333	11666.67	18171.72	2374167	11511.11	21077.78
									2074515	10941.75	17669.9	2348077	11461.54	20846.15
									2094853	11254.9	17911.77	2400843	11719.1	21314.61
									2074515	11213.59	17737.86	2322554	11413.04	20619.56
									2015802	10962.26	17169.81	2322554	11413.04	20619.56
									2035000	11133.33	17466.67	2297581	11365.59	20623.66
									1996963	10925.23	17074.77	2322554	11413.04	20695.65
									2035000	11133.33	17400	2249211	11126.32	20189.47
									2035000	11133.33	17266.67	2225781	11083.33	19979.17
									2015802	11094.34	17103.77	2249211	11273.68	20189.47
									2035000	11133.33	17466.67	2225781	11083.33	19906.25
									2035000	11133.33	17666.67	2249211	11273.68	20263.16
									1942500	10563.64	16927.27	2202835	11041.24	19917.53
									1874342	10438.6	16701.75	2202835	11041.24	19917.53
									1890929	10469.03	16477.88	2180357	11000	19857.14
									1925000	10783.78	16648.65	2158333	10676.77	19727.27
									1874342	10561.4	16026.32	2202835	10752.58	20422.68
									1907813	10875	16250	2136750	10500	20020
									1890929	10840.71	15858.41	2180357	10714.29	20428.57
									1907813	10750	16250	2136750	10500	19950
									1874342	10561.4	15842.11	2054567	9961.538	19317.31
									1874342	10561.4	15842.11	2015802	10830.19	18358.49
									1907813	10750	16250	1996963	10794.39	18317.76
									1842026	10379.31	15870.69	1996963	10794.39	18383.18
									1826282	10230.77	15735.04	1960321	10596.33	17981.65
												1996963	10794.39	18383.18
												1978472	10759.26	18212.96
												1942500	10563.64	17754.54
												1978472	10759.26	18342.59
												1925000	10531.53	17720.72
												1890929	10097.34	17592.92
												1874342	10561.4	16947.37
												1874342	10561.4	16885.96
												1810805	10322.03	16194.92
												3502869	14114.75	27196.72
												1858044	11139.13	16678.26

Human

Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z

ISAC

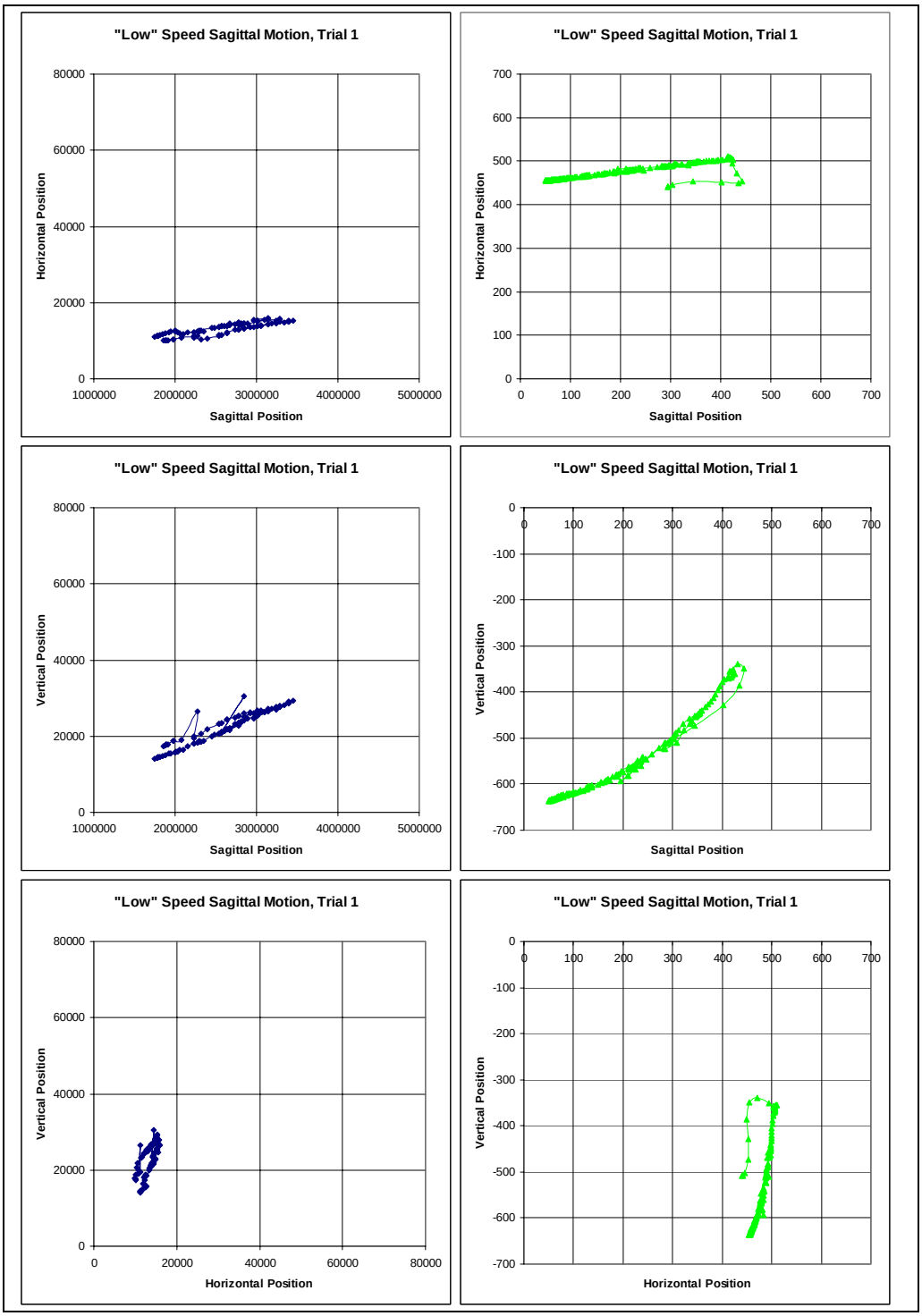
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
294.7261	440.59	-508.7428	293.8089	441.6917	-508.8801	293.5181	442.3818	-507.9909	294.6304	441.7943	-506.5179	292.0351	444.2249	-512.8567
294.7246	440.59	-508.6744	293.8716	441.6908	-508.823	293.6192	442.3233	-507.9337	294.6926	441.7934	-506.4605	292.0351	444.2249	-512.8567
294.8054	441.2184	-508.2848	293.8243	442.3207	-508.5139	293.8385	442.5496	-507.5561	297.6601	443.3177	-504.1318	297.3249	446.5846	-508.4682
303.0658	445.286	-503.0231	301.507	446.7115	-503.6171	296.3864	445.9418	-505.1251	312.6673	445.018	-488.7026	310.6408	447.4671	-494.3654
345.4532	453.1706	-473.1771	351.2226	454.6184	-471.1512	326.5998	453.4321	-487.2523	342.3839	447.4145	-461.7784	353.8248	453.5545	-459.4057
402.0831	452.2659	-428.5175	403.0138	452.2803	-427.4831	372.5328	453.2853	-445.0299	385.6831	451.095	-429.0968	398.9687	455.2263	-421.3037
435.1174	449.941	-385.3285	432.0235	448.8907	-379.4786	392.6284	451.7304	-409.7532	402.6704	454.0315	-404.6507	425.63	457.1937	-387.3079
443.0969	454.2189	-348.1268	432.4607	457.1141	-353.6268	394.4589	459.2136	-394.7463	402.2735	459.4732	-398.9479	431.5532	460.2734	-368.5709
431.6907	471.7364	-339.5604	422.2553	476.2472	-353.2008	382.6019	474.8647	-405.4272	398.3195	477.5805	-403.2184	427.4057	473.8257	-362.9033
422.8378	494.0198	-350.1006	417.0426	492.6802	-362.7799	380.8338	488.8904	-423.8512	394.1576	489.6052	-410.5848	425.0714	489.8354	-365.3689
416.4885	506.0883	-357.7427	412.1443	505.1294	-366.3037	379.0198	496.799	-440.0324	393.9962	498.4887	-415.6897	425.111	500.1663	-370.5885
414.8192	510.3029	-355.273	417.9102	507.4481	-365.4144	381.5108	497.0406	-438.5591	396.7926	498.7651	-412.282	427.5118	506.5341	-370.2415
418.3178	508.9398	-354.1404	420.6695	505.3263	-364.18	384.4649	497.259	-430.0585	399.6393	499.0469	-406.1427	431.0966	506.7508	-361.4096
422.4731	504.2245	-357.4783	426.5585	501.3363	-360.5413	390.1064	495.2531	-422.9894	404.9298	496.7255	-398.6193	435.5043	506.6127	-348.4993
425.274	503.1547	-361.142	427.9985	501.1562	-356.395	388.1226	494.7635	-411.7567	404.6903	496.4399	-391.4904	436.5798	503.5211	-341.0462
422.2446	506.982	-360.4563	427.0568	504.3835	-353.9455	386.5649	496.3785	-398.3966	401.3154	498.2403	-385.0588	434.6875	501.8104	-338.9474
419.9337	507.4665	-364.4279	426.4667	504.758	-352.1869	385.384	496.8032	-394.4871	399.0184	500.5803	-382.7897	430.4974	504.292	-335.7947
418.3935	507.4274	-367.7846	426.217	504.8152	-351.1524	384.8983	496.987	-398.002	397.7411	501.854	-383.6841	429.858	504.6638	-334.1197
413.261	505.8143	-369.189	425.4543	504.894	-351.8344	384.952	497.8962	-398.4441	397.5809	501.7473	-384.4354	426.7473	504.8862	-334.9177
413.0935	504.4676	-369.524	425.1439	504.6716	-350.5985	385.2764	497.9385	-397.7927	395.8783	501.5164	-388.0662	424.6811	504.6097	-339.0258
411.8528	504.2976	-368.9871	425.734	504.2985	-349.0901	384.027	499.2519	-398.9822	394.7396	501.784	-391.4573	424.2386	504.5505	-340.6712
403.4276	502.4541	-372.5628	425.2051	504.2282	-349.5986	384.3136	500.2773	-401.9262	394.5905	501.848	-391.5594	423.7287	504.4823	-339.4784
400.2267	502.7047	-377.7431	423.9264	504.6889	-349.949	382.5098	499.7854	-406.913	394.3271	501.6433	-392.2791	419.6597	502.0665	-343.1219
395.2458	502.1913	-386.9053	422.7069	506.3253	-350.1157	379.9412	499.4366	-412.473	393.5659	501.5395	-393.8326	415.7789	502.6228	-353.2277
391.1101	501.5393	-396.5005	422.803	506.5189	-349.2743	377.39	499.2519	-417.9936	392.5188	501.3966	-395.9658	414.2475	503.6545	-358.4195
385.9265	500.4973	-406.1977	422.6789	506.5017	-349.6315	375.7285	499.8322	-420.3214	389.7372	501.0172	-399.0887	412.9202	504.0035	-358.3609
381.8342	500.4303	-413.7653	422.5268	506.6608	-349.8418	375.5143	501.095	-419.5603	386.0121	500.509	-405.5739	408.6566	508.1494	-360.8503
376.4512	500.3365	-419.8233	423.0786	507.0083	-347.9395	374.8274	501.8866	-419.2755	382.9673	500.5862	-410.6999	406.4261	508.1717	-365.8714
371.3877	500.0313	-426.0693	423.9803	507.1342	-344.7324	372.9381	501.8568	-421.3743	381.1038	501.6395	-412.7974	405.7455	507.8093	-369.1922
366.3451	499.4822	-432.3607	425.1374	507.2958	-340.7435	366.868	500.268	-429.7804	378.5919	502.6722	-413.8717	405.2888	508.0038	-369.8186
358.5532	498.0769	-440.7497	425.0262	507.2803	-338.26	360.6335	499.2243	-441.1286	376.3282	502.7519	-416.8422	405.207	507.9917	-370.1417
354.6195	498.7539	-445.7013	424.0847	507.1488	-337.0113	353.8347	499.8692	-448.9757	375.3105	502.5231	-418.4908	405.3403	507.9241	-369.8186
352.986	498.4438	-447.6574	421.9883	506.8561	-339.1867	349.9031	499.5224	-454.4716	373.7501	502.2968	-418.1781	405.7415	506.5879	-370.671
352.3459	498.1235	-447.522	419.3752	506.4913	-345.7368	344.6481	498.1511	-459.7597	371.6397	501.9907	-419.7951	404.7726	505.8401	-374.1893
351.5134	497.8529	-448.3669	416.4434	506.082	-354.198	342.0639	497.7758	-462.4877	362.0393	499.9717	-427.2225	404.8236	505.7606	-374.1893
350.3957	497.6944	-449.6101	415.2862	505.9204	-358.0677	336.4091	496.9545	-465.095	356.752	499.8316	-435.7862	405.8022	504.5963	-373.6734
348.2085	497.4596	-450.8665	415.2083	505.9095	-357.1745	333.6485	496.6263	-468.0583	353.9039	500.2636	-440.1365	405.9224	504.6131	-373.265
347.1266	497.2308	-452.8174	414.6074	505.8256	-357.1688	332.1292	496.4054	-470.069	354.2173	500.6177	-439.1987	406.3365	504.6709	-372.3827
345.3951	496.9852	-451.6362	413.8921	505.7258	-357.7277	329.1224	495.6809	-472.364	355.0588	499.817	-438.8177	405.1868	504.5104	-371.9353
334.0992	492.7774	-457.5172	412.8095	505.5746	-359.5803	328.0766	495.5297	-473.7115	354.0689	497.9856	-441.4991	402.8877	503.7583	-375.5493
321.9153	493.515	-469.874	411.1133	505.3378	-362.8114	328.7848	495.6321	-472.7221	348.798	497.5434	-449.1469	396.982	502.9397	-383.9173
311.7896	492.2193	-482.7391	410.1125	505.7244	-365.0358	328.8748	495.6451	-472.4104	340.4075	497.6834	-459.6163	384.7859	500.3418	-401.0939
306.3127	491.8449	-489.4774	409.3664	506.6717	-363.4359	328.702	495.6201	-472.5554	335.815	497.2341	-466.7635	379.6766	500.9487	-418.7016
305.5287	491.532	-490.6068	401.1056	505.2303	-365.5166	326.3046	494.6331	-474.2833	333.9984	496.9685	-469.3895	376.3766	501.298	-425.9027
302.7047	489.2769	-494.4981	397.059	505.4211	-373.0242	325.2499	494.6243	-475.3423	333.5304	495.9554	-468.1292	375.0494	502.1613	-424.8976
299.2088	488.0808	-499.8063	392.3349	505.5837	-383.2362	319.5055	493.8015	-478.4234	329	494.3739	-471.3887	372.0192	501.7243	-422.9001
284.8628	487.3983	-510.909	389.9764	505.3224	-390.6058	317.442	493.8528	-482.4325	323.4456	494.2953	-477.5165	366.8586	497.266	-427.1151
246.982	478.9778	-546.7617	389.5497	505.2599	-393.4093	314.8322	493.4764	-485.0691	319.9023	494.4873	-481.1003	358.2092	497.2564	-438.8064
209.1445	478.8064	-582.2552	389.2203	505.2116	-393.4007	314.9556	493.4942	-485.0349	320.6976	494.6728	-480.0269	351.1472	498.1812	-449.8621
194.077	482.877	-593.468	388.3038	504.2402	-396.6319	314.9414	492.735	-485.2735	320.0762	493.3247	-479.5784	345.1107	498.2933	-456.8559
211.0069	481.9404	-581.9993	383.7138	502.3371	-403.2231	312.4654	491.7013	-488.5605	316.5344	491.0314	-483.6731	343.5746	498.0699	-458.8354
225.6413	479.1894	-568.3884	377.6554	501.1539	-414.9313	307.2696	491.108	-493.401	311.3408	490.7973	-488.4024	342.6494	497.9353	-458.1736
237.1899	483.2396	-561.0434	372.2135	500.4678	-426.0659	301.1972	490.9151	-499.9399	295.6959	487.1568	-501.5345	343.4935	495.0057	-457.9791
236.2679	483.6903	-549.9938	371.9693	500.5936	-429.7244	299.2248	490.7655	-501.9266	285.1447	487.5	-512.8209	341.4058	494.4979	-459.9988
239.5966	484.6229	-541.4675	370.8188	500.4309	-427.1356	299.2767	490.7729	-501.6225	276.2236	486.4442	-521.0548	334.7354	495.4007	-465.8603
238.4531	483.8566	-542.9195	367.6442	498.4776	-429.0604	299.9936	489.5613	-501.8825	272.7761	486.8674	-524.2926	331.6212	495.2481	-469.7323
229.5328	481.4848	-549.3593	364.2283	497.4591	-435.9535	294.4436	487.6964	-504.1058	274.456	487.7139	-521.7334	332.1113	495.3902	-468.9837
219.3513	480.1535	-561.09	358.6521	498.4779	-442.8729	286.2718	486.9032	-511.3393	274.3899	485.5842	-522.4306	328.9133	494.7909	-470.4917
215.1062	480.5151	-565.1177	356.3555	498.3085	-446.1372	279.6335	486.489	-518.1467	269.6577	485.3502	-525.8537	326.1001	492.9728	-474.3323
218.6681	480.7005	-561.5956	354.1027	499.6778	-447.4761	275.1574	486.1743	-522.1035	254.0356	484.708	-537.1775	322.0508	493.3238	-478.4187
223.0911	479.0869	-558.5068	354.9455	499.8005	-446.1485	273.407	485.9309	-523.7543	245.0179	483.9566	-545.5736	320.4007	493.4405	-480.1511
226.2247	480.2374	-555.918	353.2766	498.9445	-444.9583	269.9844	484.1469	-526.545	243.5841	483.8564	-546.2511	318.5915	493.3229	-482.0847
225.076	480.4295	-555.897	350.5954	498.9387	-446.2839	264.8635	484.1582	-530.7771	243.8809	483.8451	-545.0891	314.332	491.4148	-488.0959
224.9814	480.4159	-556.5394	342.7721	497.7296	-452.5541	257.5552	483.3849	-536.3525	245.9036	481.998	-544.9265	306.9781	490.1291	-496.4389
224.9601	480.4129	-556.2516	335.1156	495.9644	-463.3151	253.7077	483.6434	-539.1636	248.2093	481.2157	-543.8393	293.2569	489.2047	-509.2956
211.6896	476.586	-561.4335	332.1362	495.1048	-467.6618	248.1215	482.9132	-541.665	247.0789	481.8313	-544.2204	280.1867	487.6138	-522.5872
196.6074	476.1996	-572.1106	330.1826	494.8278	-470.2283	240.9552	482.3893	-546.8987	246.684	482.3259	-544.166	276.049	487.0896	-526.1755
185.5354	476.5976	-579.2856	330.4635	494.8676	-469.9405	234.5465	481.4753	-551.586	247.1936	482.2869	-543.8052	277.0621	487.4778	-524.3014
187.7224	475.9264	-579.1187	331.3214	493.5509	-470.015	229.1348	480.7035	-555.3237	247.5377	482.3347	-543.5326	274.6953	486.6553	-525.3234
191.4908	474.149	-576.8965	328.1167	492.6825	-474.169	225.3504	480.1638	-557.1996	245.7699	482.089	-544.7087	267.0831	467.8216	-539.4398
188.1809	472.7916	-579.368	323.8953	492.9496	-479.202	224.3855	480.1781	-557.5999	243.712	483.0553	-545.6704	247.6995	431.6837	-584.9595
179.2531	473.6725	-583.2748	320.6484	492.7772	-483.0184	220.0915	478.817	-559.2159	242.5538	483.1608	-546.0176	242.6446	406.1435	-645.395
169.8126	472.6006	-588.6395	319.9806	492.8232	-483.7089	216.0114	478.8318	-561.6085	243.1764	483.2508	-545.5325	245.7543	391.7804	-658.4249
168.6411	472.4314	-590.7856	320.4108	492.8138	-483.0214	206.5191	476.8201	-567.1731	242.3286	482.9652	-545.2726	283.4907	385.9971	-616.7553
168.717	472.2841	-590.0335	316.0468	490.6217	-487.8618	200.209	476.4865	-572.0426	240.1892	482.6571	-546.8648	300.7392	385.4232	-572.4136
166.4924	470.6413	-590.7813	308.2743	490.9125	-495.7385	194.4257	475.7086	-574.8776	233.7375	481.465	-549.6882	308.8755	391.5884	-529.9871
154.4431	469.5026	-595.3305	306.1904	490.8227	-498.614	189.0149	474.6334	-577.97	228.7325	480.8007	-553.566	297.8652	413.3771	-524.0788
137.1787	467.456	-603.11	308.3192	491.3901	-495.8164	184.8064	474.2105	-580.2967	227.4533	480.6175	-554.5161	263.7916	448.4447	-553.4233
132.1224	467.48	-606.8499	308.3624	491.3962	-495.4424	175.6271	472.3381	-584.4658	227.6482	480.6968	-554.1606	220.4861	473.4639	-569.7051
133.6938	467.7427	-604.8471	309.424	490.0615	-495.5107	164.6421	471.0421	-590.2683	222.0087	477.9377	-557.3544	201.8333	486.7805	-571.5419
136.8535	466.6192	-604.0137	310.0014	490.8829	-493.8655	152.4513	469.3328	-596.2865	210.802	477.993	-564.7531	203.5026	485.7773	-574.2372
136.431	466.4631	-602.8139	307.8996	491.1292	-495.8701	147.4212	468.6274	-598.9853	203.058	477.2164	-570.1805	213.0004	483.1439	-570.6495
131.4166	466.6708	-605.0495	304.1947	490.0116	-498.937	146.1484	467.9621	-599.1796	200.2703	476.816	-571.7823	215.338	481.8829	-568.5124
127.9695	466.1499	-606.4277	298.2985	488.6741	-504.9651	144.3968	467.9622	-599.828	196.2596	475.6101	-573.5129	205.6764	480.0005	-581.4924
128.5621	466.234	-606.1473	289.2712	488.4537	-512.6534	138.941	466.9717	-602.2421	195.4995	475.19	-574.2675	192.118	475.071	-596.8264
130.5977	465.9201	-605.6792	283.2754	487.7388	-519.0294	134.4501	466.6788	-603.9653	191.0653	474.1343	-576.8574	182.4662	473.7099	-604.3708
130.1977	465.8649	-605.4868	281.7098	487.643	-520.2065	132.3676	466.4205	-605.3348	188.2243	473.7439	-578.6481	182.7903	473.8823	-604.0004

ISAC

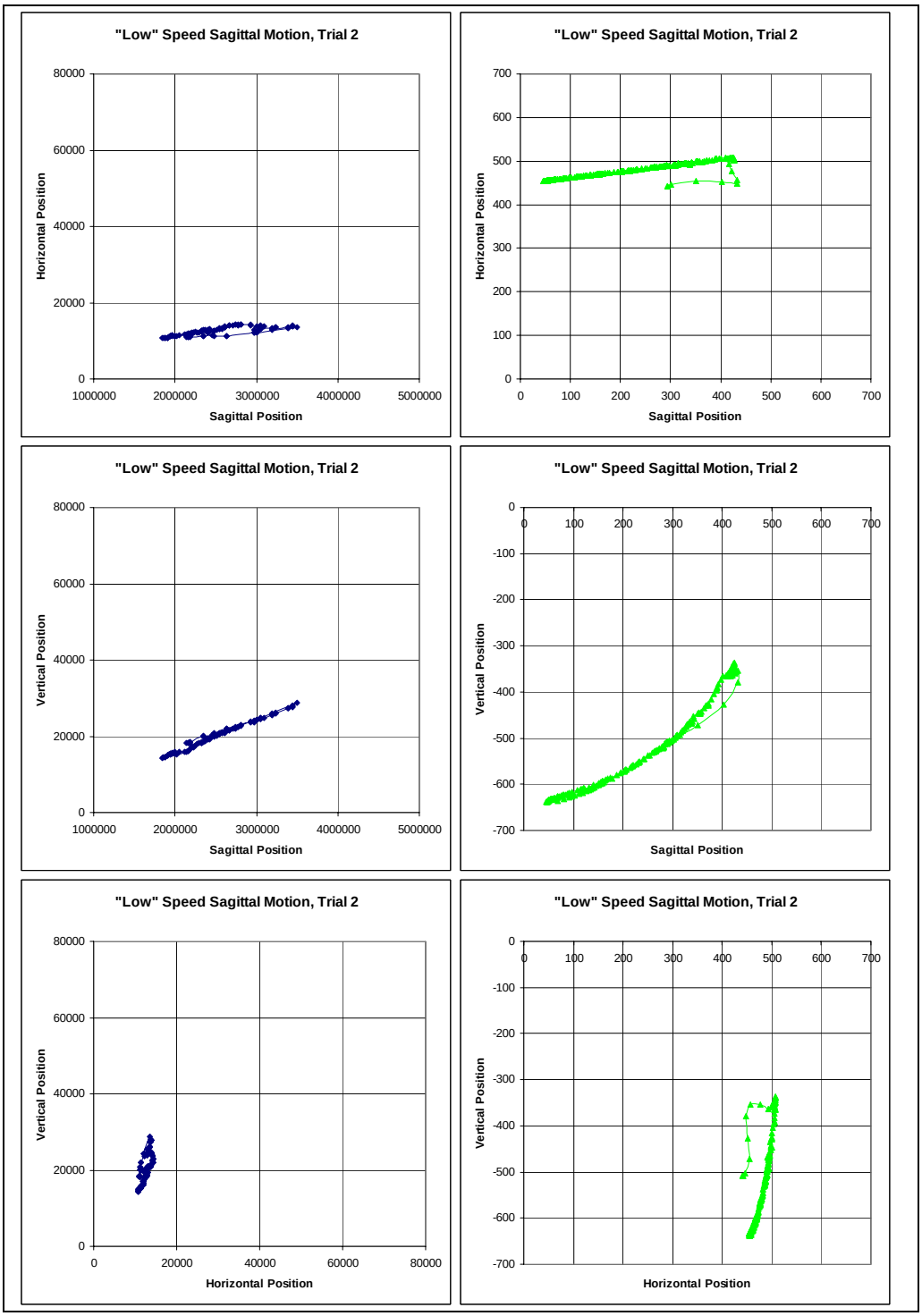
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
128.9891	466.043	-606.1162	281.4471	487.5441	-519.9667	131.9929	466.1126	-604.9224	183.2317	472.8052	-582.1383	184.3792	474.1074	-602.2383
129.45	466.0761	-605.9289	280.1764	487.1803	-521.2112	131.1645	465.9666	-605.396	179.9722	472.6929	-583.4558	187.6134	473.9182	-593.9825
125.5812	464.9525	-607.0127	279.9262	487.8239	-520.3174	126.2592	465.3213	-607.2529	179.9544	472.6905	-583.5218	190.1922	474.581	-581.7245
113.3001	463.7301	-611.9397	279.5645	487.7725	-519.4537	122.5722	465.0228	-608.9313	175.3404	471.2855	-584.9986	190.2118	474.6274	-580.4403
99.827	462.0819	-616.975	278.2514	487.156	-519.8069	121.0583	464.8713	-609.371	170.7738	471.4253	-588.7839	187.0617	473.8847	-583.9989
95.00162	461.4744	-619.6994	273.2595	486.6342	-522.3366	117.819	463.9822	-610.6752	164.4446	471.2071	-591.4748	178.268	472.7877	-588.5141
92.80464	461.2113	-619.795	268.4457	486.3696	-526.2943	113.6818	463.8116	-611.7686	163.1306	471.7895	-591.5732	177.0771	473.1144	-589.6362
89.70689	460.3469	-620.2133	266.6432	486.1713	-527.5382	108.7476	463.1227	-613.4217	158.1841	469.8389	-593.7539	176.4309	472.9817	-589.7478
85.89536	459.9777	-621.3025	265.5571	485.8392	-528.1702	105.5301	462.6735	-614.4646	145.946	468.839	-601.2377	174.7198	472.0905	-590.3661
79.69978	459.0887	-623.0827	265.3467	485.927	-527.9453	103.945	462.4522	-614.9944	140.6828	468.9039	-603.4891	174.3782	471.6799	-590.8572
75.88344	458.7422	-624.5509	262.7782	485.2102	-529.4728	103.4487	462.3829	-615.1508	141.087	469.203	-602.6795	169.0276	471.5388	-592.561
75.71454	458.7183	-624.7197	259.7343	484.7791	-532.0113	101.6273	462.1286	-615.719	143.9446	468.3121	-602.0118	166.7144	471.2548	-593.8753
76.219	458.6645	-624.6578	250.1174	482.9159	-537.2887	97.7282	461.5843	-616.9064	147.1562	468.4852	-600.4577	165.8119	471.0511	-594.2779
73.77966	458.2407	-625.1325	241.4961	482.4665	-545.4225	91.48415	460.7125	-618.7256	146.197	468.9797	-600.5591	166.0895	471.0898	-594.0326
71.0878	458.0631	-626.1504	233.8041	481.422	-551.0584	89.22434	460.4438	-620.1622	146.065	468.9958	-601.3532	166.2287	471.1092	-593.3877
68.25787	457.6044	-626.7993	232.0694	481.2264	-552.3028	85.70834	459.838	-620.9715	145.886	468.9701	-602.7493	162.2904	470.4841	-594.76
67.3878	457.4817	-628.0078	229.4829	479.8763	-553.4143	84.12626	459.5957	-621.5832	136.1029	465.2168	-607.6694	155.154	469.6018	-598.1423
65.98735	457.2843	-629.0997	221.9466	479.578	-557.9231	80.08394	459.0561	-622.8228	103.9214	461.2722	-622.7299	151.5296	469.1317	-599.1892
62.70246	456.821	-629.8536	219.6503	479.3011	-559.6237	76.42405	458.5474	-623.657	78.52799	459.4382	-637.9705	149.3929	468.9394	-600.2951
60.92154	456.6412	-631.2462	218.6206	479.1544	-560.4459	74.93223	458.4427	-624.5051	64.23126	457.3146	-651.2178	148.2821	468.7129	-600.8207
59.64277	456.4071	-631.8085	215.5969	478.4796	-561.8905	72.88485	458.156	-625.0186	67.69467	457.372	-647.7353	144.9361	468.2444	-601.1343
57.33309	456.0809	-632.3695	204.2133	477.056	-570.1844	70.24652	458.0228	-625.9247	74.08983	457.4959	-641.894	138.7803	467.5494	-603.6258
56.07092	455.9533	-632.9672	202.4997	476.8585	-571.0741	66.2049	457.5609	-627.1108	69.57495	457.1695	-642.4958	133.7402	466.8711	-605.8042
55.9972	455.8923	-633.1555	200.6511	476.3201	-571.9399	65.29522	457.4679	-627.749	59.30316	456.4818	-643.7902	132.843	466.6799	-606.2198
55.23971	455.7853	-633.2762	195.2821	475.3383	-574.1451	65.24867	457.4424	-627.8949	56.42864	456.2591	-644.7446	129.6042	466.1292	-606.7857
51.59087	455.286	-634.5	186.2538	474.1582	-579.2397	64.92257	457.3955	-628.0673	54.9865	456.0334	-645.219	123.283	465.4853	-609.1164
51.00585	455.2349	-635.1393	175.275	472.818	-585.3025	62.62735	457.0654	-628.9142	55.73358	455.6211	-645.5359	117.2928	464.5775	-611.1989
50.78736	455.1566	-635.6487	169.8907	472.1349	-588.3974	59.20442	456.573	-631.0603	53.47088	455.2917	-647.5729	112.5698	463.9941	-613.3902
50.91209	455.1742	-635.5765	166.4253	471.6427	-590.0521	56.76437	456.1538	-633.2589	48.34213	454.9185	-650.9913	107.8165	463.2918	-614.8548
50.597	455.1297	-635.9313	163.1334	471.0985	-591.573	56.39161	455.9311	-634.2356	47.83055	454.9219	-651.4002	102.8397	462.586	-616.3159
51.35869	455.2372	-635.8186	158.9444	470.3555	-593.6083	55.93676	455.8333	-634.3642	52.09489	455.5182	-650.7627	99.9917	462.2335	-618.2091
54.28342	455.6503	-635.6505	157.9755	470.48	-594.0018	54.12062	455.578	-635.2818	61.62146	456.7404	-641.6312	94.96243	461.3457	-619.7726
56.50997	455.9478	-635.2086	158.717	470.6229	-593.6316	51.67251	455.2497	-636.7192	72.50861	458.2845	-632.9451	91.64089	461.0458	-621.2092
57.82278	456.1329	-634.355	159.8512	470.3703	-593.2693	49.81336	454.9879	-638.2146	77.56267	458.98	-628.4829	88.37829	460.5818	-621.9664
59.18279	456.3421	-634.0519	158.6092	469.4513	-594.2687	48.55364	454.8105	-639.1502	76.12804	458.7143	-629.7709	88.38948	460.5834	-622.3483
60.78582	456.6219	-633.7026	155.3934	468.8674	-595.8369	48.11664	454.7489	-639.2258	74.19495	458.6468	-630.8377	88.00631	460.5289	-621.9916
63.73914	456.9856	-632.665	151.0758	468.3896	-597.6976	47.68162	454.6725	-639.3569	74.71838	458.7423	-630.7534	84.06557	459.9684	-622.8994
67.00583	457.4469	-631.4854	140.5179	466.8531	-601.9056	46.89264	454.5616	-639.934	74.60007	458.6016	-631.4938	83.22088	459.8706	-623.4436
68.84214	457.6287	-630.2106	130.5714	465.9164	-606.2497	46.41391	454.5091	-640.0079	72.16993	458.1361	-632.8716	79.88833	459.1582	-623.2443
71.61635	458.0581	-629.6365	122.0675	464.8026	-609.326	46.20128	454.4644	-640.1488	71.75981	458.2183	-633.5479	77.55744	459.0217	-624.1438
79.12204	459.1794	-627.753	121.4082	464.8008	-609.5976	45.72401	454.3973	-640.2777	86.03521	460.8931	-630.4833	73.77795	458.4645	-624.9726
84.93226	459.7076	-624.7252	120.0306	464.6389	-609.8139	45.46116	454.3603	-640.3214	97.20116	461.8366	-626.9066	69.65675	457.88	-626.2668
86.90773	459.9131	-623.7224	121.35	464.6438	-609.4318	45.22877	454.3276	-640.2126	107.1299	462.3319	-623.2098	69.31372	457.8314	-626.5992
84.56216	459.9262	-624.6133	119.9704	464.8969	-609.6966	45.60779	454.3956	-639.9463	109.1929	463.487	-621.4484	69.39865	457.8434	-626.5798
86.99523	460.2928	-624.3095	116.3001	464.4371	-610.941	45.9324	454.4266	-639.6531	111.3438	463.8759	-620.6761	69.39865	457.8434	-626.5798
95.86803	461.5724	-621.8267	115.0479	464.3748	-611.5016	49.3	454.9156	-638.4037	118.7093	464.9249	-618.1438	69.19033	457.8139	-626.9319
99.67336	461.4489	-620.0217	115.169	464.4207	-611.4447	53.19188	455.3663	-635.6175	124.9881	465.482	-615.7807	69.06858	457.7966	-627.1372
101.9705	462.2025	-618.6644	113.5237	463.8462	-612.0827	55.19258	455.6124	-633.8633	129.5915	466.0959	-613.7446	68.55827	457.7242	-627.2526
104.8206	462.7339	-618.174	108.2707	463.1106	-613.812	55.77851	455.694	-633.7764	130.7792	466.294	-612.6481	67.204	457.5322	-628.037
109.7242	463.3692	-616.7764	97.52719	461.6566	-617.2536	55.85307	455.7044	-633.6623	130.7643	466.2919	-612.5227	64.26551	457.1154	-628.8554
118.9709	464.697	-614.0274	91.70272	461.0306	-619.5282	56.11208	455.7404	-633.6126	130.6672	466.2783	-612.4288	62.60498	456.8981	-630.6363
123.9249	465.3633	-611.3669	89.48841	460.7397	-620.6816	61.75982	456.5983	-632.4042	131.1871	466.3511	-612.1771	60.08412	456.5401	-631.5599
125.3053	465.5571	-611.0418	88.05497	460.373	-621.108	67.58366	457.2423	-629.9869	139.4155	467.639	-608.9057	59.62565	456.475	-631.929
128.1668	466.0527	-610.3162	84.69626	459.9	-622.015	71.24671	457.7288	-628.5177	144.0446	468.1196	-603.3438	59.76107	456.4942	-631.9517
137.6346	467.421	-606.9599	81.04491	459.3858	-623.0198	71.87597	457.8753	-628.1166	149.161	468.836	-600.4356	62.89154	456.9023	-631.1465
148.8431	468.8622	-601.9929	79.27597	459.1367	-623.5977	74.56259	458.309	-627.4571	149.8113	468.927	-600.1222	67.66263	457.578	-629.9142
157.5698	469.7167	-598.1048	77.50343	458.8871	-624.0405	78.06667	458.7969	-626.5826	150.239	469.1296	-600.1261	71.85007	458.1911	-628.5245
162.2961	470.5609	-594.8795	75.05262	458.542	-624.8951	88.9759	460.409	-623.7228	153.075	469.5652	-599.0008	72.86214	458.0924	-628.4052
163.8705	470.7804	-594.1228	71.27292	458.0097	-625.9935	95.253	460.9929	-620.2459	164.2249	471.1376	-594.1918	73.10259	458.3079	-628.1091
167.2894	471.5697	-593.1042	69.03609	457.7142	-626.8523	103.3904	462.2961	-617.9919	173.4666	472.3601	-587.5911	77.52955	458.9331	-627.0433
173.4352	472.4767	-590.5955	68.92965	457.6992	-627.1556	107.0247	462.7744	-616.8291	180.7058	472.7524	-584.297	79.6244	459.1212	-626.2953
188.1216	474.5507	-583.3893	68.67447	457.6632	-627.2132	112.7784	463.7418	-615.3723	183.3239	473.6616	-582.6173	84.03835	459.8298	-625.3739
200.7624	476.1984	-573.8381	65.83006	457.2621	-627.8949	121.7232	465.0541	-611.9849	185.2447	473.9736	-581.8447	88.21668	460.419	-623.5407
211.2371	476.9061	-567.8897	63.58357	457.0371	-630.3792	126.8487	465.1252	-610.3564	197.55	475.7918	-576.9492	93.05952	461.0777	-622.1999
216.4278	478.1077	-563.4269	62.66833	456.9071	-630.7738	126.9824	465.4829	-609.5746	208.9094	477.4389	-567.713	98.91243	461.9019	-620.3826
216.4388	478.0604	-563.9842	61.29295	456.6759	-631.0777	126.7717	465.6083	-610.0482	219.7362	478.4204	-561.3838	100.4594	462.1197	-619.4244
222.391	479.34	-560.4801	60.65082	456.6028	-631.2014	129.6291	466.0066	-608.9663	228.3562	479.3609	-555.548	106.9776	463.0646	-617.9908
234.0313	480.9815	-552.5159	60.99411	456.6515	-631.1308	131.8282	466.3132	-608.1182	231.7394	480.3983	-552.8055	115.4807	464.2637	-614.6399
244.7043	482.5413	-543.9242	60.82248	456.6271	-631.1661	133.8605	466.5966	-607.3223	234.6785	480.7573	-552.1203	126.0817	465.7278	-611.116
258.5502	484.267	-534.231	58.76171	456.3349	-631.5847	139.4858	467.5818	-606.3918	238.3808	481.3815	-549.7791	135.8373	466.9377	-606.7685
272.6329	486.2444	-521.4979	58.07436	456.2374	-631.722	154.1698	469.9753	-600.1529	243.1571	482.0512	-546.3743	140.4116	467.5772	-604.6414
282.5378	487.5731	-512.1741	58.07436	456.2374	-631.722	172.1098	473.4162	-590.1919	246.821	482.565	-543.4721	142.51	467.8706	-604.116
286.6966	488.4083	-511.7245	57.88415	456.2104	-631.7849	185.5532	473.8033	-583.2671	248.6215	482.8174	-541.8808	148.5309	468.7124	-601.9174
289.7887	488.908	-510.4332	57.26408	456.1225	-631.9326	190.1994	474.6693	-578.9987	247.3088	482.6334	-543.0223	163.5542	470.8129	-595.4381
293.1974	489.3894	-506.8783	56.92016	456.0737	-632.0002	188.7817	475.4275	-579.4138	247.2125	482.7301	-543.0563	177.4586	472.757	-587.3608
296.2518	489.6259	-504.792	56.76659	456.0519	-632.0052	189.2985	475.8525	-580.5909	247.259	482.6815	-543.0563	183.4034	471.194	-584.0571
296.1871	489.0329	-505.1353	56.40418	456.0005	-632.1011	199.3304	477.2756	-575.148	249.0005	482.926	-542.0752	181.9707	472.2992	-584.5102
289.9916	488.1729	-511.4608	56.07858	455.9543	-632.1394	211.0059	478.983	-567.36	256.5848	484.105	-536.5784	181.1406	472.8122	-585.1033
281.6796	487.7008	-522.1271	56.09704	455.957	-632.1107	223.0077	479.3767	-560.116	268.3309	485.7591	-526.7556	185.4347	473.4031	-583.7517
283.2496	488.6713	-524.1036	55.83901	455.9204	-632.1608	226.2131	479.5228	-557.3189	275.7465	486.8033	-518.6722	197.8809	475.2963	-577.0168
285.2103	488.8897	-523.267	55.2368	455.835	-632.2769	226.5343	479.3132	-557.5561	282.3073	486.1149	-514.7632	211.3767	476.6389	-568.2388
308.5396	494.1949	-509.5913	54.7205	455.7617	-632.3758	229.4298	479.9719	-556.7282	283.1583	487.3487	-513.0692	218.0814	475.7445	-564.4002
323.621	493.0521	-483.4371	54.87408	455.7835	-632.3715	233.0508	480.6864	-553.7343	283.4905	487.7067	-514.0863	215.096	477.3879	-565.0554
336.2594	490.8953	-468.392	55.13224	455.8201	-632.3221	236.4036	481.1043	-551.354	284.199	487.8685	-513.3785	214.7018	477.673	-566.8454
335.8859	494.8349	-466.5328	54.96014	455.7957	-632.3551	239.7735	481.4696	-549.0109	284.0897	487.6657	-513.6261	223.549	480.2605	-560.1299
334.9193	495.4268	-467.728	54.96014	455.7957	-632.3551	240.2004	481.5293	-548.6243	283.7909	487.4368	-513.9904	231.7916	481.3954	-552.5239
340.0154	497.2564	-463.9906	54.94161	455.7931	-632.3837	241.1565	482.418	-548.4798	279.4687	486.8351	-518.6744	239.1142	480.5769	-548.614
346.3196	497.7171	-454.1685	54.56527	455.7397	-632.731	257.8049	487.3843	-539.3368	274.2083	486.4051	-525.2742	239.893	481.8082	-547.2218
355.6732	498.366	-443.0132	54.08738	455.6061	-633.2531	271.0246	488.9659	-524.993	274.8582	487.0424	-526.7897	239.7732	482.4895	-547.4416

ISAC

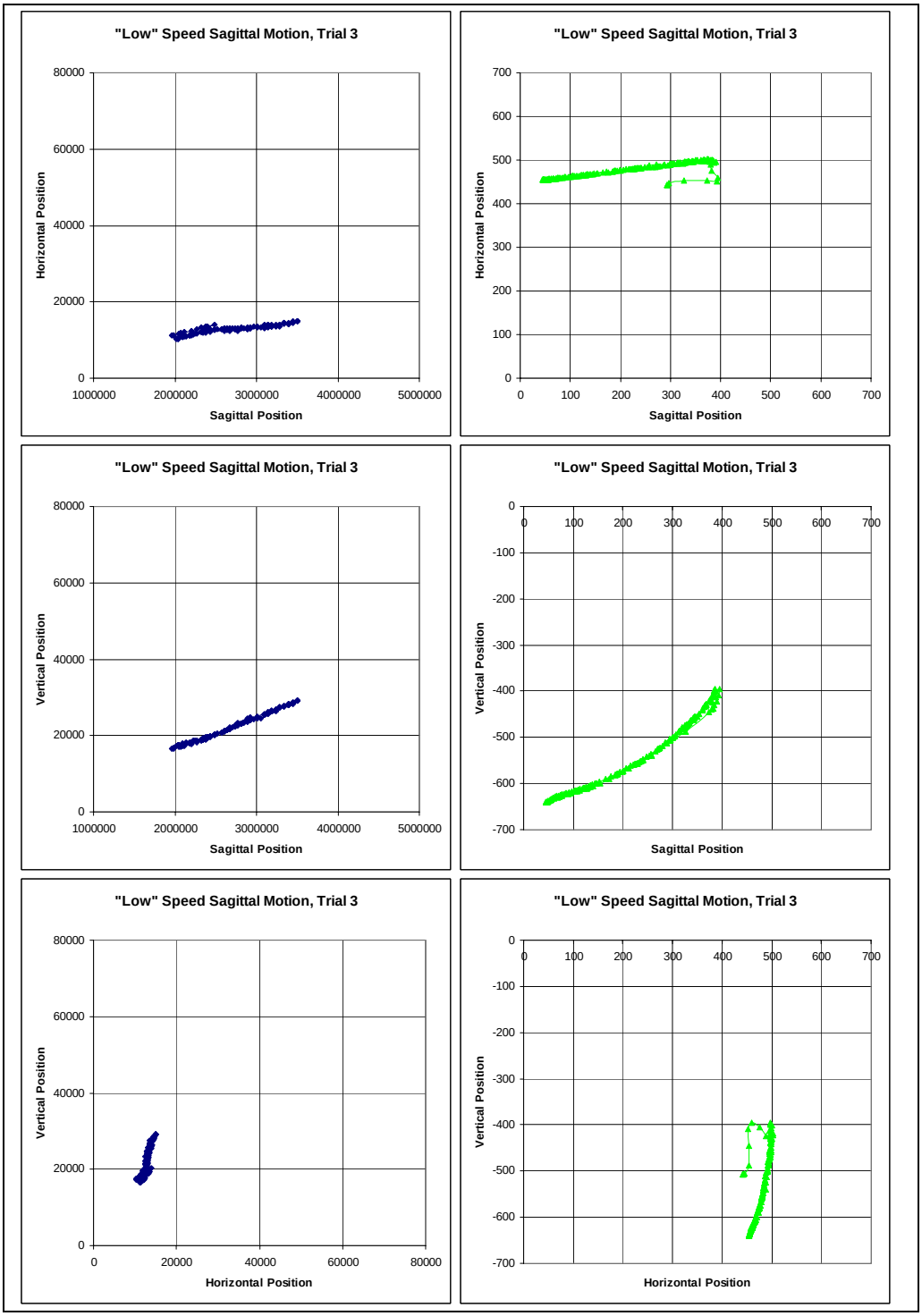
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
			54.40094	455.3875	-633.318	288.177	489.2511	-512.5178	278.4811	487.6184	-524.7079	241.3001	482.7089	-546.267
			53.06908	454.9955	-633.9639	299.0705	490.2843	-502.3096	291.1343	489.4179	-515.0742	244.0286	482.9915	-544.7464
			50.81412	454.6906	-634.7198	299.5345	490.6126	-501.2875	296.7431	489.3699	-507.5722	247.7537	483.525	-542.3307
			50.0722	454.6673	-635.0639	300.5076	491.8727	-502.0919	299.3069	487.3123	-506.7003	250.2649	483.4938	-539.8761
			50.13978	454.6765	-635.0769	304.9985	492.2589	-497.8628	299.8185	490.0618	-504.4858	253.3113	483.7004	-538.9999
			49.90923	454.6451	-635.1921	315.4313	493.7699	-488.1009	302.262	491.7298	-500.8264	253.4953	483.8391	-537.9125
			49.55494	454.5968	-635.6024	320.1082	494.0276	-482.3641	304.8315	492.0341	-497.9207	253.0616	483.7778	-538.5881
			49.35326	454.5694	-635.7868	324.5389	491.2096	-480.0362	310.0914	492.6569	-492.1522	252.6545	483.7202	-538.9225
			48.93087	454.5118	-636.4063	326.7241	493.5557	-476.1734	317.9442	492.2607	-484.2098	252.7903	483.7394	-538.8111
			48.57425	454.4632	-636.815	327.9986	494.5181	-473.6775	323.9569	492.747	-477.1586	252.9938	483.7682	-538.6439
			48.25835	454.4654	-637.0959	327.1215	494.3937	-474.8747	326.1497	494.1139	-474.3705	254.2779	483.9498	-537.6161
			47.87031	454.5478	-637.1239	329.8437	494.7797	-472.6695	327.7239	494.5504	-472.2884	263.7124	484.6416	-529.6384
			47.0857	454.5737	-637.3931	333.7096	494.9655	-468.1273	332.5149	495.3031	-467.4284	264.9708	484.8754	-528.7873
			46.42615	454.5556	-637.4952	337.584	495.5111	-463.0351	336.7009	495.8253	-461.6098	264.2904	484.7805	-529.7768
			45.70941	454.4542	-637.8634	340.1685	495.7276	-459.649	343.892	496.8466	-453.9499	266.2452	485.1119	-528.8972
			46.03849	454.5008	-637.8328	343.5553	495.98	-457.5688	347.2892	497.3291	-448.5699	273.9362	486.1857	-522.3707
			48.99134	454.9184	-637.8932	345.013	496.1838	-455.5362	351.7997	497.9697	-443.3328	280.6003	487.1161	-515.9082
			68.17791	458.4492	-636.6341	346.8494	496.4406	-454.311	353.3624	498.1916	-440.8815	285.0475	487.1112	-512.2015
			81.44812	459.2021	-631.6733	353.2837	497.4165	-449.0195	358.7826	498.9614	-438.5551	286.8572	487.6119	-510.1104
			92.25701	460.6764	-628.219	360.5453	498.4333	-438.4903	362.7436	499.5239	-434.8179	286.9272	487.6216	-510.6974
			97.44074	461.8203	-625.9484	364.5638	498.9173	-432.097	367.4687	500.195	-431.771	289.8576	488.2814	-508.3504
			101.9173	462.6117	-624.6829	367.1613	497.9375	-429.5893	371.0262	500.6202	-426.2357	292.6403	488.7329	-505.2015
			111.4429	464.1424	-620.9782	366.5829	498.8049	-430.524	372.6692	500.7729	-423.7038	298.3274	489.5258	-500.0601
			119.6103	464.9644	-618.1315	366.5535	499.1165	-431.2914	373.9391	500.9528	-422.5569	300.5238	489.7663	-497.4172
			126.1806	465.8344	-614.0732	367.4155	499.316	-429.9038	374.8891	501.0873	-421.1589	299.4177	489.4159	-498.9863
			131.6127	466.6026	-612.0182	367.7702	499.4449	-429.1313	375.5659	501.1831	-419.7125	298.6326	489.503	-500.177
			134.4471	467.0034	-610.4923	368.3571	498.9719	-429.0506	375.2284	501.1353	-420.1733	298.885	489.6035	-499.8191
			138.2569	467.8759	-608.794	368.3941	499.0563	-428.5817	375.1021	501.1175	-420.4466	297.8218	489.1947	-501.5524
			141.3795	468.2897	-606.8662	367.3664	499.23	-429.7488	375.9884	501.243	-419.3368	297.7324	489.2474	-502.1293
			146.462	469.0177	-604.2091	365.8584	499.0195	-432.2184	377.0464	501.3928	-417.3768	298.8686	489.6012	-500.9426
			149.0182	468.816	-602.3044	365.4376	498.9607	-432.6423	377.9646	501.5228	-417.4431	305.1123	490.5384	-495.3023
			150.9032	469.1873	-601.2013	364.4181	498.8184	-434.7678	381.9646	502.0893	-413.4125	313.352	491.6888	-485.7756
			151.942	469.5497	-600.5675	364.8623	498.8804	-434.4461	385.3924	502.5748	-405.7084	317.8308	490.7254	-481.7302
			154.0437	469.9573	-599.5609				388.1312	502.9626	-400.7357	316.7514	491.1986	-483.6386
			155.4735	470.1978	-598.2101				389.4019	502.3901	-399.0823	315.4528	491.9133	-485.3756
			156.6858	470.3707	-597.5967				389.3367	502.381	-400.4156	315.7467	492.0919	-485.512
			157.967	470.3296	-596.9611				391.5291	503.1076	-399.6711	316.2177	492.2267	-485.6818
			157.8106	470.382	-596.9656				393.4022	503.4558	-396.7073	328.4795	494.0149	-480.025
			161.5647	471.1046	-595.2261				395.2746	503.7198	-394.0575	337.0924	495.2225	-468.181
			178.1126	473.5922	-587.3164				395.783	503.5368	-393.731	343.7588	494.3001	-460.7078
			193.901	475.8566	-576.176				397.6365	503.9676	-393.3252	343.8777	495.1327	-459.9945
			202.7879	474.4598	-571.5509				399.7773	504.2691	-388.6369	343.6697	495.996	-459.9921
			205.2906	475.9029	-569.2222				401.5285	504.6017	-384.8554	345.5397	497.2304	-456.4774
			204.8263	476.3507	-569.3771				403.4726	504.8759	-380.5969	347.3234	498.5396	-452.9389
			207.0919	476.6652	-568.3642				406.209	505.2618	-373.9733	349.3692	497.7759	-450.7845
			213.7914	477.5952	-564.5788				406.5612	505.3114	-372.3478	349.5653	497.6523	-450.9789
			219.0474	477.6837	-561.3848				406.6746	505.3274	-372.6699	352.9041	498.6615	-445.5813
			226.5164	479.0565	-556.2186				407.2122	505.4032	-372.1394	356.9935	499.248	-439.0217
			232.9141	480.0411	-551.2409				407.6882	505.4704	-371.9959	361.3614	499.8744	-432.121
			242.7092	481.4471	-545.3978							365.2027	498.2988	-427.83
			253.143	482.8892	-537.3825							367.1274	498.2486	-425.0425
			261.835	484.0905	-530.1387							367.1952	498.5738	-424.5399
			271.49	485.425	-524.4534							366.9748	498.5433	-424.9643
			278.1157	486.2796	-518.4082							369.8953	499.1851	-421.9249
			284.2626	487.1279	-512.9846							373.8371	500.3748	-415.3488
			285.2042	487.2579	-512.0561							379.0218	501.4279	-406.5459
			285.0339	487.2969	-512.6245							381.1179	501.9694	-401.7162
			285.1687	488.1301	-511.8458							374.6465	500.0853	-414.0343
			286.9489	489.5182	-510.8323							357.0241	496.0937	-445.8747
			288.1259	489.9424	-509.3829							348.2819	499.6632	-471.4277
			289.4299	490.068	-509.1133							348.4867	502.3555	-477.1864
			290.3024	489.9386	-508.3494									
			289.8264	489.6784	-508.9818									
			290.4431	490.0229	-508.3303									
			292.6805	490.4104	-506.1152									
			293.4411	489.1663	-506.4045									
			293.7163	489.3338	-506.0901									
			293.3268	489.4721	-506.3453									
			293.3291	489.4724	-506.4478									
			293.4522	489.4898	-506.3203									
			292.9036	489.4122	-507.1673									
			293.339	489.4738	-506.8921									
			293.8949	489.5524	-506.386									
			293.9089	489.2964	-506.9194									
			297.6272	489.8193	-507.0611									
			315.1768	494.4234	-492.569									
			327.6411	495.1809	-476.6076									
			337.6996	490.7904	-468.7231									
			339.1754	491.1247	-467.6248									
			337.8603	493.6485	-467.3856									
			339.9091	494.7344	-463.9695									
			343.4705	495.2252	-459.2012									



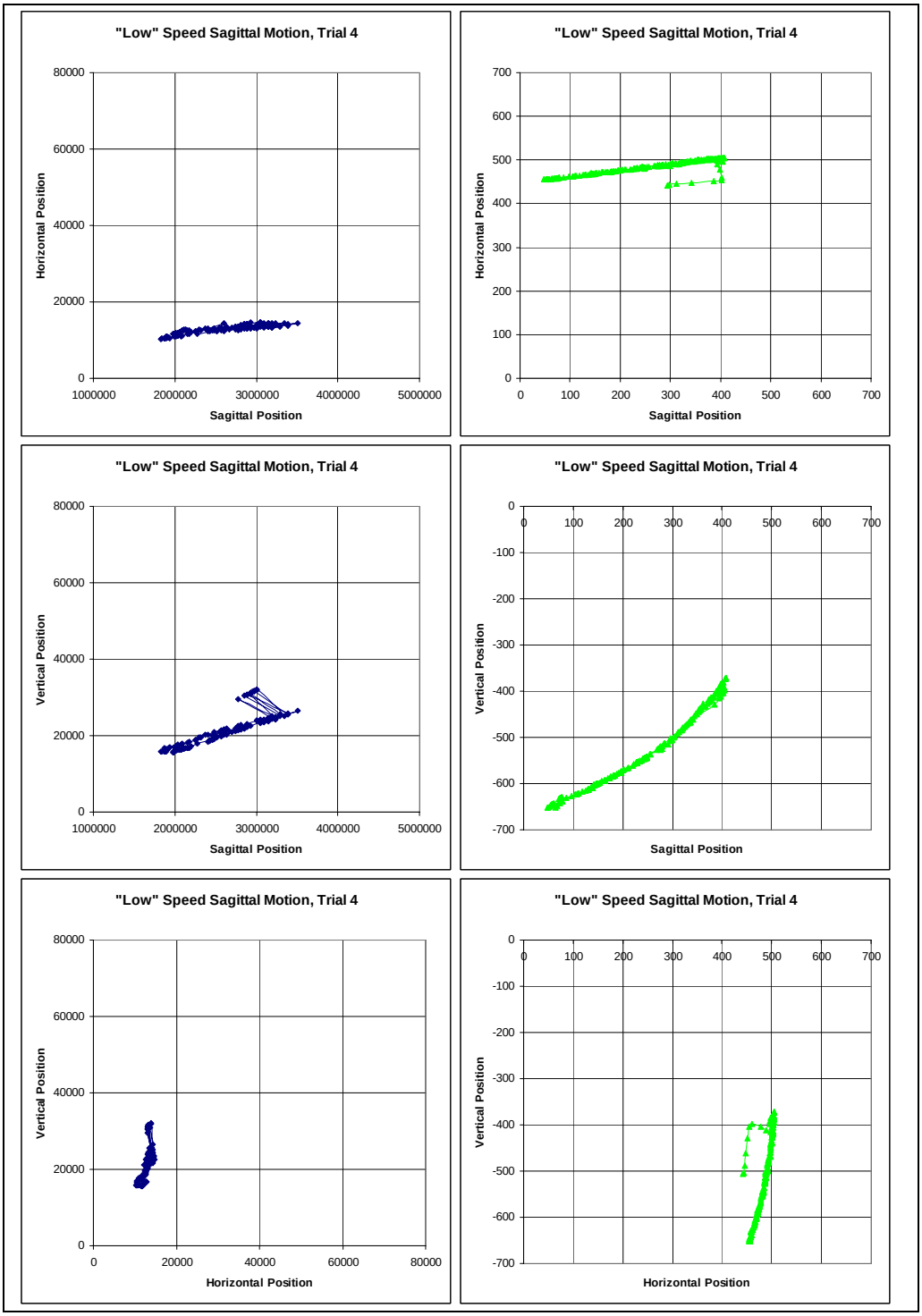
"Low" speed sagittal motion, trial 1; human motion is shown on the left, that of ISAC on the right



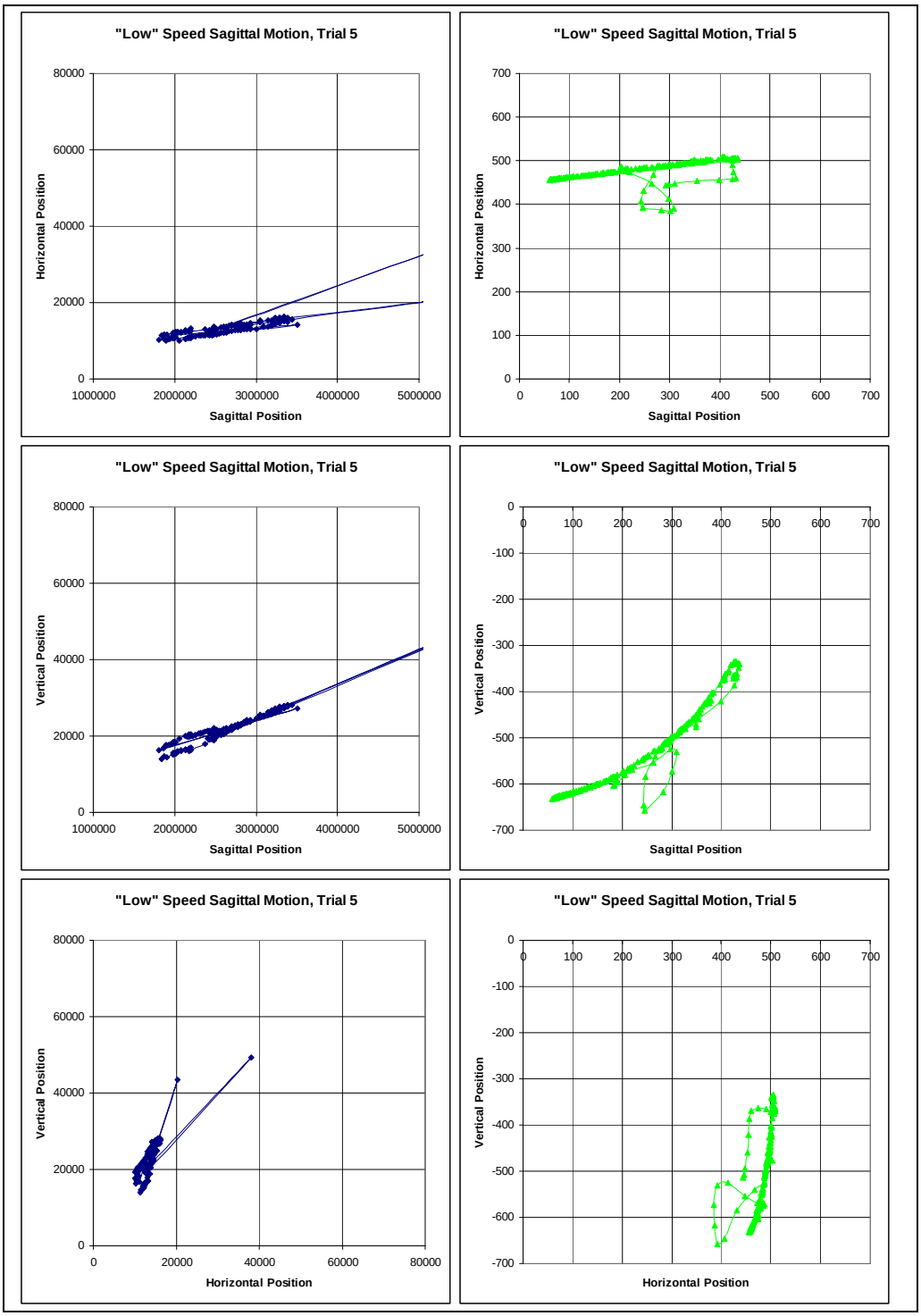
"Low" speed sagittal motion, trial 2; human motion is shown on the left, that of ISAC on the right



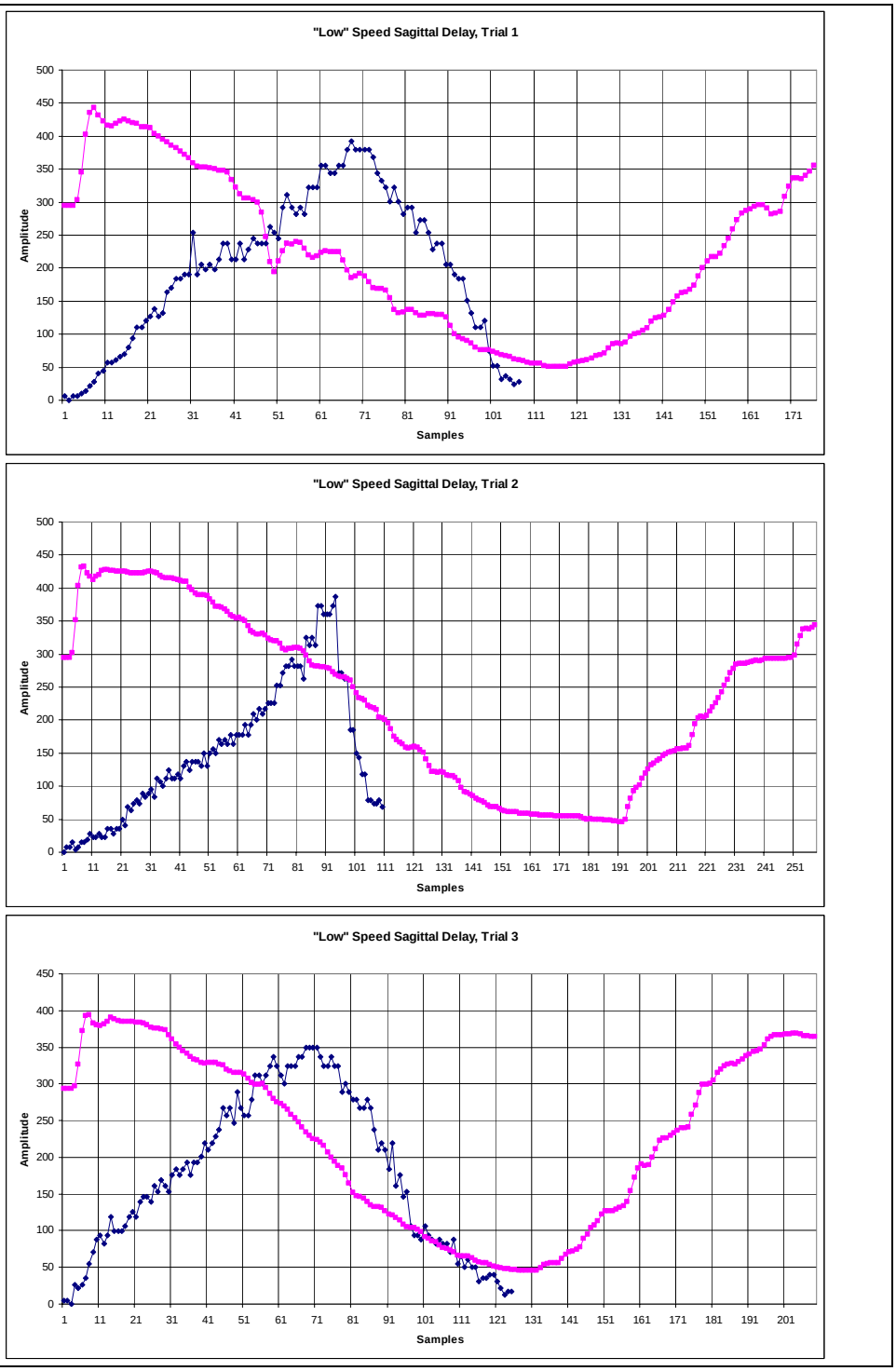
"Low" speed sagittal motion, trial 3; human motion is shown on the left, that of ISAC on the right



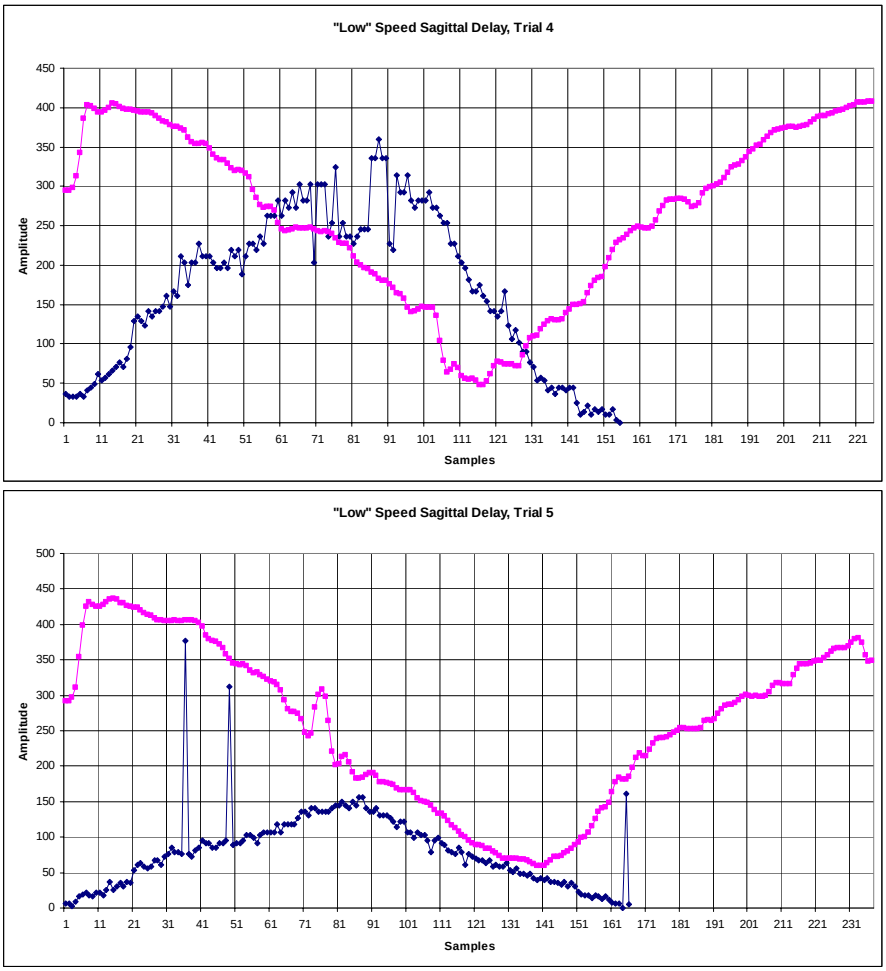
"Low" speed sagittal motion, trial 4; human motion is shown on the left, that of ISAC on the right



"Low" speed sagittal motion, trial 5; human motion is shown on the left, that of ISAC on the right



"Low" speed sagittal delay, trials 1-3; human motion is shown on the on bottom initially, that of ISAC on the on top initially



"Low" speed sagittal delay, trials 4-5; human motion is shown on the on bottom initially, that of ISAC on the on top initially

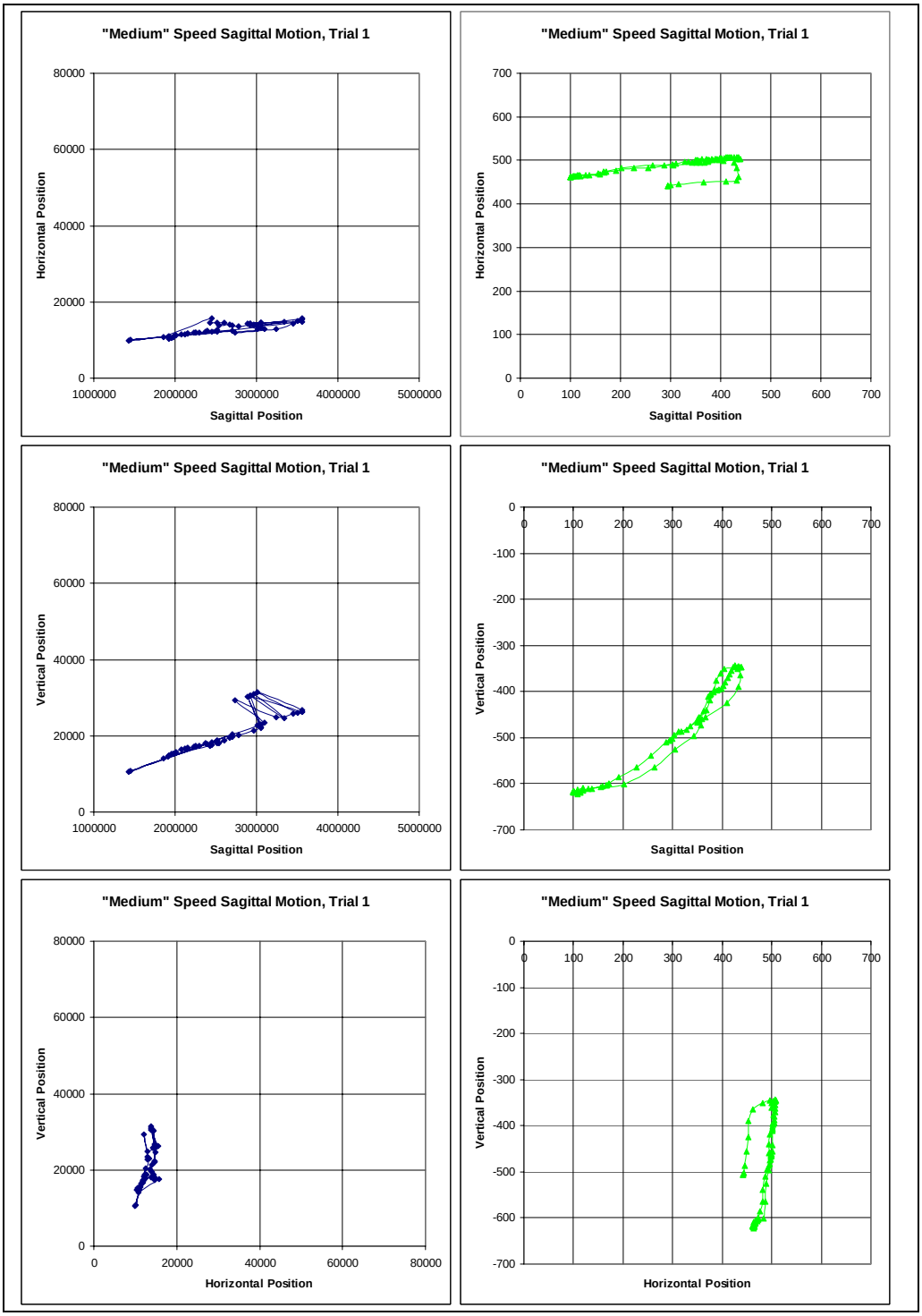
Sagittal motion at "Medium" speed

Human

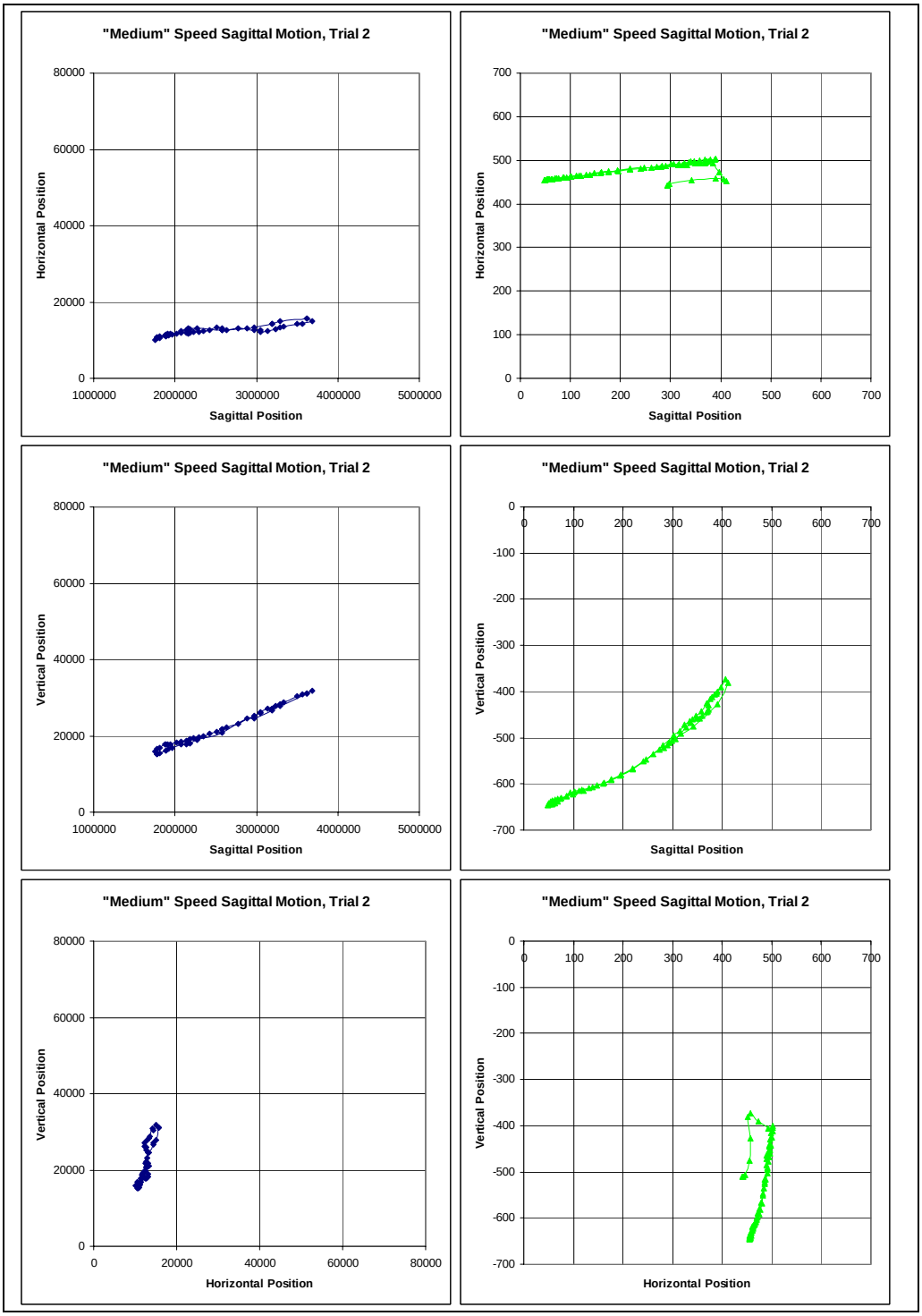
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
1858044	10773.91	14000	1810805	10915.25	15364.41	1682480	10968.5	12952.76	1780625	10266.67	14058.33	1890929	10964.6	14681.42
2456035	15689.66	17701.15	1780625	10733.33	15225	1780625	11083.33	13650	1780625	10266.67	14058.33	1890929	10964.6	14743.36
2428125	14636.36	17340.91	1780625	10616.67	15166.67	1780625	11083.33	13883.33	1795588	10411.76	14294.12	1907813	11000	14812.5
2513824	14576.47	18035.29	1890929	11088.5	16106.19	1780625	10850	13825	1795588	10411.76	14411.76	1925000	11162.16	14945.95
2605793	14512.2	18695.12	1925000	11288.29	16648.65	1842026	10741.38	14422.41	1826282	10470.09	14658.12	1978472	11277.78	15231.48
2543750	13833.33	18166.67	1960321	11495.41	16761.47	1907813	11000	14750	1858044	10530.43	15156.52	2035000	11533.33	15666.67
2670938	14000	19425	2074515	12436.89	17737.86	1960321	10853.21	15091.74	1907813	10750	15750	2115594	11574.26	16079.21
2704747	13734.18	19759.49	2158333	13080.81	18242.42	2225781	11375	22895.83	1942500	10818.18	15972.73	2249211	12452.63	16947.37
2775000	13545.45	20090.91	2136750	12740	17850	2456035	11827.59	19149.43	2054567	11711.54	16961.54	2297581	12720.43	17010.75
2967708	14000	21291.67	2180357	12857.14	18071.43	2513824	11776.47	19435.29	2115594	11990.1	17257.43	2348077	13000	17230.77
3052500	14600	22200	2273138	13106.38	18840.43	2574398	12060.24	19734.94	2180357	12285.71	17714.29	2456035	13114.94	18586.21
3052500	14600	22100	2574398	12734.94	20915.66	2605793	12121.95	19719.51	2249211	12747.37	17757.89	2605793	13487.8	19378.05
2887500	14378.38	30175.68	2887500	13054.05	24500	2637963	12012.35	20049.38	2322554	13086.96	17956.52	2637963	13222.22	19530.86
3052500	14600	22300	2967708	13416.67	24597.22	2849000	12226.67	21560	2202835	11762.89	17319.59	2670938	13300	19687.5
2927055	14287.67	30397.26	3189179	14313.43	26746.27	2927055	11986.3	22726.03	2456035	12310.34	19229.88	2775000	13363.64	20545.46
2927055	14287.67	30493.15	3189179	14313.43	27059.7	3096739	12275.36	24144.93	2543750	12500	19833.33	2967708	14000	22069.45
3338672	14875	24609.38	3287308	14969.23	27892.31	3142280	12558.82	24397.06	2670938	12950	20475	3052500	14600	22700
2967708	13805.56	30916.67	3621610	15779.66	31084.75	3189179	12432.84	24656.72	2887500	13243.24	22040.54	3009507	14098.59	22577.46
2927055	13904.11	30589.04	3621610	15779.66	31203.39	3237500	12515.15	24712.12	3009507	13507.04	23169.01	3096739	14304.35	23130.44
3561250	15633.33	26366.67	3684052	14965.52	31741.38	3287308	13030.77	24876.92	3096739	13695.65	23942.03	3237500	15060.61	24075.76
3561250	15400	26250	3502869	14344.26	30295.08	2811513	11973.68	29750	3237500	14212.12	25030.3	3338672	15312.5	24718.75
3502869	15032.79	25934.43	3561250	14233.33	30916.67	3338672	13125	25046.88	3237500	14000	25242.42	3338672	15093.75	24828.13
3009507	13704.23	31549.29	3338672	13562.5	28765.63	3446371	13774.19	25967.74	3391667	14111.11	26222.22	3502869	15721.31	26049.18
3561250	14700	26833.33	3287308	13246.15	28323.08	3561250	14233.33	26833.33	3446371	14451.61	26758.06	3446371	15129.03	25854.84
3446371	14225.81	25854.84	3237500	12939.39	27893.94	2887500	13054.05	30270.27	3684052	14724.14	28844.83	3684052	16172.41	27758.62
3237500	12939.39	24924.24	3142280	12352.94	27176.47	3684052	14965.52	27637.93	3561250	14466.67	27650	2115594	12267.33	20861.39
2739423	12025.64	29256.41	3052500	12200	26100	3748684	15105.26	28245.61	3502869	13655.74	27311.47	2849000	13533.33	30333.33
3096739	12884.06	23536.23	3052500	12600	25900	3748684	15105.26	28368.42	3338672	12906.25	25593.75	3621610	14593.22	27406.78
3009507	12915.49	22676.06	2967708	12638.89	25180.55	2775000	12090.91	29272.73	3237500	12303.03	24924.24	3621610	14355.93	27288.13
3052500	13400	23100	2775000	13000	23272.73	3391667	12777.78	25666.67	3142280	12352.94	23779.41	3502869	13885.25	26278.69
1453571	10142.86	10761.9	2637963	12703.7	22209.88	2704747	10898.73	28708.86	3052500	12400	22600	3446371	13548.39	25854.84
2704747	12493.67	20379.75	2574398	12566.26	21843.37	3189179	11597.01	24029.85	3009507	12521.13	22281.69	3391667	13222.22	25555.55
1434060	9912.752	10664.43	2574398	13072.29	21674.7	2637963	10456.79	27913.58	2887500	12297.3	21189.19	3391667	13444.44	25444.45
2513824	12270.59	18776.47	2513824	13258.82	21164.71	3009507	10943.66	22183.1	2849000	12600	21093.33	2456035	11183.91	26310.35
2456035	12149.42	18183.91	2428125	12727.27	20522.73	3009507	11140.84	21985.92	2704747	12139.24	19936.71	2967708	12055.56	22166.67
2513824	12764.71	18776.47	2348077	12384.62	19923.08	2967708	11083.33	21777.78	2637963	12358.02	19444.45	2704747	11430.38	20379.75
2400843	12505.62	17853.93	2297581	12268.82	19720.43	2887500	10972.97	21094.59	2574398	12228.92	19060.24	1352373	9392.405	9924.051
2374167	12288.89	17966.67	2225781	12104.17	19322.92	2775000	11000	20000	2484593	12046.51	18476.74	2637963	12012.35	19444.45
2297581	11967.74	17462.37	2158333	11666.67	18808.08	2670938	10850	19250	2400843	11719.1	18011.24	1369712	9602.564	10006.41
2249211	11863.16	17242.11	2180357	11857.14	19142.86	2574398	11216.87	18385.54	2348077	11615.38	17769.23	2513824	12105.88	18447.06
2249211	11863.16	17315.79	2136750	11900	18760	1387500	9363.637	10136.36	2297581	11516.13	17537.63	1378548	9709.678	10116.13
2225781	11958.33	17135.42	2074515	11893.2	18485.44	2513824	11776.47	17870.59	2249211	11568.42	17536.84	1387500	9818.182	10181.82
2158333	11808.08	16757.58	2015802	11754.72	18226.42	2456035	11666.67	17379.31	2202835	11329.9	17103.09	2348077	11615.38	17153.85
2158333	11808.08	16898.99	1942500	11709.09	17881.82	2428125	12090.91	17261.36	2136750	11200	16730	2348077	11615.38	17230.77
2115594	11574.26	16564.36	1907813	11625	17687.5	2374167	11977.78	16800	2115594	11435.64	16564.36	2273138	11468.08	16680.85
2074515	11485.44	16310.68	1890929	11460.18	17778.76	2513824	13588.24	17541.18	2015802	10962.26	15849.06	2273138	11765.96	16755.32
2015802	11226.42	15716.98	1874342	11298.25	17684.21	2322554	12478.26	16510.87	1960321	10853.21	15348.62	2225781	11666.67	16625
1996963	11186.92	15570.09	1780625	10850	16566.67	2297581	12870.97	16182.8	3502869	13885.25	25360.66	2158333	11383.84	16191.92
1978472	10888.89	15361.11	1810805	10559.32	16906.78	2202835	12773.2	15371.13	1907813	10750	14937.5	2115594	11297.03	15801.98
1978472	10759.26	15296.3	1751434	10098.36	16008.2	2158333	12515.15	15272.73	1890929	10840.71	14929.2	2136750	11480	15890
1960321	10596.33	15220.18				2094853	12352.94	14892.16	1858044	10530.43	14730.43	1996963	11186.92	14850.47
1925000	10279.28	14882.88				2035000	11800	14733.33	1826282	10230.77	14598.29	1996963	11448.6	14523.36
1925000	10531.53	14756.76				2015802	11622.64	14462.26	1810805	10322.03	14355.93	1960321	11238.53	14192.66
1907813	10750	14562.5				1996963	11448.6	14261.68	1795588	10294.12	14117.65	1978472	11407.41	14388.89
1925000	11036.04	14882.88				1978472	11277.78	14129.63	1780625	10266.67	14058.33	1907813	11000	14000
						1925000	10783.78	13621.62	1795588	10294.12	14176.47	1942500	11200	14190.91
						1890929	10592.92	13318.58				1942500	11072.73	14190.91
						1907813	10875	13250						
						1874342	10561.4	13324.56						
						1826282	10111.11	13042.74						

ISAC

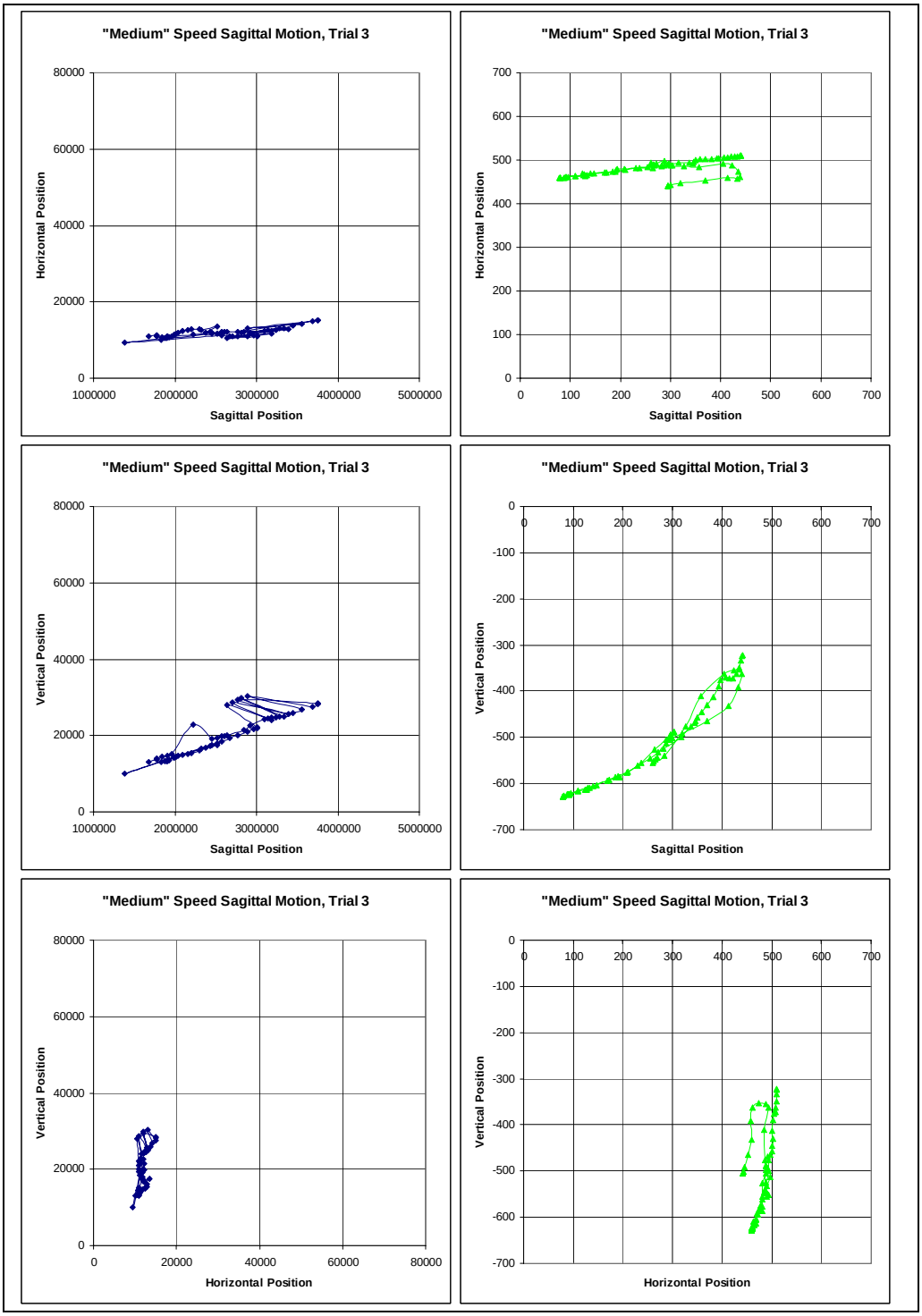
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
294.2803	441.4558	-506.3158	293.7455	441.6355	-510.259	294.7434	440.8758	-504.8347	293.8684	442.7212	-508.2959	291.1896	444.9714	-511.0572
294.3425	441.4548	-506.2586	293.7833	441.5778	-510.2248	294.8423	440.8168	-504.7433	294.1001	442.3746	-508.2959	291.3157	444.9709	-510.943
294.3425	441.4548	-506.2586	293.8001	442.2066	-509.8245	294.8806	440.7589	-504.7433	297.5549	443.5512	-505.5666	291.5862	445.313	-510.5542
299.5696	444.0032	-502.8019	297.8473	446.4099	-506.7599	294.9436	440.7578	-504.7203	317.3656	447.1862	-485.7392	309.7935	453.1759	-499.9583
316.6484	445.1918	-486.5422	342.4412	455.2927	-475.5392	299.8399	443.2395	-501.8165	357.8935	449.738	-447.366	354.4014	454.9784	-456.9596
365.8574	449.8932	-455.3948	390.4013	457.8303	-426.7298	320.1497	445.9371	-490.9768	410.1029	451.9844	-404.4086	404.2428	452.2994	-398.7064
409.8827	452.2247	-424.7339	411.6163	452.3324	-381.7748	369.4524	452.9319	-464.0258	432.9591	452.4025	-378.5027	420.6705	451.3892	-366.1942
432.8227	453.343	-389.4696	406.7874	456.7012	-372.988	413.123	459.0908	-432.0356	431.7886	458.3004	-370.4102	417.2468	461.3893	-362.8642
435.8633	461.2856	-365.4703	397.6975	472.4063	-391.4221	433.3619	458.0859	-391.9288	421.1826	478.4096	-375.1137	410.562	480.3035	-382.2146
431.2881	481.3819	-351.8006	384.5216	492.291	-406.9766	439.7174	461.1561	-363.1238	414.2381	491.8544	-381.1409	404.6955	497.2103	-404.8356
426.3775	495.5719	-344.2484	375.2002	497.9886	-415.9589	434.7073	473.8528	-352.351	407.3055	504.5448	-383.4294	398.602	499.846	-421.3783
426.6549	506.144	-342.3877	371.6495	497.5135	-429.8575	422.8828	487.7712	-354.9016	406.8061	505.1718	-383.7533	398.6993	498.6718	-420.9139
432.2033	507.9139	-344.2715	371.9129	497.4691	-439.7036	404.7323	492.6953	-363.0879	405.0325	502.7578	-386.6811	398.9193	498.3611	-417.819
434.9795	507.5587	-346.3484	371.3463	495.4872	-441.9123	356.7384	483.7746	-409.9318	401.9104	496.7832	-400.2184	397.178	494.1886	-414.0103
438.1198	503.8108	-347.2129	368.7259	494.0483	-444.6928	327.2592	485.5302	-477.1368	395.9494	498.8249	-406.8086	388.0834	493.2809	-399.1133
438.7251	503.2398	-346.3223	359.7414	493.9924	-451.8179	283.1645	485.921	-539.2903	390.4546	499.6963	-405.6916	368.5557	492.6147	-397.0973
431.3904	503.1305	-346.9824	354.0637	493.4127	-457.4873	261.1473	490.6542	-554.8851	387.8428	502.5052	-400.8881	362.606	494.1288	-408.9852
404.7243	499.2714	-351.435	349.1888	493.3821	-457.5923	261.547	493.0934	-551.9524	383.1165	500.2782	-396.8141	349.0158	492.7604	-438.0709
396.6137	500.2619	-359.751	333.8262	489.7219	-463.652	265.9769	489.6963	-548.5567	375.8694	493.5101	-406.2637	335.1435	490.1758	-466.1935
388.4635	503.7615	-377.1125	324.6443	489.7512	-472.428	269.0428	487.9486	-544.7052	362.7775	492.6798	-429.4509	318.5258	485.8747	-491.9472
384.8127	501.6658	-394.8197	314.5184	489.5278	-484.7509	271.1426	490.863	-532.2517	347.0592	494.4472	-454.1224	298.9351	488.3698	-511.3374
374.9134	495.7868	-418.476	301.6928	491.2503	-495.9168	287.0621	497.6695	-512.6751	332.6569	494.8173	-472.4629	282.5381	488.0086	-526.0072
367.5881	494.4537	-439.698	281.4804	485.2631	-515.6987	296.2601	491.8417	-493.8246	322.4706	493.1723	-486.4287	261.6526	485.1669	-543.482
351.2421	495.6146	-457.765	247.2681	482.4073	-547.0403	303.2854	487.4432	-487.0211	317.6818	492.9163	-491.4724	243.9094	482.647	-557.555
336.3147	496.7944	-475.7565	219.4815	480.0729	-567.7233	301.0356	487.8681	-489.9584	304.7179	487.7631	-503.8004	221.3062	479.4369	-566.846
329.065	496.1752	-483.5938	193.5448	477.6839	-582.6141	294.4932	488.9913	-497.8093	286.9313	488	-518.8284	207.8448	477.7618	-574.4112
311.544	493.48	-487.6453	175.5065	475.0273	-592.1678	288.0444	488.1554	-505.0965	257.1987	484.42	-538.491	190.1005	475.2239	-583.0041
303.7979	491.3521	-494.8721	161.6591	471.9582	-598.0136	264.5201	481.7844	-526.5503	226.4539	480.066	-563.6485	176.5047	473.2383	-590.1514
286.9888	487.693	-509.3764	148.2686	469.9867	-602.4242	237.4071	480.7142	-555.3735	207.7555	478.9832	-574.7003	163.8106	471.4253	-594.0349
256.2044	481.5572	-538.4996	131.2353	467.1573	-608.4954	208.0562	480.2228	-576.8852	197.2165	477.7852	-579.5873	145.6265	468.7237	-601.0627
227.1906	481.4525	-564.9042	117.7207	465.43	-613.8408	193.4437	480.3373	-585.6141	187.8803	474.3869	-583.825	137.1777	467.9871	-607.3941
190.7646	476.7298	-585.8764	111.5808	464.6414	-615.0441	192.4536	478.3249	-586.389	171.7128	471.5547	-590.0581	138.8582	469.4813	-603.9817
170.5554	473.4687	-599.9488	102.2795	463.1627	-617.3012	189.7759	474.0005	-584.6304	152.9136	470.2328	-597.4452	168.2999	475.6547	-594.936
166.6437	473.2039	-603.7081	100.7861	462.8913	-618.3636	171.8944	470.8634	-591.8621	135.5848	467.888	-604.7976	205.813	480.069	-569.9583
168.2507	472.5726	-602.6819	92.74397	461.0574	-619.1618	139.9007	468.7546	-606.1593	122.2426	465.8816	-610.6927	234.5786	478.8587	-548.8311
158.861	467.4179	-604.2145	85.49865	460.3777	-626.2955	124.6946	468.6695	-613.8693	116.8731	465.1007	-612.543	234.3866	480.9265	-546.7053
136.6717	466.8569	-611.5366	73.1873	458.6862	-631.8899	127.6469	467.4552	-612.8678	108.7185	463.2007	-615.7412	226.3874	482.4651	-556.8987
114.2338	464.9756	-619.2306	68.63645	458.0271	-637.555	133.4704	465.6744	-609.9446	103.1015	462.6494	-618.142	223.0055	482.7688	-559.5501
107.7747	464.3274	-622.0216	62.93767	457.1283	-641.9278	128.9471	463.9238	-609.9465	93.64148	461.3792	-620.7671	196.9841	470.3339	-573.5143
113.0896	464.3515	-618.8193	59.57712	456.4857	-642.7311	109.7429	462.8225	-615.4365	88.85007	460.9075	-622.0928	154.804	467.5935	-603.8388
119.4761	463.5074	-613.608	57.57488	456.3559	-643.4395	95.61906	461.1664	-620.6449	86.62484	460.5861	-622.4988	117.1428	465.14	-622.8572
118.8929	464.0123	-609.088	55.64438	455.9936	-643.8726	91.03967	460.7456	-622.2564	85.81608	460.4234	-622.6894	111.6754	465.8805	-625.3742
108.9535	463.261	-612.4579	52.7723	455.5503	-644.6869	88.87242	459.8597	-623.0433	86.48247	460.5194	-622.2526	129.4723	464.827	-614.3333
101.202	462.3537	-616.6434	49.03302	455.017	-645.5925	87.55943	459.8194	-623.4074	92.96282	461.4527	-620.9545	140.6941	465.7095	-604.3356
99.62507	462.2069	-617.2965	48.10916	454.8853	-645.5821	80.16634	459.3922	-626.4701	101.3876	462.6661	-617.8952	136.338	466.8435	-602.937
99.49906	462.6255	-617.3735	50.69213	455.2853	-641.8341	79.17791	459.3164	-628.2327	114.9748	464.623	-613.7295	131.6091	467.9217	-607.07
100.7742	462.5778	-617.5662	53.8968	455.7437	-637.8332	78.47113	459.237	-628.1204	130.2393	466.5036	-608.7805	130.4684	468.3637	-607.1663
100.192	462.4939	-617.7454	57.6157	456.1723	-635.8058	79.70029	459.218	-627.9077	143.1737	468.3405	-603.4202	133.4158	467.3115	-606.6871
99.96435	462.4611	-618.4889	63.23801	456.9148	-633.6949	80.42239	459.3416	-627.0821	164.6993	471.3976	-595.9346	135.6625	468.263	-605.1061
106.2033	463.3597	-618.2669	69.12411	457.746	-632.1438	91.66204	461.4343	-624.7494	181.1788	473.738	-587.3252	168.8761	479.5543	-596.18
118.3496	465.168	-614.8201	75.84732	458.4672	-629.5862	109.978	464.098	-617.8187	208.4059	477.6048	-574.9506	222.5259	485.1506	-566.4199
130.4448	466.6918	-610.2299	86.61479	459.964	-626.0497	126.0177	465.7496	-612.438	229.4613	480.1309	-560.1864	277.2023	488.0492	-527.494
154.6533	471.0744	-606.8457	99.82004	461.7997	-621.7161	147.8275	468.6492	-604.056	251.0167	483.0416	-545.5203	315.0152	492.5392	-502.0261
202.1811	483.3351	-601.6983	120.4426	464.6666	-614.9077	168.799	471.7432	-593.917	264.2292	484.8889	-535.5025	326.4256	493.7979	-489.8741
264.2158	487.7658	-564.0592	138.0599	467.1157	-607.433	184.5832	473.9659	-586.1838	277.3371	487.1496	-527.7325	348.6338	501.3874	-480.2715
305.413	488.0533	-526.0925	159.5669	469.9182	-598.5763	210.0123	477.9762	-574.9466	296.669	489.9447	-514.36	371.6914	504.3329	-467.2645
342.4286	494.1941	-496.7848	176.4625	472.3317	-590.0469	230.0826	480.8905	-560.934	311.9901	492.1114	-498.0038	382.6707	499.9712	-442.8151
358.0102	497.9238	-473.2177	196.4023	475.1813	-579.8327	253.9865	483.7956	-545.3354	329.2248	494.5487	-479.6724	389.3012	498.7953	-432.8194
358.3846	494.5083	-459.6409	219.0465	478.3246	-566.0313	282.6601	487.9013	-525.3241	338.6577	493.9029	-468.4493	383.2192	501.6903	-425.3313
354.385	497.3411	-462.8501	240.9168	481.4143	-551.0542	316.8365	494.0427	-500.1468	345.9802	496.5437	-461.2208	379.4143	504.0196	-410.5632
350.5852	497.7972	-467.8535	261.2506	483.49	-536.1358	337.0214	494.0447	-476.2851	357.753	498.274	-447.4383	382.3872	507.5227	-396.5248
349.5494	499.6985	-465.6492	273.2857	485.1318	-526.6052	344.7867	492.436	-469.0659	366.3317	499.7964	-433.2294	386.2596	506.6978	-388.4469
349.5001	500.6044	-461.5895	282.6769	486.7229	-521.4233	346.0642	496.4805	-465.2352	379.6327	501.6774	-414.4229	395.1364	505.4834	-374.1523
353.6424	500.6864	-456.3418	290.5537	487.9962	-516.3468	350.4477	500.1359	-457.4237	384.7278	501.9018	-401.9146	400.3134	507.2692	-372.5307
362.8984	502.2967	-441.871	306.1503	490.4159	-503.4821	358.5748	501.4964	-444.7236	383.6491	497.1538	-408.2133	402.5807	507.2573	-369.2707
372.1783	502.3102	-410.8708	317.5207	492.1325	-490.452	368.9529	501.7606	-429.5751	372.7663	497.5831	-429.4694	415.5974	508.8117	-363.6844
374.4646	500.7852	-407.9529	328.0023	493.5917	-477.8002	381.9843	501.2709	-413.0404	371.9271	501.2293	-446.3806	422.9037	509.788	-355.0498
375.4578	500.9253	-407.0904	336.2495	494.4485	-467.5538	393.0817	503.0732	-389.2944	373.1025	501.8805	-450.7516	427.3259	509.2439	-344.7972
377.0461	501.474	-405.821	340.4913	496.5115	-460.8446	396.9042	503.6943	-375.6604	386.0687	501.8412	-442.616	431.1269	508.9596	-334.0176
381.9537	502.17	-401.8381	346.4876	497.5157	-453.1362	407.0234	505.3766	-370.678	417.1492	508.3226	-407.5695			
389.2074	503.1987	-397.7133	357.7229	499.1978	-443.208	413.9106	506.4364	-371.9656	431.3634	511.0217	-358.6349			
392.5509	503.7572	-395.354	369.3191	500.936	-426.1984	420.9855	507.1656	-372.1617						
397.9609	504.5255	-393.6967	379.1295	502.2594	-412.7953	428.8267	508.5428	-362.7463						
401.854	504.9061	-388.6611	387.9454	502.2697	-403.668	433.6568	509.1324	-348.7289						
406.4679	505.6466	-379.4062	390.8409	502.5913	-400.231	438.4394	509.8068	-332.7229						
411.5563	506.3682	-370.1256				440.6103	510.113	-324.0883						
414.9443	506.8487	-361.6385				441.1388	510.1875	-321.9479						
418.5542	507.4502	-354.1997												
421.3738	507.8506	-347.8793												



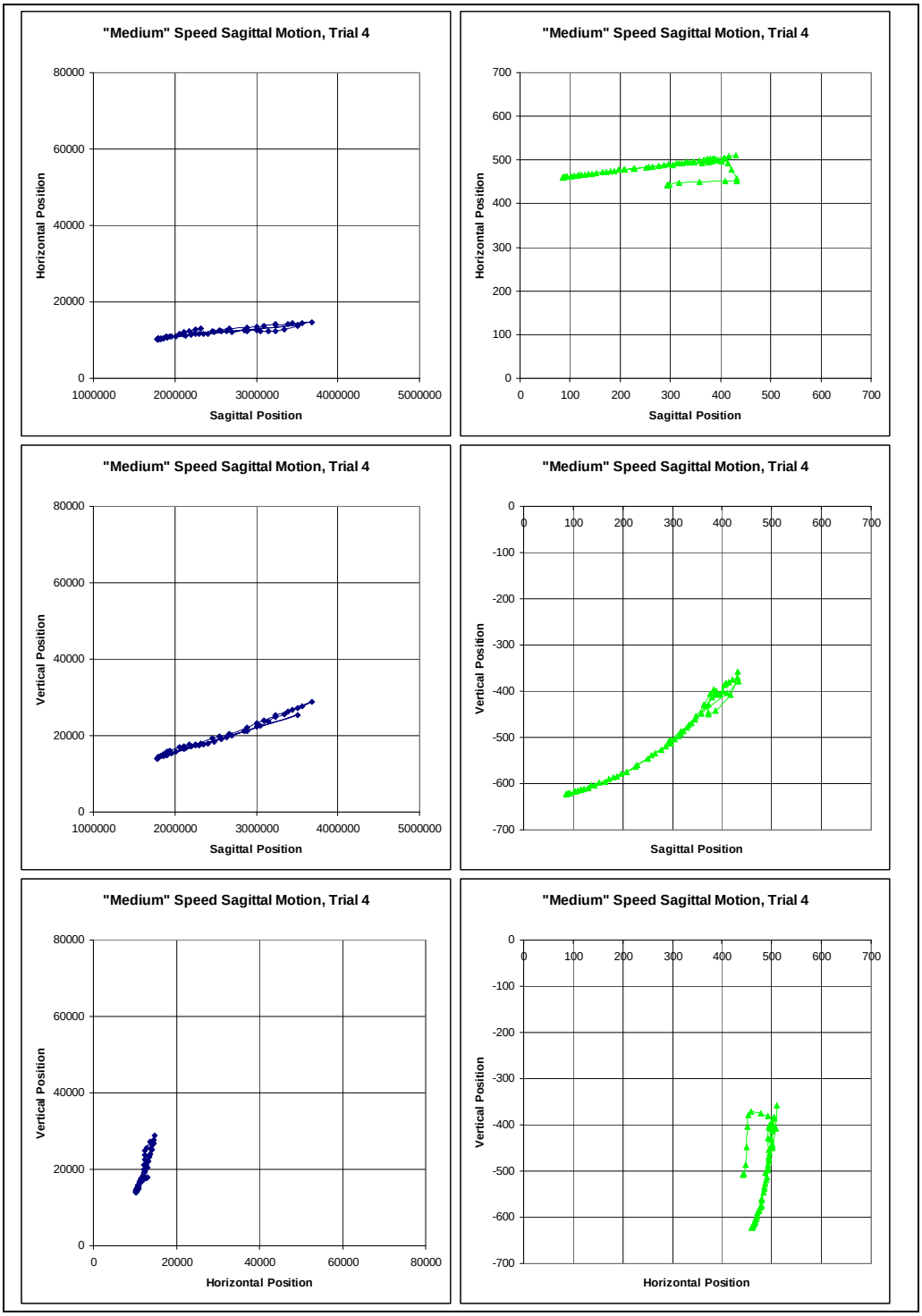
"Medium" speed sagittal motion, trial 1; human motion is shown on the left, that of ISAC on the right



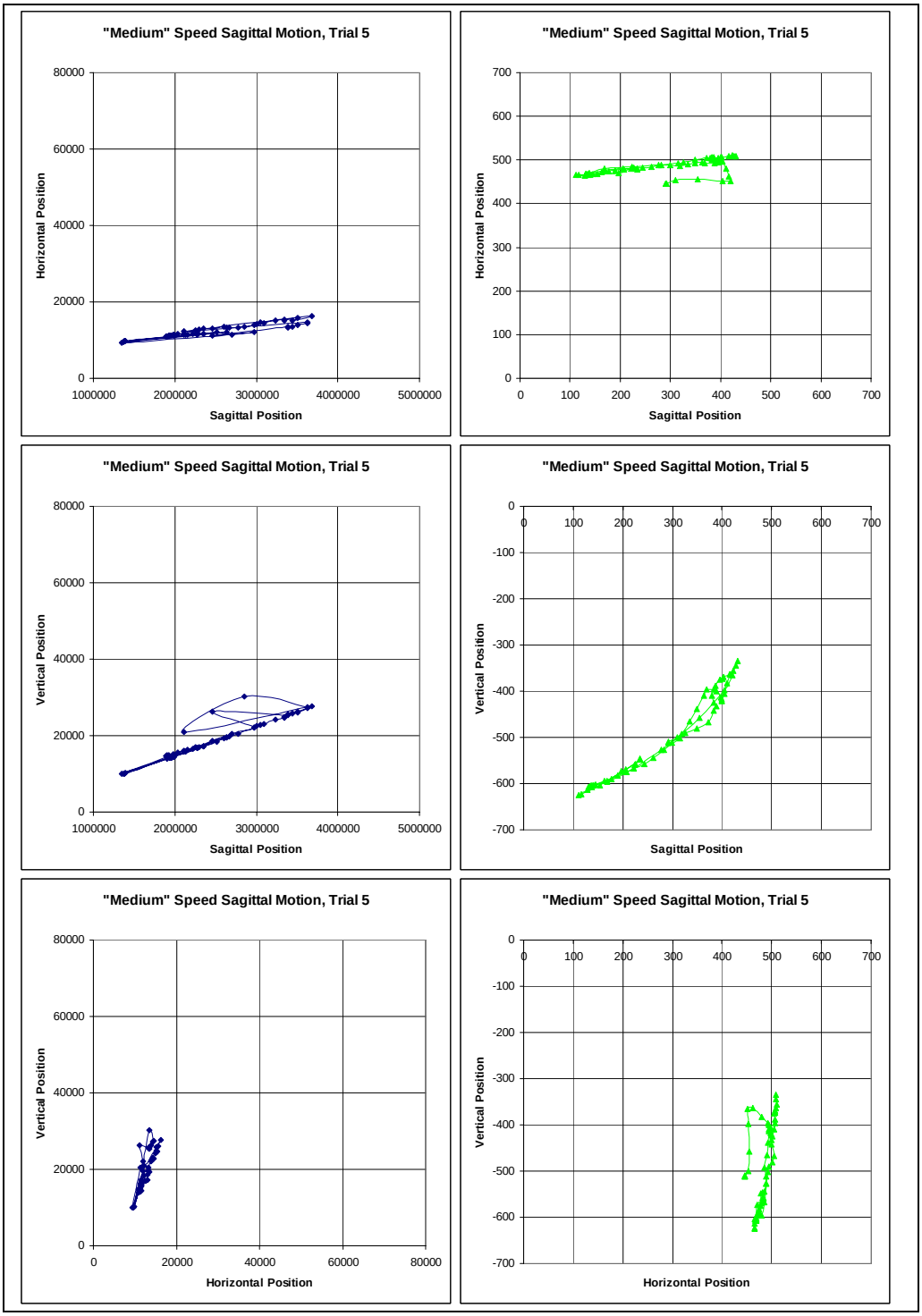
"Medium" speed sagittal motion, trial 2; human motion is shown on the left, that of ISAC on the right



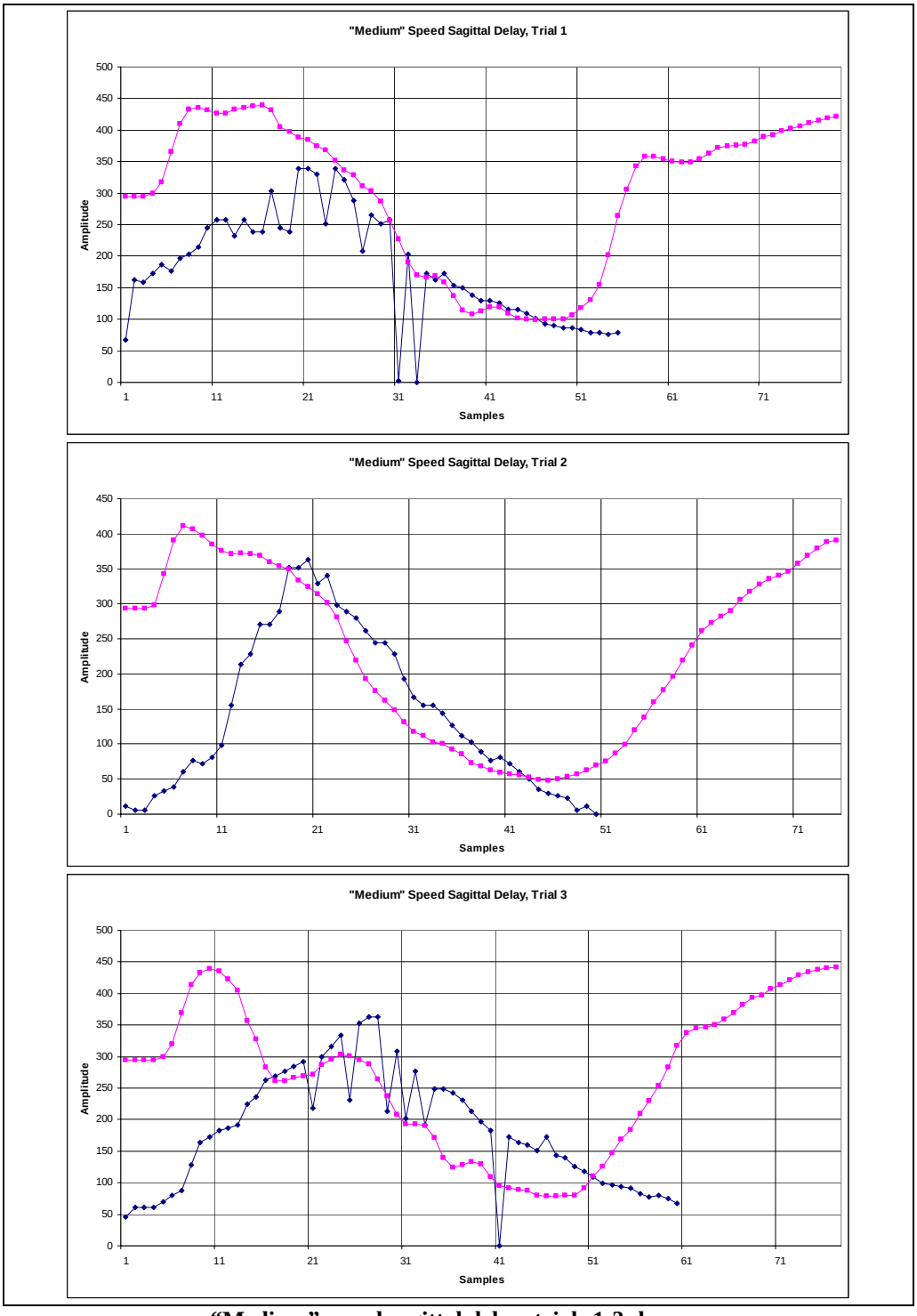
"Medium" speed sagittal motion, trial 3; human motion is shown on the left, that of ISAC on the right



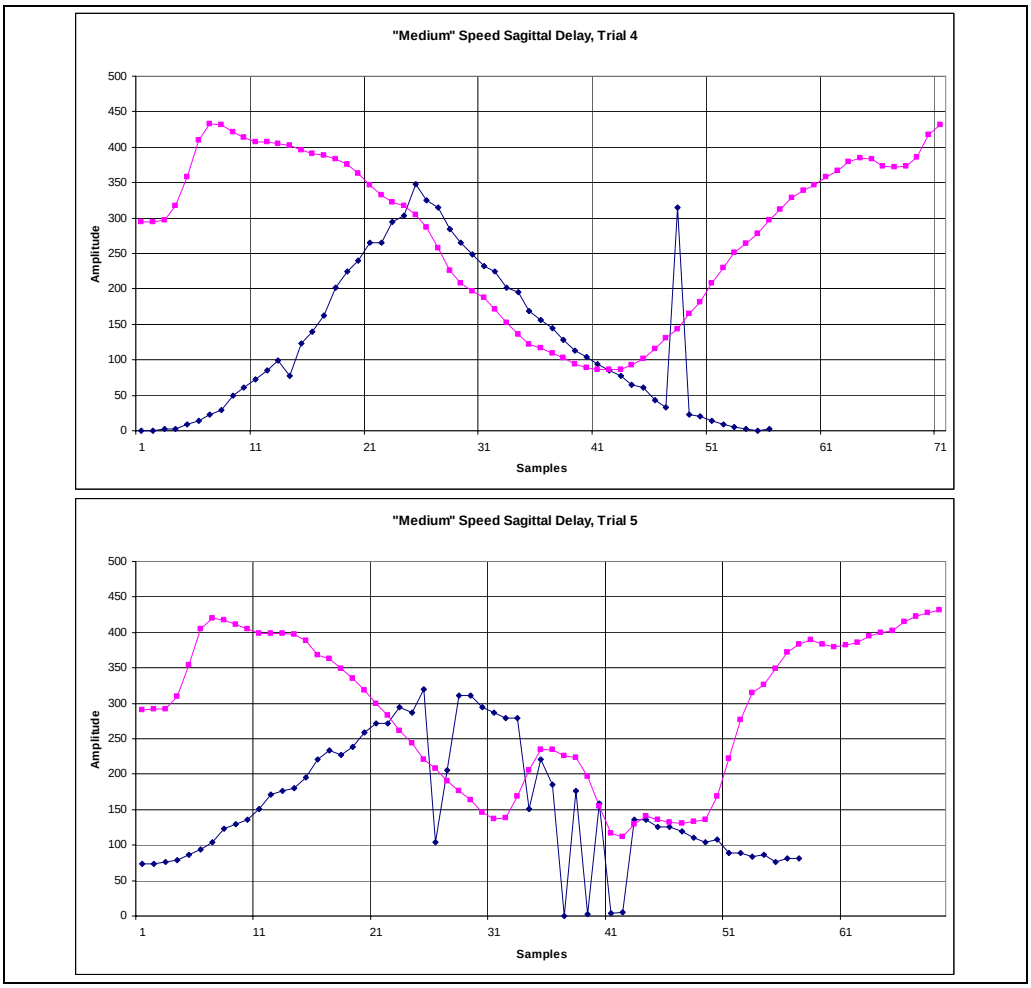
"Medium" speed sagittal motion, trial 4; human motion is shown on the left, that of ISAC on the right



"Medium" speed sagittal motion, trial 5; human motion is shown on the left, that of ISAC on the right



“Medium” speed sagittal delay, trials 1-3; human motion is shown on the on bottom initially, that of ISAC on the on top initially



“Medium” speed sagittal delay, trials 4-5; human motion is shown on the on bottom initially, that of ISAC on the on top initially

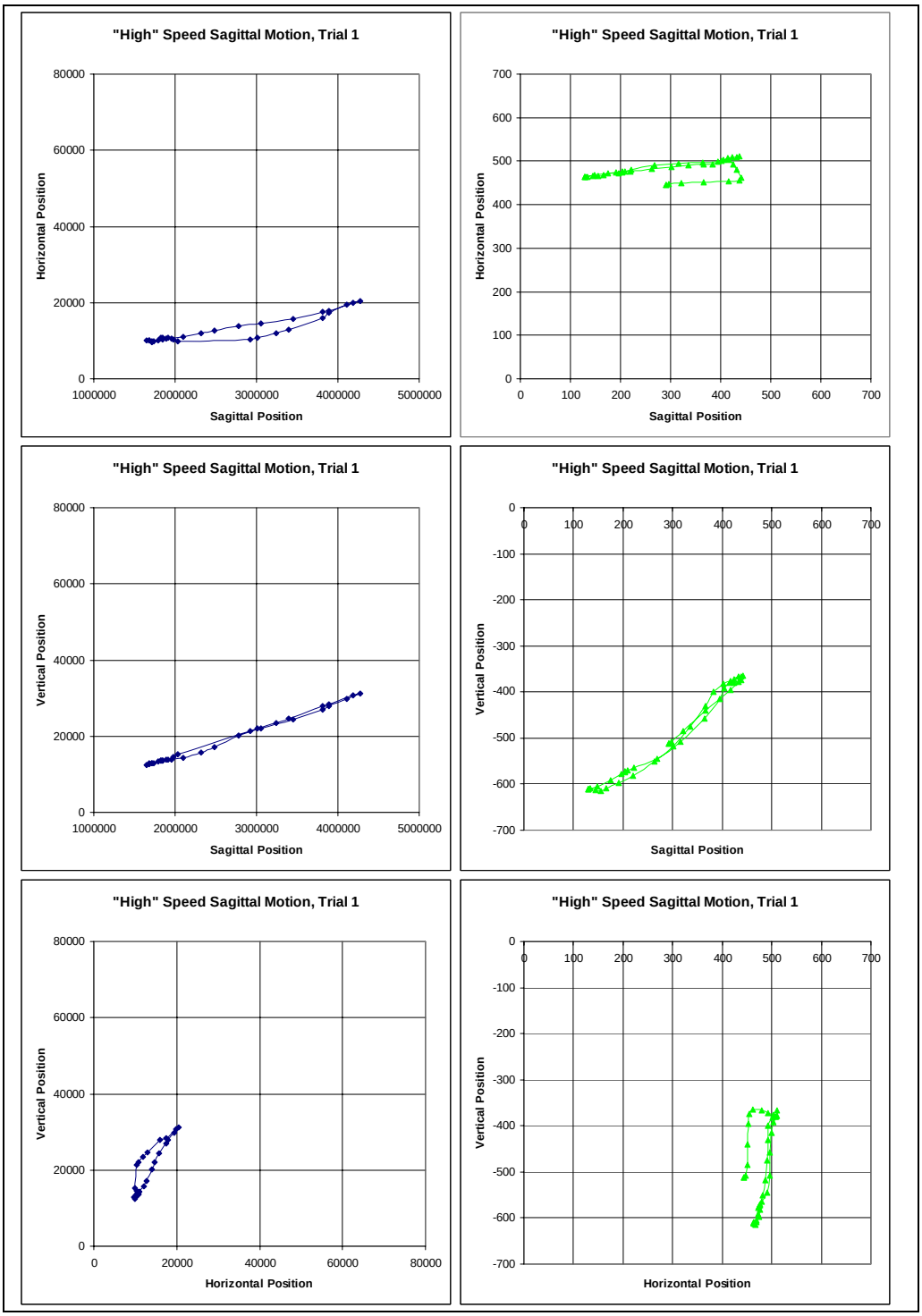
Sagittal motion at "High" speed

Human

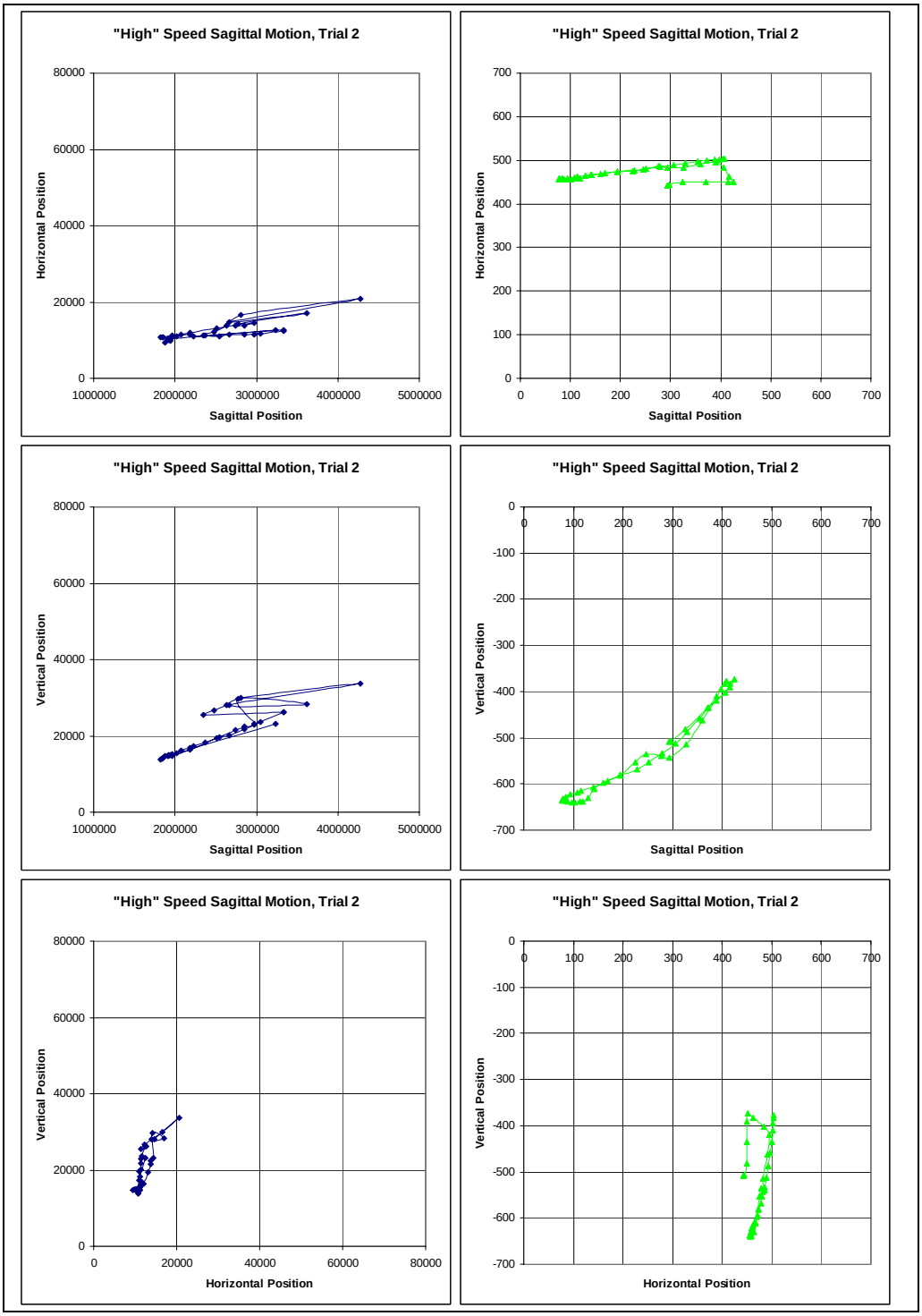
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
1826282	10709.4	13521.37	1826282	10829.06	13820.51	1594590	10134.33	11753.73	1656395	10581.4	12100.78	1643654	10984.62	12061.54
1842026	10741.38	13698.28	1842026	10741.38	13939.66	1606579	10052.63	11789.47	1631107	10206.11	12022.9	1656395	10906.98	12100.78
1826282	10470.09	13581.2	1858044	10773.91	14243.48	1606579	9947.368	11736.84	1669336	10390.63	12250	1682480	10858.27	12070.87
1907813	10750	13875	1960321	11238.53	14642.2	1656395	10255.81	12046.51	1723186	10048.39	12588.71	1765909	11396.69	12438.02
1960321	10596.33	13743.12	2180357	12000	16428.57	1695833	10444.44	12055.56	1842026	10258.62	13336.21	1874342	11789.47	13324.56
2094853	10980.39	14343.14	2513824	13094.12	19352.94	1858044	11260.87	13147.83	1960321	10467.89	14192.66	2015802	12547.17	14000
2322554	12021.74	15673.91	2739423	13820.51	21448.72	1960321	11880.73	13678.9	2158333	11525.25	15484.85	2136750	13020	14840
2484593	12697.67	17011.63	2849000	13720	22400	1996963	12102.8	14130.84	2249211	11568.42	16357.89	2225781	13416.67	15385.42
2775000	13909.09	20181.82	2967708	14388.89	23138.89	2158333	12797.98	15696.97	1318982	9938.271	9808.642	2400843	14078.65	16516.85
3052500	14600	22100	2775000	14272.73	29727.27	2273138	12808.51	17053.19	1318982	9938.271	9808.642	2400843	13606.74	16359.55
3446371	15806.45	24500	3621610	16966.1	28355.93	2967708	14388.89	21777.78	2637963	12876.54	19617.28	2513824	13917.65	17458.82
3815625	17500	27000	2670938	14700	28000	3096739	14710.14	22420.29	3009507	14098.59	22478.87	2605793	13829.27	18353.66
3885000	17945.46	27872.73	4273500	20720	33600	3237500	15696.97	23121.21	3096739	14304.35	22724.64	2704747	13911.39	19227.85
4109135	19384.62	29750	2811513	16578.95	29842.11	3502869	17327.87	25704.92	3338672	15312.5	24390.63	2887500	14567.57	20810.81
4273500	20440	31220	2637963	13913.58	28086.42	3885000	17945.46	29400	3287308	14969.23	24015.38	2967708	14583.33	21388.89
4189706	19901.96	30745.1	2484593	12209.3	26779.07	3815625	16750	28500	3621610	16016.95	26694.92	3096739	15115.94	22217.39
3885000	17436.36	28381.82	2348077	11307.69	25461.54	3561250	14933.33	27183.33	3621610	15779.66	26694.92	3237500	15909.09	23439.39
3815625	16000	27875	3338672	12687.5	26250	2273138	11765.96	24127.66	3956945	17370.37	28907.41	3391667	16555.55	24333.33
3391667	13000	24666.67	3338672	12468.75	26250	3142280	12970.59	23676.47	4031604	17830.19	29056.6	3391667	16333.33	24555.55
3237500	11878.79	23439.39	2967708	11472.22	22847.22	3096739	12884.06	23231.88	2297581	11064.52	24010.75	3621610	16966.1	26576.27
3009507	10746.48	21985.92	3052500	11600	23600	2887500	12297.3	21283.78	2180357	10142.86	22642.86	3621610	16966.1	26694.92
2927055	10260.27	21383.56	2849000	11480	21746.67	2811513	12526.32	20355.26	2849000	10546.67	20066.67	3748684	17070.18	27631.58
2035000	9933.333	15266.67	2670938	11375	20212.5	2574398	11891.57	18216.87	2704747	10544.3	18962.03	3621610	16254.24	26694.92
1978472	10370.37	14583.33	2543750	11000	19583.33	2543750	12500	17750	2513824	10788.24	17458.82	3502869	14803.28	25934.43
1890929	10469.03	13814.16	2374167	11200	18355.55	2273138	11319.15	16085.11	2400843	11247.19	16516.85	3338672	14000	24828.13
1842026	10379.31	13517.24	2225781	10937.5	17208.33	2180357	11000	15500	1996963	10401.87	13738.32	3189179	13268.66	23402.98
1795588	10058.82	13294.12	2180357	11571.43	16928.57	2015802	10433.96	14594.34	1890929	10221.24	13132.74	3189179	13268.66	23298.51
1737195	9845.528	12975.61	2074515	11349.51	16106.8	1907813	10125	13875	1826282	10230.77	12743.59	3009507	13309.86	21690.14
1709400	9688	12936	2015802	11094.34	15518.87	1858044	10408.7	13817.39	1695833	9666.667	12000	2849000	12973.33	20346.67
1709400	9912	12880	1960321	10724.77	15155.96	1765909	10355.37	13190.08	1631107	9778.626	11541.98	2811513	13078.95	20078.95
1682480	9976.378	12622.05	1925000	10279.28	15072.07	1682480	10196.85	12511.81				2605793	13658.54	18524.39
1682480	10086.61	12787.4	1874342	9456.141	14675.44	1606579	10052.63	11894.74				2456035	13114.94	17459.77
1643654	10015.38	12492.31	1942500	9927.272	15018.18	1559672	9759.124	11291.97				2273138	12510.64	16531.91
			1925000	10153.15	14819.82	1494231	9643.356	10720.28				2225781	12395.83	16333.33
			1907813	10250	14625	1483854	9819.444	10597.22				2115594	11990.1	15801.98
			3237500	12515.15	23227.27	1443750	9743.243	10263.51				2015802	11226.42	15056.6
						1453571	9761.904	10428.57				1960321	11110.09	14642.2
						1443750	9837.838	10500				1890929	10840.71	14185.84
												1810805	10677.97	13288.14
												1765909	10818.18	12785.12
												1737195	10983.74	12463.42
												1723186	11064.52	12193.55
												1682480	10858.27	11795.28
												1682480	10968.5	11740.16
												1656395	10689.92	11775.19
												1656395	10689.92	11883.72

ISAC

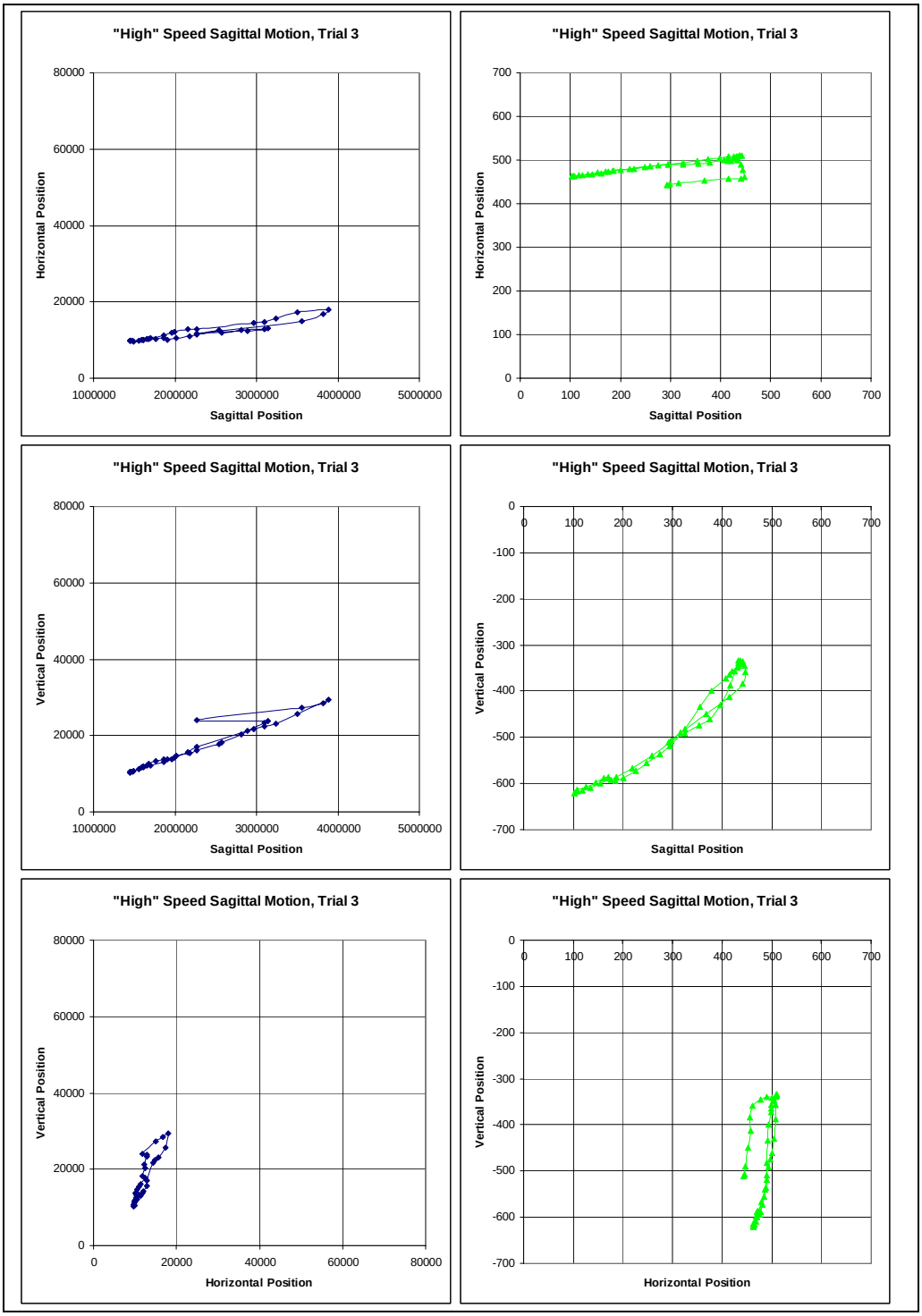
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
291.7569	444.6264	-512.2414	293.0633	442.6159	-508.8235	292.403	443.6509	-510.9084	293.1159	443.3016	-509.1179	293.8305	441.9201	-508.1618
291.7946	444.5691	-512.1047	293.4397	442.6115	-508.5147	292.4167	443.5365	-510.8971	293.2561	443.1858	-509.0607	293.9075	441.8046	-508.1618
297.309	446.8771	-508.0193	296.9779	444.4273	-505.7716	298.0685	445.5317	-507.3162	293.767	444.0994	-508.2675	294.8054	442.5957	-506.9457
321.1645	450.3729	-484.5973	324.9416	450.0379	-482.148	315.9529	447.0565	-489.8287	308.774	449.8989	-498.1985	315.2954	449.9184	-492.5402
366.0212	451.9238	-440.7968	370.222	449.7201	-434.8559	367.4178	452.5293	-449.1672	362.5496	454.8221	-458.1307	372.2727	452.1721	-449.6099
416.7372	453.732	-396.2824	414.7816	450.2485	-389.6837	414.8817	457.9873	-413.2292	416.5155	456.1266	-418.4605	422.2994	449.0754	-403.3833
437.9228	455.4263	-375.0412	425.2615	451.1957	-373.0914	440.79	456.6258	-382.9492	444.8651	452.842	-380.2373	444.3079	447.6601	-367.5925
441.1821	462.5292	-365.2494	416.1725	463.2687	-383.2482	447.2862	461.8618	-359.2493	449.2191	460.8533	-351.2728	442.0782	462.7394	-348.5588
432.7938	479.3758	-365.6522	406.9943	483.79	-402.2657	444.3572	477.9855	-346.0749	439.9885	478.5733	-342.7217	433.6571	480.7727	-346.0497
424.6078	492.8846	-372.6817	389.3605	494.5846	-419.6004	440.4072	489.8284	-339.782	432.0568	491.1675	-352.9118	425.9587	497.1328	-351.3027
416.44	503.0642	-376.6898	359.545	491.2819	-461.7718	434.2323	500.6553	-340.5834	423.4123	502.3735	-359.6779	423.4281	505.0723	-358.1406
402.4872	500.9539	-381.2085	326.6488	482.6117	-514.7352	432.0353	503.5791	-344.9399	422.359	506.1871	-357.1797	423.6473	504.1113	-360.0816
383.3128	493.3655	-400.0649	293.0365	482.0424	-543.0365	430.0487	503.322	-348.6276	423.3865	505.5172	-353.4478	420.6621	502.5533	-364.3359
366.9165	492.102	-429.5928	276.4078	486.1655	-540.2573	420.2331	498.144	-355.9213	423.4437	502.7365	-356.4077	414.6117	496.2431	-384.2577
334.8848	489.8555	-474.9276	246.4704	478.0463	-536.0855	414.158	498.635	-365.3878	409.6491	494.1073	-371.0157	401.9302	496.7856	-402.6442
301.4615	486.4851	-516.8958	224.2187	475.5393	-553.7535	407.5367	499.2846	-372.6541	376.4355	489.9946	-408.3948	382.6811	497.6001	-417.6874
263.0044	481.5312	-551.5641	193.5097	473.0174	-580.3556	377.9792	493.1269	-399.2303	334.7253	492.1393	-467.436	365.9105	495.413	-430.3287
220.2174	476.2161	-582.3872	159.9678	469.3374	-598.2218	355.0793	491.8735	-434.3805	295.4015	492.2368	-519.0813	346.3559	492.8628	-452.5141
191.7843	473.7951	-598.1311	142.5656	467.5038	-611.6866	324.675	489.5453	-482.0032	261.5722	485.5041	-556.9108	322.6315	489.1425	-483.5404
166.2849	468.6133	-609.8228	130.355	463.916	-630.6232	294.1227	489.9722	-519.1707	237.3038	480.594	-574.0583	301.4287	486.285	-509.3307
154.8461	466.7726	-615.0842	119.0416	458.9498	-638.3023	275.4153	488.4017	-536.7164	220.496	477.8811	-579.5283	272.1536	483.5434	-535.3994
143.8712	465.2077	-612.8611	114.0842	457.922	-638.4391	248.4716	484.0722	-554.7999	191.3615	474.175	-589.9536	246.8304	481.1943	-555.7217
133.8334	463.8782	-609.7881	101.8692	457.2266	-639.7961	226.9071	480.1301	-571.8213	166.68	471.757	-600.8698	217.8873	477.6728	-571.6873
129.3214	463.3258	-611.4853	96.45575	456.8685	-640.3996	201.2013	477.134	-587.3657	139.2787	468.1234	-612.3013	193.296	475.1928	-584.6447
129.1641	463.429	-611.4539	87.65239	456.014	-638.4378	184.6264	474.9969	-594.748	127.3995	466.8826	-616.8357	179.724	473.4065	-591.2238
129.9576	463.5268	-611.0541	84.87256	456.6572	-636.8724	175.7836	473.545	-591.7313	121.5095	465.9863	-617.2529	159.7484	469.9807	-592.8935
148.0839	468.5091	-605.0287	77.53816	456.9658	-635.6112	171.1287	472.8708	-586.5255	113.6657	463.6397	-618.5121	152.0187	469.4884	-596.3903
174.3118	472.7222	-591.1826	77.01262	457.3276	-634.0894	162.2357	470.0211	-588.1846	110.5234	463.7869	-619.3877	148.2991	469.6	-598.0821
196.0533	472.4207	-578.3197	78.42446	457.4854	-632.569	144.5307	467.6367	-598.0571	110.8168	464.08	-618.8534	146.2803	469.5532	-598.1836
204.0219	473.9793	-573.241	84.12974	458.1402	-627.8303	124.8468	465.8604	-606.6756	118.8282	465.5333	-616.4769	148.8794	469.7911	-597.194
204.2812	475.9967	-572.9682	93.6929	459.3652	-623.0393	108.4551	463.684	-613.6802	139.9796	468.5627	-607.8298	152.9863	470.6448	-596.4
209.3902	476.9368	-570.2997	108.2376	461.3816	-617.9543	102.7458	463.0726	-620.7241	155.0624	469.1505	-597.6084	168.2127	475.238	-590.6983
221.1794	479.769	-564	114.331	462.4079	-615.8729	102.4414	463.0283	-620.7675	172.8667	472.3562	-590.6603	197.0872	479.5043	-574.6119
268.6468	491.1365	-544.1795	140.8777	466.8011	-607.7731	106.9768	462.8486	-617.971	192.6403	475.9419	-580.2951	221.47	479.2102	-560.6346
316.0342	495.4507	-508.4853	169.5733	471.2992	-592.7961	117.7151	464.8419	-614.2247	214.871	479.2548	-567.3906	241.1334	483.2261	-545.8488
365.0592	496.0837	-457.9649	194.7646	474.11	-581.4782	135.41	468.1926	-608.5773	256.3961	485.5632	-546.7381	256.7959	486.8856	-534.7195
394.3351	499.6239	-415.5554	228.0162	477.8868	-567.9696	154.2745	470.6135	-599.6291	290.3864	489.8228	-521.7187	276.5966	489.7415	-528.343
404.8414	503.856	-391.2156	251.5547	480.9433	-552.5598	186.0122	475.2418	-585.7019	323.2084	490.0534	-492.5678	306.2172	494.7263	-513.0748
414.383	507.0352	-380.5671	279.3804	485.4729	-533.948	219.4013	480.111	-566.1895	340.433	493.8504	-471.1025	328.5365	497.3911	-488.6742
423.9232	508.6659	-380.0212	306.385	489.5802	-511.5648	258.8909	485.5812	-539.5215	361.4997	499.113	-452.1912	354.4815	497.278	-458.0878
432.9865	510.0547	-377.582	329.5566	492.7364	-487.6091	296.2578	489.8216	-509.1022	405.2048	502.9541	-407.651	372.87	499.5974	-428.6012
437.8858	510.7564	-366.631	353.9611	496.9	-458.8015	325.7538	493.6327	-491.4693	415.2158	497.7139	-389.3403	390.9642	502.357	-398.3534
			372.2066	499.4254	-435.5464	354.1697	498.0764	-474.7677	418.5191	504.2304	-377.9287	402.5567	503.9708	-378.6557
			388.6768	501.2054	-410.4626	375.742	501.37	-460.4143				414.993	505.7022	-369.8231
			397.0458	502.3537	-393.5161	397.5571	504.4682	-429.2256				428.2653	508.0065	-364.512
			404.8953	503.8635	-382.3729	416.5759	507.1692	-386.9841				433.9424	509.1727	-359.9423
			408.49	504.3603	-378.0149	426.0605	507.3338	-357.0161				437.5783	510.4321	-351.1134
						431.9346	508.1527	-339.6803				440.1031	512.2021	-340.6539
						433.1873	509.1586	-334.0455				441.8353	512.5488	-334.713
						436.7412	509.9399	-334.382						
						439.2459	510.2952	-337.0359						
						441.6561	510.5428	-334.7689						



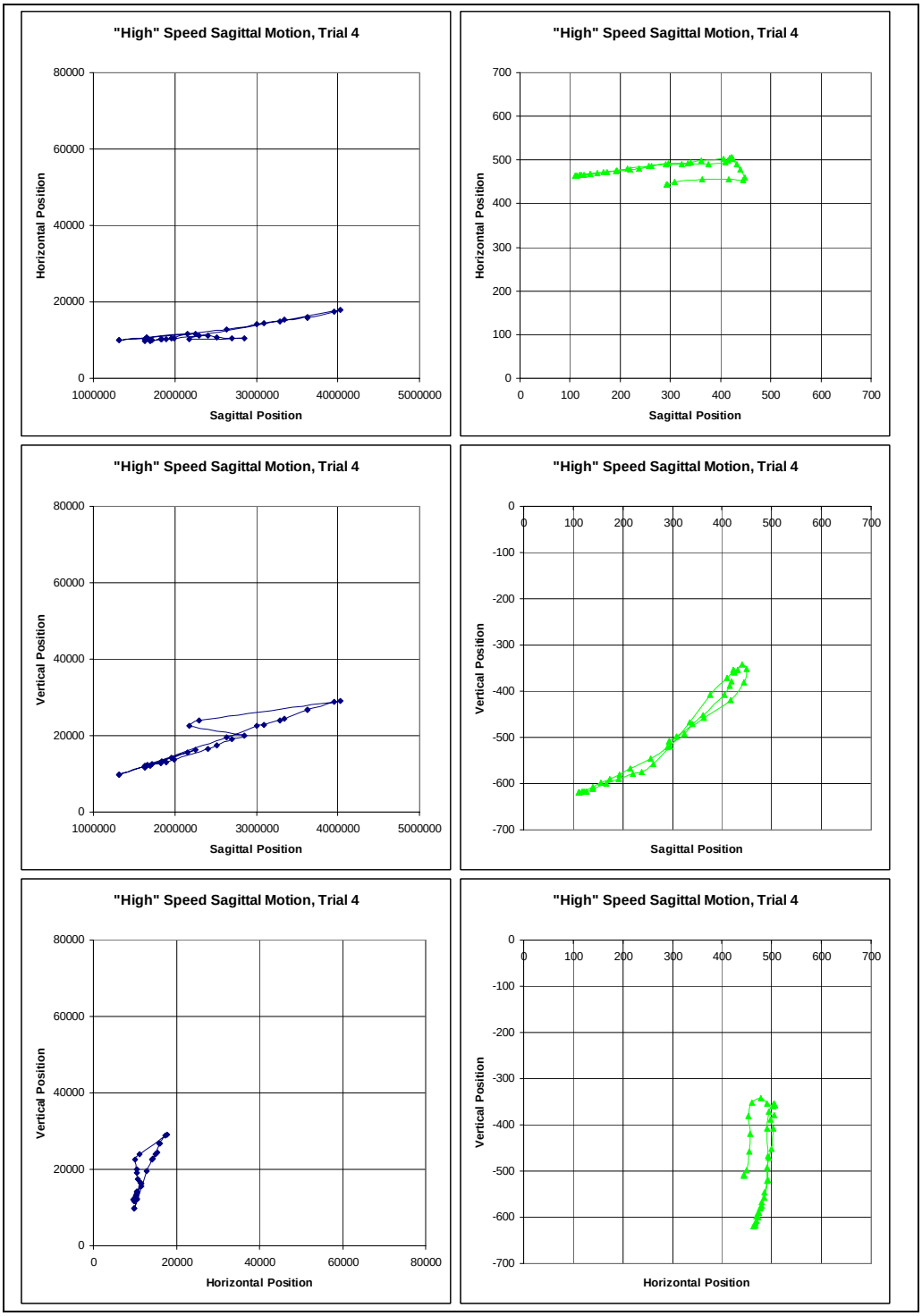
"High" speed sagittal motion, trial 1; human motion is shown on the left, that of ISAC on the right



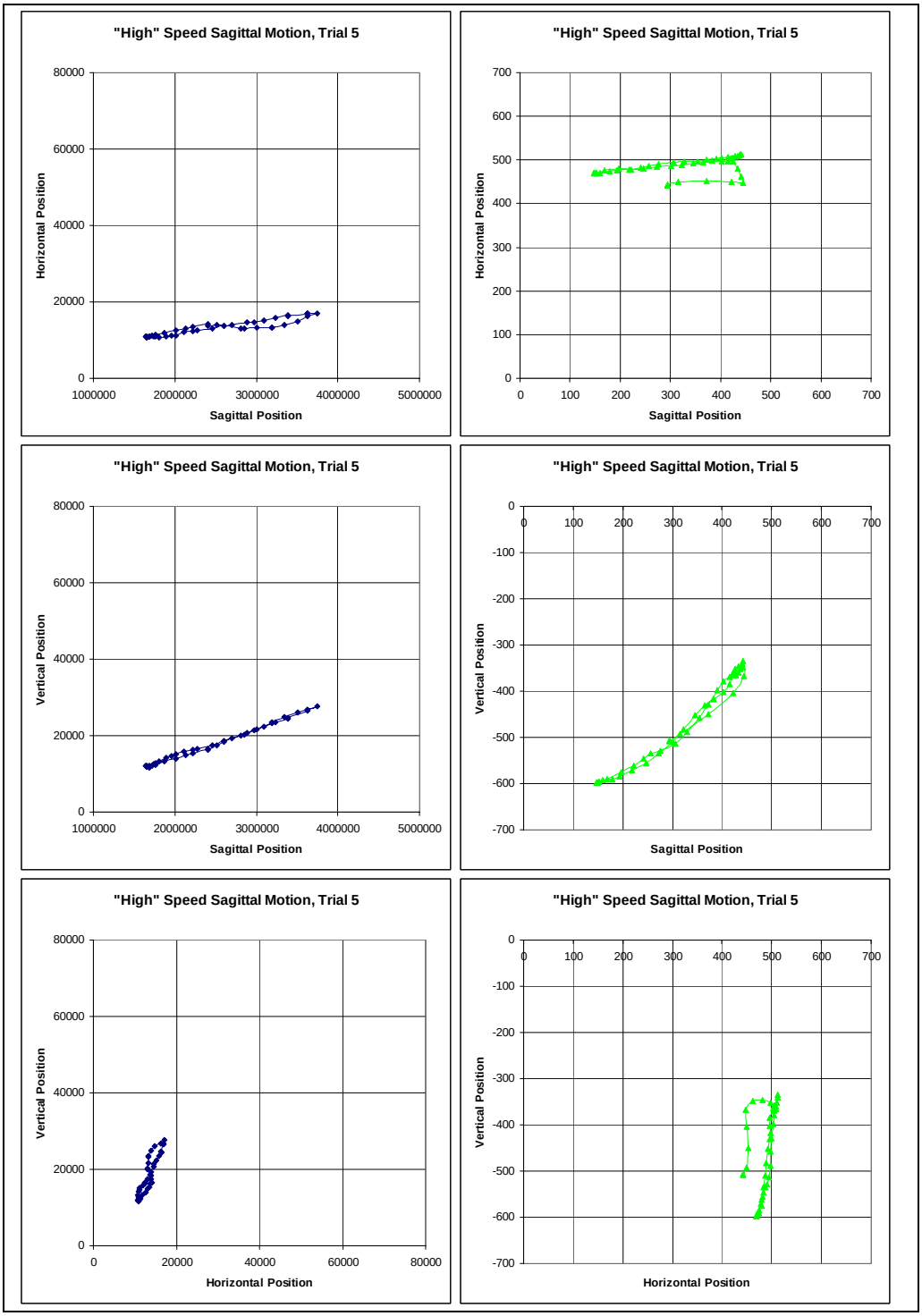
"High" speed sagittal motion, trial 2; human motion is shown on the left, that of ISAC on the right



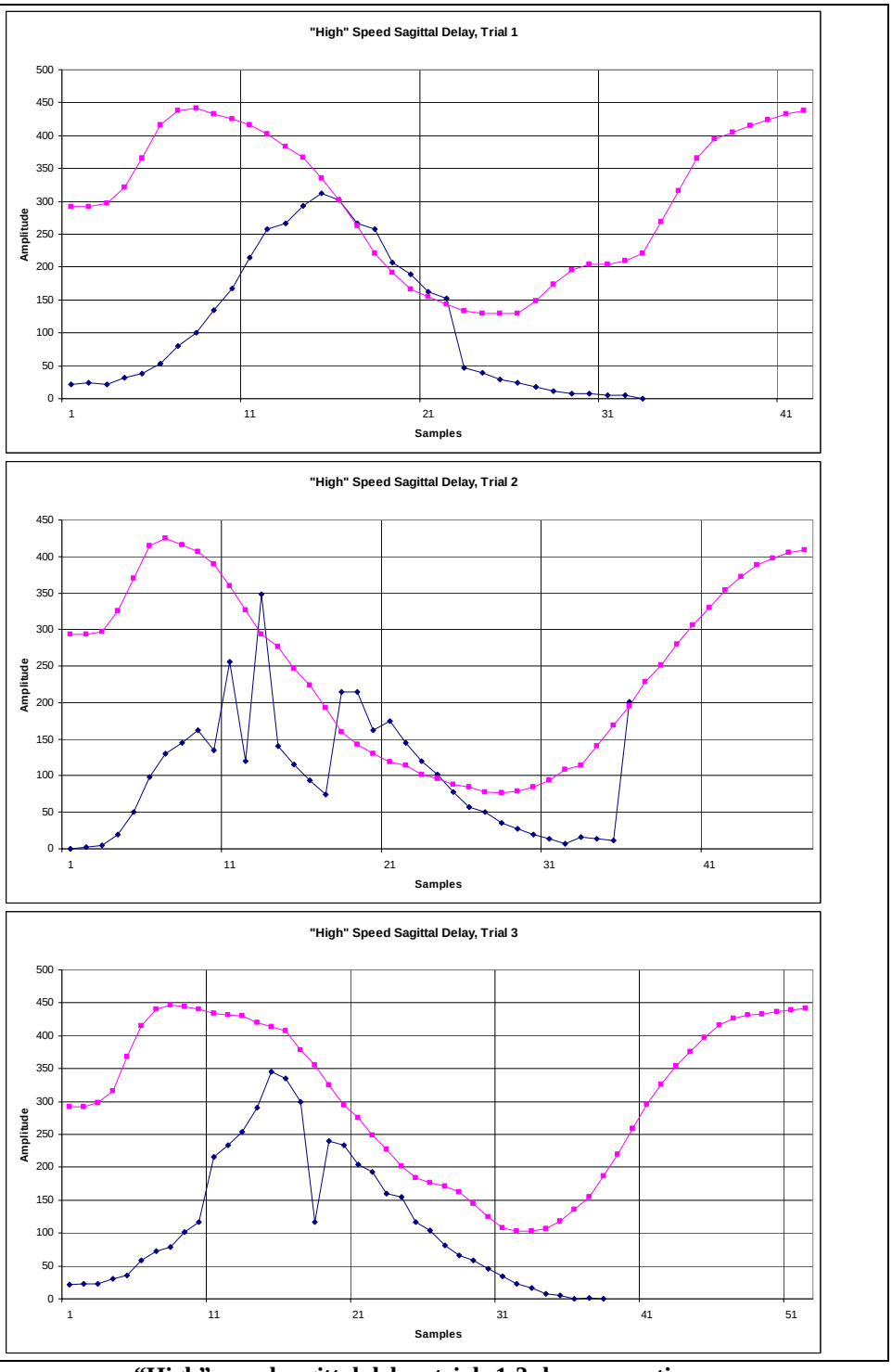
"High" speed sagittal motion, trial 3; human motion is shown on the left, that of ISAC on the right



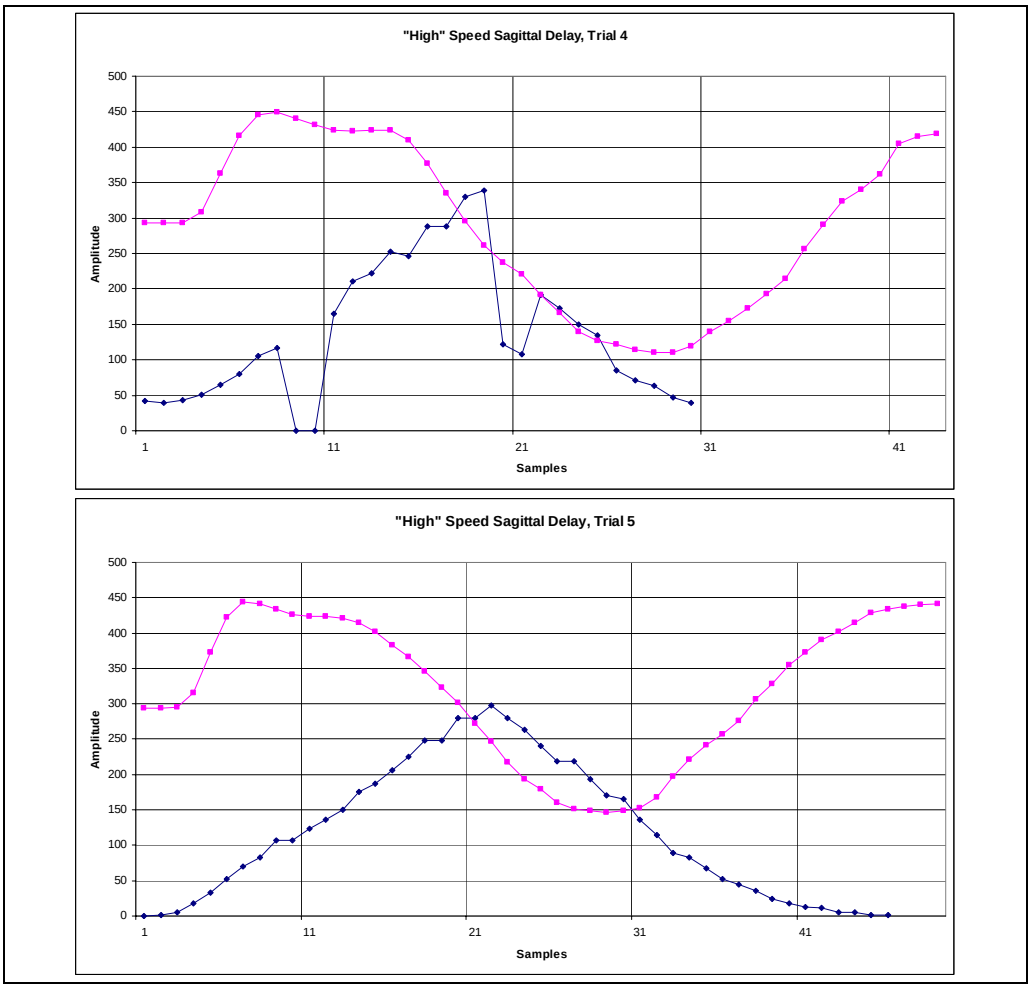
"High" speed sagittal motion, trial 4; human motion is shown on the left, that of ISAC on the right



"High" speed sagittal motion, trial 5; human motion is shown on the left, that of ISAC on the right



"High" speed sagittal delay, trials 1-3; human motion is shown on the on bottom initially, that of ISAC on the on top initially



“High” speed sagittal delay, trials 4-5; human motion is shown on the on bottom initially, that of ISAC on the on top initially

Diagonal motion at "Low" speed

Human

Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
3684052	16172.41	14120.69	3561250	19133.33	12950	3621610	18389.83	13525.42	3748684	21245.61	13754.39	3502869	18245.9	13311.48
3684052	16172.41	14120.69	1559672	12313.87	10576.64	3748684	19280.7	13877.19	3684052	21000	13275.86	3446371	18064.52	13096.77
3748684	16333.33	14491.23	3621610	19338.98	13169.49	3561250	18200	13883.33	3684052	21241.38	13758.62	3561250	18666.67	13650
1631107	12129.77	9832.062	3621610	19338.98	13050.85	3561250	18666.67	13766.67	3621610	21000	13644.07	3446371	18064.52	13209.68
1643654	12276.92	9800	1571140	12455.88	10654.41	3561250	20533.33	15866.67	3885000	22781.82	14636.36	3502869	18245.9	13540.98
3502869	18704.92	16754.1	1571140	12455.88	10757.35	3561250	20300	15983.33	3748684	22228.07	14491.23	3391667	17666.67	13111.11
3561250	19366.67	17733.33	3621610	19813.56	13525.42	3502869	20081.97	15606.56	3748684	22719.3	14859.65	3502869	18475.41	13770.49
3446371	19193.55	17048.39	3502869	19622.95	13311.48	3561250	20300	16216.67	3621610	22186.44	14949.15	3391667	17888.89	13555.56
3391667	19222.22	16888.89	1606579	12894.74	11105.26	3502869	20081.97	16180.33	3684052	22931.04	15448.28	3287308	17338.46	13353.85
3446371	19870.97	17838.71	1643654	13353.85	11415.38	3446371	20096.77	16145.16	3684052	23413.79	15931.03	3287308	17553.85	13676.92
3446371	20322.58	18064.52	3446371	20096.77	13887.1	3561250	21000	16683.33	3561250	22866.67	15516.67	3391667	18111.11	14111.11
3561250	21700	19133.33	3621610	21237.29	14830.51	3561250	21000	17033.33	3684052	23655.17	16172.41	3446371	18741.94	14564.52
3502869	21688.53	18819.67	1669336	13671.88	11757.81	3502869	20770.49	16983.61	3621610	23372.88	16372.88	3338672	18375	14437.5
3446371	21225.81	18854.84	3561250	21700	15166.67	3446371	20548.39	17048.39	3621610	23372.88	16372.88	3338672	18375	14437.5
3391667	21000	18777.78	3621610	22423.73	15542.37	3446371	20774.19	17048.39	3561250	23333.33	16333.33	3338672	18812.5	14765.63
3446371	21903.23	19080.64	3621610	22661.02	15779.66	3446371	21225.81	17612.9	3621610	23610.17	16966.1	3391667	19444.45	15555.56
3446371	21903.23	19080.64	3561250	22633.33	15866.67	3391667	21000	17333.33	3621610	24084.75	17440.68	3391667	19666.67	15777.78
3338672	21437.5	18593.75	3502869	22377.05	15836.07	3502869	21918.03	18016.39	3684052	24620.69	17982.76	3338672	19468.75	15640.63
3391667	21888.89	19222.22	3561250	22633.33	16216.67	3561250	22400	18550	3446371	23483.87	17048.39	3338672	19687.5	15750
1890929	15424.78	12637.17	3502869	23065.57	16065.57	3446371	21677.42	18064.52	3561250	24266.67	17733.33	3338672	19906.25	15968.75
3446371	22806.45	20209.68	3502869	22836.07	16295.08	3338672	21437.5	17828.13	3446371	23935.48	17500	3338672	20125	16187.5
3287308	21861.54	19384.62	3502869	22836.07	16639.34	3391667	21888.89	18666.67	3561250	24966.67	18433.33	3338672	20125	16187.5
3391667	23000	20222.22	3502869	23295.08	16983.61	3446371	23032.26	19080.64	3561250	25200	18666.67	3338672	20125	16515.63
3287308	22507.69	19923.08	3502869	23295.08	17098.36	3338672	22093.75	18703.13	3561250	25433.33	19133.33	3287308	20138.46	16261.54
3391667	23222.22	20666.67	3391667	23000	17000	3391667	22777.78	19333.33	3446371	24612.9	18629.03	3338672	20562.5	16953.13
30525000	157000	231000	3502869	23754.1	17901.64	3391667	22777.78	19555.55	3446371	24838.71	18854.84	3237500	20151.52	16545.46
3338672	23406.25	20890.63	3391667	23444.45	17444.45	3391667	23000	19888.89	3561250	25666.67	19833.33	3338672	21000	17062.5
3338672	23843.75	20890.63	3391667	23444.45	17777.78	3287308	22507.69	19492.31	3446371	25064.52	19532.26	3237500	20787.88	16969.7
3287308	23800	21107.69	3446371	24161.29	18290.32	3391667	23222.22	20444.45	3502869	25590.16	20196.72	3338672	21218.75	17609.38
3502869	25360.66	22262.29	3502869	24672.13	19049.18	3391667	23444.45	20444.45	3446371	25516.13	20096.77	3287308	21430.77	17553.85
3287308	24230.77	21215.38	3391667	24111.11	18555.55	3391667	23666.67	21222.22	3446371	25516.13	20322.58	3287308	21646.15	18092.31
3502869	25819.67	22836.07	3391667	24333.33	19000	3391667	23666.67	21111.11	3446371	25516.13	20661.29	3287308	21861.54	18092.31
3338672	24937.5	22093.75	3338672	24062.5	18921.88	3391667	24111.11	21444.45	30525000	173000	188000	3237500	21848.48	18242.42
3446371	25741.94	23258.06	3391667	24555.55	19444.45	3391667	24111.11	21555.55	3446371	25967.74	21000	3338672	22750	19140.63
3338672	25375	22531.25	3287308	24015.38	19276.92	3338672	24062.5	21765.63	3446371	25967.74	21338.71	3237500	22272.73	18772.73
3338672	26687.5	24609.38	3446371	25516.13	20209.68	3338672	24281.25	21656.25	3446371	26419.36	21677.42	3189179	22044.78	18701.49
5935417	44333.33	58916.67	3287308	24446.15	19707.69	3338672	24500	21875	3502869	26737.71	22147.54	3237500	22484.85	19196.97
3338672	28000	25921.88	3338672	25375	20562.5	3338672	24937.5	22421.88	3502869	26737.71	22606.56	3237500	22696.97	19515.15
3287308	27030.77	24338.46	3391667	26111.11	21222.22	3237500	24181.82	21954.54	3338672	25812.5	21546.88	3237500	22909.09	19621.21
3338672	26906.25	24281.25	3287308	25092.31	21107.69	3189179	23925.37	21835.82	1.07E+08	630000	731500	3142280	22235.29	19352.94
3189179	25388.06	23089.55	3338672	25812.5	21437.5	3338672	25156.25	22750	3338672	26687.5	22421.88	3287308	23584.62	20246.15
3287308	25307.69	22938.46	3338672	25812.5	21875	3391667	25888.89	23555.55	3338672	26687.5	22531.25	3189179	22671.64	20059.7
3391667	25888.89	23222.22	3287308	25523.08	22076.92	3189179	24343.28	22253.73	3287308	26384.62	22184.62	3287308	23800	20784.62
3338672	25375	22640.63	3287308	25953.85	22076.92	3287308	25307.69	23153.85	2.14E+08	1295000	1414000	3189179	23089.55	20582.09
3287308	24661.54	22076.92	3338672	26468.75	22968.75	3189179	24970.15	23089.55	3287308	26600	22507.69	3237500	23757.58	21212.12
3446371	25516.13	22693.55	3287308	25953.85	22938.46	3189179	25805.97	23611.94	3287308	26600	22723.08	3189179	23507.46	21000
3338672	24500	21656.25	3287308	26169.23	23046.15	3287308	27030.77	24984.62	3189179	26014.93	22358.21	3237500	23757.58	21742.42
3338672	24281.25	21437.5	3391667	27444.45	24222.22	3287308	27461.54	25630.77	3237500	26515.15	23121.21	3189179	23507.46	21731.34
3338672	24062.5	21328.13	3338672	27125	24062.5	2322554	21456.52	20923.91	3338672	27562.5	24062.5	3052500	22600	21100
3391667	24333.33	21333.33	3237500	26515.15	23651.52	3237500	27787.88	25878.79	3287308	27461.54	24015.38	3189179	23716.42	21940.3
3338672	23625	20890.63	3237500	26515.15	23757.58	3142280	26147.06	23985.29	3237500	27787.88	24712.12	3142280	23676.47	21926.47
3446371	24161.29	21338.71	3237500	26727.27	24075.76	3189179	26014.93	23402.98	3287308	28538.46	25307.69	3189179	24134.33	22567.16
3446371	23935.48	21000	3287308	27246.15	24661.54	3237500	26090.91	23439.39	3189179	27686.57	24552.24	3189179	24343.28	22671.64
3338672	22750	20125	3287308	27461.54	24769.23	3237500	25878.79	23015.15	3237500	27575.76	24500	3096739	23637.68	22318.84
3391667	23000	20222.22	3287308	27246.15	24984.62	1.07E+08	630000	745500	3237500	27363.64	24393.94	23741666	134555.6	164888.9
3338672	22531.25	19468.75	3237500	26939.39	24712.12	3391667	26111.11	23111.11	3237500	27151.52	24075.76	3237500	25242.42	23651.52
3502869	23524.59	20311.47	3096739	25869.56	23434.78	3237500	24393.94	21848.48	3237500	26939.39	23969.7	3096739	24043.48	22927.54
3502869	23295.08	20196.72	3237500	26727.27	24181.82	17806250	101500	117250	3287308	26384.62	24015.38	4360715	31857.14	43428.57
3391667	22777.78	19222.22	3189179	26014.93	23402.98	3287308	24230.77	21753.85	3237500	25242.42	23015.15	4360715	31571.43	44000
3446371	23032.26	19306.45	3237500	26303.03	23333.33	3237500	23545.46	21212.12	3237500	25030.3	22909.09	4451563	32375	44479.17
3391667	22555.55	18888.89	3338672	26687.5	23734.38	3391667	24777.78	22000	3237500	24606.06	22909.09	3237500	26090.91	24712.12
3446371	22580.64	18967.74	3287308	26169.23	22938.46	3237500	23545.46	20787.88	3391667	25222.22	23777.78	4645109	34391.3	47021.74
3446371	21451.61	18064.52	3391667	26555.55	23444.45	3287308	23800	21107.69	3338672	24500	23187.5	4273500	31920	43260
3446371	20548.39	18177.42	3287308	25523.08	22184.62	3287308	23369.23	20892.31	3338672	24281.25	22968.75	4546277	33808.51	46170.21
3561250	21233.33	18433.33	3338672	25593.75	22203.13	1695833	15444.44	16666.67	3338672	24062.5	22750	4546277	33808.51	46468.09
3502869	20540.98	17901.64	3391667	25888.89	22222.22	3391667	23888.89	21000	3338672	23843.75	22640.63	4748334	35622.22	48377.78
3391667	19444.45	17555.55	3338672	25156.25	21656.25	3338672	23187.5	20781.25	3338672	23625	22750	3096739	25463.77	24652.17
1765909	13710.74	11107.44	3338672	24718.75	21437.5	3391667	23666.67	21000	3287308	23153.85	22076.92	3189179	26432.84	25492.54
1737195	13373.98	10869.92	3338672	24500	21218.75	3287308	22723.08	20246.15	3391667	23666.67	22777.78	3142280	26147.06	25529.41
1737195	13373.98	10756.1	3391667	24555.55	21444.45	3446371	23709.68	21000	3338672	22750	22093.75	4856250	36590.91	50590.91
3561250	19833.33	17150	3287308	23584.62	20246.15	3446371	23483.87	20887.1	3338672	22531.25	21984.38	3237500	26727.27	26727.27
3502869	19393.44	16868.85	3391667	24111.11	20777.78	3391667	23000	20333.33	3338672	22531.25	21765.63	3096739	25463.77	25971.01
3621610	19813.56	17203.39	3446371	24612.9	20887.1	3391667	23000	20111.11	3338672	22093.75	21437.5	3142280	26147.06	26250
3561250	18666.67	16450	3446371	24161.29	20661.29	3338672	21875	19250	3391667	22333.33	21777.78	3142280	26352.94	26970.59
3561250	18900	16100	3446371	23935.48	20435.48	3287308	21430.77	19169.23	3338672	21875	21109.38	3096739	26072.46	26579.71
3621610	16254.24	15661.02	3391667	23666.67	20000	3287308	21215.38	18846.15	3391667	21888.89	21444.45	3096739	26681.16	27289.86
			3502869	24213.12	20426.23	3338672	21437.5	18703.13	3338672	21218.75	20890.63	3009507	26323.94	26816.9
			3446371	23483.87	19645.16	3446371	22354.84	18854.84	3446371	21903.23	21338.71	3096739	27086.96	27797.1
			3446371	23258.06	19419.36	3446371	22354.84	18629.03	3338672	21000	20453.13	3096739	27086.96	28101.45
			3502869	23754.1	19508.2	3338672	21437.5	17828.13	3446371	21677.42	20774.19	3052500	27000	27600
			3391667	22777.78	18666.67	3287308	20784.62	17230.77	3391667	21222.22	20222.22	3052500	26800	27300
			3446371	23258.06	18629.03	3502869	21229.51	18245.9	3338672	20562.5	19796.88	3009507	25929.58	26915.49
			3502869	23295.08	18819.67	3446371	20096.77	17161.29	3391667	20777.78	19888.89	3052500	26000	26500
			3446371	22580.64	18177.42	3391667	19666.67	16666.67	3391667	20777.78	19777.78	3009507	25140.85	25338.03
			3446371	22354.84	18177.42	3502869	20081.97	16983.61	3391667	20555.55	19555.55	3052500	25200	25200
			3446371	22129.03	17725.81	3561250	20300	17266.67	3287308	19492.31	18738.46	3009507	24746.48	24647.89
			3502869	22377.05	17901.64	3502869	19852.46	16868.85	3391667	19888.89	19333.33	3142280	25735.29	25426.47

Human

Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
			3446371	21451.61	17274.19	3561250	19366.67	16566.67	3391667	19888.89	18888.89	3096739	25057.97	24956.52
			3561250	22166.67	17616.67	3561250	18900	15983.33	3502869	20081.97	19393.44	26709376	159250	192500
			3391667	20777.78	16666.67	3446371	18290.32	15467.74	3391667	19000	18555.55	3096739	24449.28	24550.72
			3561250	21700	17266.67	3561250	18433.33	15866.67	3391667	18777.78	18444.45	3009507	23563.38	23563.38
			3561250	21466.67	16916.67	3684052	18827.59	15931.03	3391667	17888.89	17888.89	3052500	23800	23400
			3502869	21000	16524.59				3391667	17888.89	17777.78	3009507	23169.01	23169.01
			3502869	21000	16409.84				3561250	18666.67	18550	3009507	23169.01	22873.24
			3621610	21474.58	16728.81				3561250	18666.67	18433.33	3052500	23400	22900
			3561250	21000	16333.33				3748684	18789.47	18666.67	3142280	24088.23	23470.59
			3446371	20096.77	15693.55				3684052	18103.45	17862.07	3009507	22577.46	22380.28
			3561250	20300	15866.67				3621610	17677.97	17559.32	3096739	23434.78	22623.19
			3561250	20300	15400				3561250	17500	17266.67	3142280	23470.59	22750
			3684052	20758.62	15931.03				3684052	18103.45	17620.69	3237500	24181.82	23333.33
			3502869	19393.44	15147.54				3621610	17203.39	17440.68	3142280	23470.59	22441.18
			3502869	18934.43	14918.03				3621610	16966.1	17084.75	3142280	23264.71	22338.23
			3561250	19133.33	15050				3684052	17137.93	17379.31	3142280	23058.82	22029.41
			3684052	19793.1	15448.28				3561250	16333.33	16566.67	3189179	23298.51	22149.25
			3561250	18900	14233.33							3142280	23058.82	21617.65
			3748684	19280.7	14736.84							3052500	22200	20800
			3502869	18016.39	13540.98							3096739	22014.49	21000
			3561250	17733.33	13533.33							3142280	22235.29	21000
												3096739	22014.49	20797.1
												3052500	21400	20000
												3142280	21823.53	20485.29
												1858044	15765.22	15278.26
												1874342	15964.91	15350.88
												3052500	21000	19300
												3189179	21417.91	20059.7
												3142280	21000	19455.88
												3142280	20794.12	19352.94
												3287308	21430.77	19923.08
												3237500	20787.88	19409.09
												3287308	21000	19600
												3237500	20363.64	19196.97
												3237500	20151.52	19090.91
												3391667	20555.55	19222.22
												3338672	18812.5	18046.88
												3189179	17238.81	15880.6
												3287308	17769.23	16153.85
												3338672	17718.75	15750
												3391667	18111.11	16000
												3391667	17888.89	15777.78
												3338672	17500	14765.63
												6284559	26764.71	42000
												3391667	17222.22	14555.56
												3338672	16625	14328.13
												3287308	16476.92	14215.38

ISAC

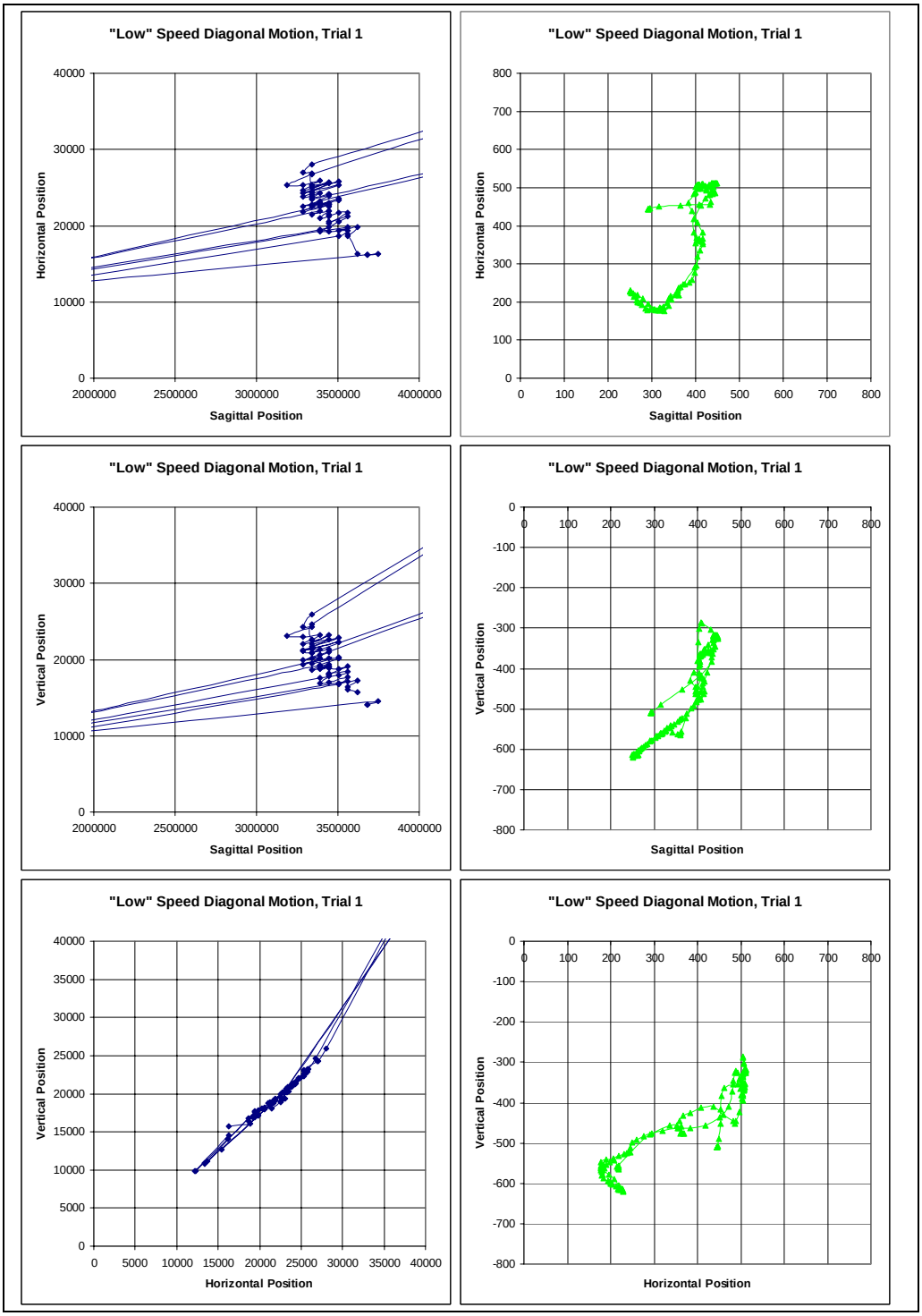
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
291.9048	444.5685	-510.6915	294.0719	441.2303	-509.8942	293.774	441.9781	-510.5088	291.0928	444.2876	-511.8438	293.0071	442.1598	-509.7471
292.0063	444.5108	-510.6002	294.1348	441.2293	-509.8371	293.8994	441.9764	-510.3602	291.0292	444.288	-511.8666	292.8292	442.3333	-509.8383
292.0453	444.4535	-510.6002	298.7093	443.7168	-505.3578	293.3932	443.0127	-510.1885	294.1584	444.0393	-509.8939	293.3274	442.6128	-509.2442
292.1713	444.4528	-510.4858	328.3225	448.3922	-479.5007	300.6178	447.1829	-505.4739	320.2305	441.9887	-490.978	302.0669	447.7245	-504.0929
292.0693	444.5105	-510.543	375.8322	447.6265	-433.7477	352.2762	456.8953	-471.1644	375.538	437.4181	-440.4292	350.1222	456.8993	-476.6297
295.6237	446.5231	-507.7876	421.13	447.5736	-389.5835	408.8978	454.8858	-424.0512	427.6554	418.8166	-391.861	415.4773	453.9559	-429.4237
315.3992	449.2955	-488.8261	438.0987	449.3631	-369.18	442.909	448.7888	-374.7274	461.7481	392.3589	-363.7609	447.0123	447.4943	-379.0728
364.7843	452.8448	-452.1063	440.4673	464.0021	-364.9122	447.0087	457.9303	-337.2621	475.2733	375.0458	-351.8913	452.8862	457.9061	-341.3975
408.6259	454.5553	-417.3012	440.1141	482.0927	-371.5658	438.0698	474.7767	-326.4348	485.4623	367.8036	-349.0362	443.2324	478.8999	-321.0688
432.5273	454.8815	-383.9635	437.9405	500.2801	-376.2822	431.8602	493.3053	-331.6366	491.5599	364.0087	-352.155	437.2666	491.518	-321.3971
435.3255	461.9632	-364.0424	437.4044	505.9412	-373.8218	429.9528	506.5031	-339.2766	491.1835	360.4674	-357.2674	431.3189	503.943	-327.4358
430.3352	482.1001	-353.8169	442.3658	504.729	-363.4221	433.0228	508.6736	-347.1762	495.3456	345.4518	-358.3509	433.8583	507.7742	-335.5844
424.7375	492.8985	-353.8351	447.5939	504.3646	-347.5926	443.8493	509.3418	-353.106	500.7916	318.4342	-355.7583	441.996	508.8973	-345.4675
423.4452	503.9046	-360.3277	450.0481	501.174	-333.777	446.6907	507.1764	-353.3159	505.8988	276.9937	-353.4188	448.9551	507.7628	-355.0132
430.5827	507.8725	-361.6746	448.1917	499.3552	-322.1729	451.0198	503.849	-346.8983	508.5505	242.6072	-347.6651	451.7279	504.1276	-356.6169
433.216	506.3957	-357.5568	450.1987	494.5165	-316.6651	451.6208	505.5419	-337.8565	508.7558	222.0725	-333.2546	452.451	506.6989	-349.6355
437.7392	502.5605	-352.3347	452.0107	488.531	-318.0868	447.5877	506.2539	-328.659	504.0016	216.1485	-321.0673	450.5022	508.5421	-341.3693
440.4803	502.0694	-345.0058	459.553	472.5393	-320.3434	445.1226	506.214	-321.1511	501.4042	220.7945	-316.5265	442.0765	507.6876	-325.365
436.9455	504.8618	-337.0047	471.5773	444.3334	-318.664	445.7597	506.4863	-316.0724	499.962	221.8108	-318.0358	439.3357	507.3165	-305.3873
436.6876	505.198	-331.1507	488.1742	387.8198	-313.0095	444.7745	505.1338	-313.4988	499.1936	213.0081	-321.5704	437.398	505.5695	-291.984
436.6843	505.3826	-328.5905	502.847	312.3675	-309.7961	445.3068	502.8545	-311.6438	495.1078	204.8153	-331.5876	436.5511	505.0876	-289.2309
437.0507	505.5236	-327.3995	505.5167	257.5827	-313.7852	444.8152	498.2184	-305.2004	492.127	199.8462	-344.5431	435.8583	506.0129	-291.9035
437.291	505.648	-326.0805	502.6827	213.7576	-321.4331	444.157	492.0346	-297.7568	489.1373	195.9434	-353.9412	436.276	506.8095	-294.0322
437.8723	504.9831	-325.7147	500.5512	203.5078	-326.6335	441.1688	491.274	-290.0697	486.8737	189.206	-357.0612	436.4686	506.9282	-293.4469
438.0092	505.1864	-325.0417	496.5647	209.2121	-330.9237	433.6562	490.3401	-291.4782	481.1028	176.3195	-363.1222	436.6917	506.9584	-292.953
437.5255	507.6291	-321.9433	491.2807	224.0352	-334.7533	432.5736	489.0656	-303.6255	471.1373	154.4858	-377.151	432.8123	506.7091	-303.4173
436.9043	510.0563	-319.0536	483.807	224.3745	-342.7188	435.1244	484.1038	-315.6234	462.7344	135.0882	-390.7977	430.844	507.4499	-319.5491
436.8097	510.7888	-317.6992	482.3464	218.7006	-353.6557	435.7029	479.4182	-322.8984	457.8732	125.4186	-394.7546	430.3648	506.1017	-332.8652
437.17	510.8406	-318.0418	479.678	219.3255	-361.7371	437.6214	469.5199	-333.1347	456.2975	126.5104	-395.7665	430.2804	505.5417	-335.6983
438.4424	511.0233	-318.1753	473.4864	221.4415	-376.5526	441.4423	459.1013	-346.797	450.8498	129.2141	-397.4628	434.2798	496.0125	-334.8521
440.6145	511.3353	-317.3483	469.2935	215.7781	-391.3127	442.7593	441.1401	-366.2971	440.2493	133.0615	-405.4042	435.4592	491.515	-336.6751
442.1768	511.5596	-316.6652	462.4835	201.6924	-403.2417	447.3575	425.6286	-382.1925	434.5055	132.8648	-418.0266	428.9142	486.7557	-348.4193
444.3756	511.8755	-317.7217	454.8481	178.4921	-412.7536	448.7374	415.4795	-387.9377	424.4406	125.5225	-434.1047	427.6528	479.8617	-361.2264
446.6199	512.0073	-322.0783	444.6262	154.0922	-421.9608	448.054	406.894	-392.6939	420.9337	120.0386	-443.0163	429.1436	470.5442	-370.1196
447.9148	511.7154	-324.8838	435.0975	136.3185	-431.0869	446.2378	399.6901	-397.9243	414.4385	116.3539	-446.5474	429.4291	458.9183	-378.2306
448.018	511.7301	-325.1366	428.4604	135.3478	-439.9541	440.9897	380.2522	-404.5304	400.2018	106.5198	-455.6625	431.461	443.9676	-383.2191
444.9573	511.3896	-321.0702	422.4492	143.5632	-450.4771	440.2302	361.6385	-413.3694	394.3656	98.13124	-463.8003	435.755	417.6265	-389.8177
430.5993	507.8748	-304.639	416.0985	147.9068	-459.4301	443.4966	346.4596	-418.0373	385.8382	97.47069	-472.2771	443.6179	387.0872	-396.8915
411.0388	504.7126	-288.9762	411.0253	151.372	-466.4844	445.4873	339.4408	-416.1705	381.1677	101.6218	-480.7278	446.6333	360.2702	-405.3996
408.8557	505.8976	-284.9873	402.654	150.344	-474.2338	442.9826	340.899	-417.5158	374.553	105.4977	-490.4537	444.7244	337.3178	-416.6444
404.882	506.2032	-300.3892	400.5407	144.8442	-472.9443	430.6061	342.0149	-432.2059	368.084	107.9018	-498.8574	439.2588	323.4656	-430.4407
401.6246	505.0458	-334.4986	390.1639	140.8441	-476.3516	422.8682	343.8436	-447.4052	344.9264	124.2697	-525.687	433.514	317.1214	-442.9589
399.169	501.7922	-380.059	375.5029	136.0815	-489.9348	419.6806	339.0065	-455.9415	332.4258	143.4587	-557.8606	430.1068	317.0133	-448.5269
405.8877	496.6706	-422.1804	367.164	131.3772	-504.4943	422.3235	328.2142	-455.4781	329.141	145.6292	-575.3139	424.394	314.1333	-455.2929
399.822	489.1278	-445.8118	362.3936	130.227	-508.9068	425.4843	318.2724	-452.0999	327.7757	139.8501	-572.7705	418.8572	307.128	-464.0764
397.0989	486.5208	-451.417	350.9056	131.1665	-514.7805	422.5099	301.6273	-456.1902	327.7131	132.322	-565.4514	415.3231	299.59	-470.75
396.7099	482.9381	-446.3645	344.76	136.6646	-525.6378	415.0531	278.6283	-469.4954	324.7685	129.956	-560.5595	409.8785	286.133	-478.3051
383.9428	460.5566	-429.0013	335.3793	143.1506	-537.1279	410.5133	257.5975	-479.6909	320.7616	135.2303	-551.8008	407.4103	268.8409	-482.9659
392.0743	438.0288	-410.0262	326.5049	147.009	-546.2396	410.3372	252.0303	-480.1476	321.1955	133.6593	-546.5154	403.3988	252.3204	-486.7535
404.901	407.3677	-411.3646	319.394	153.4219	-553.7098	409.5086	253.9912	-479.2925	321.0205	131.144	-545.9333	398.9353	244.4822	-491.4372
415.4617	383.3438	-425.6172	315.5434	156.3387	-557.6884	406.3246	256.6374	-482.5341	319.0735	134.556	-548.9735	392.2907	245.0551	-496.8847
416.2691	367.0636	-432.9583	310.4112	157.2053	-559.9987	397.8817	261.4283	-491.6691	314.0912	141.173	-556.6974	381.6204	249.2537	-506.0056
413.0248	359.3926	-445.5998	302.1331	165.13	-567.7769	389.6246	265.8059	-502.0136	303.7057	152.12	-570.0998	375.0975	250.5632	-514.1499
407.9049	363.6092	-459.5272	296.3399	172.0641	-574.836	387.1845	265.9906	-505.1651	296.2266	164.4285	-587.0429	367.8772	247.3251	-520.5129
402.9022	362.1529	-472.2468	291.923	174.5011	-578.4821	381.3813	263.6319	-508.7487	292.0479	167.5058	-601.2782	365.5115	244.0373	-523.6223
404.3733	360.2662	-476.4893	290.9746	172.6101	-578.6126	375.2003	259.5833	-515.0745	283.6266	175.9421	-614.9024	361.9105	243.2836	-524.7518
404.1327	363.8915	-474.4712	286.3505	173.1482	-581.4251	367.8581	243.7204	-523.596	279.3168	182.5982	-620.8377	357.4927	245.6285	-528.7246
401.9424	368.211	-476.0176	280.1219	181.2054	-587.3325	361.4019	219.1333	-529.2652	281.2717	178.6717	-616.9554	350.4876	249.7523	-535.8196
408.0163	366.2955	-476.3521	275.2915	188.099	-592.6308	360.2849	204.4626	-529.1047	284.7139	172.231	-597.8717	338.0501	252.2022	-547.5815
415.3681	357.1865	-462.1485	272.0127	192.9063	-595.1559	356.8589	202.5691	-530.2684	288.1099	166.0353	-585.0703	332.8538	251.9481	-553.6489
416.081	351.8803	-455.4589	270.975	194.3807	-595.9589	350.5745	208.2837	-536.8654	292.2012	158.6034	-580.0388	329.4741	248.9618	-555.7817
409.2459	336.5775	-457.5186	272.0445	191.5278	-594.4583	342.0212	214.4807	-546.2164	290.4046	159.0912	-582.0521	308.7293	246.3995	-572.2473
397.1436	290.3249	-478.333	272.1037	191.2243	-593.9384	337.4568	214.0025	-550.4399	289.3351	161.0498	-583.8609	298.6948	243.0414	-588.8848
372.3122	245.5742	-523.3352	273.433	189.4949	-593.2412	337.2947	209.4459	-550.0351	287.7847	161.3675	-583.8847	294.5326	232.6941	-609.6248
362.0149	219.3358	-557.2755	275.1655	184.8207	-590.2417	337.546	207.5363	-546.2262	277.3192	176.4157	-592.583	277.3986	230.5552	-626.1296
361.7027	218.9756	-565.1884	275.0729	185.0403	-590.228	335.8798	209.2119	-545.3936	268.7664	191.8344	-604.0553	254.3098	251.9617	-655.1135
360.3567	219.3439	-561.0802	275.0196	185.4163	-590.6264	326.0566	214.1259	-553.8544	269.4964	190.5143	-603.4858	231.3382	279.7824	-678.9478
353.6142	217.0199	-562.3393	277.4545	181.9407	-588.512	319.9661	219.1691	-562.313	271.6192	185.3436	-599.1527	222.2508	290.0808	-681.9914
342.244	215.3035	-558.5758	283.4947	173.1997	-583.0566	315.7273	215.5181	-565.6359	270.6963	187.1399	-599.4581	227.0834	280.5966	-678.8081
339.6102	207.6329	-542.9211	288.0333	165.26	-577.117	314.7104	206.9072	-565.3965	268.3797	190.4244	-601.6526	226.3611	276.1521	-677.6948
338.4138	190.2911	-540.1137	290.57	163.0847	-575.4456	311.0649	197.5412	-567.132	262.6647	200.4369	-606.327	216.9074	286.5401	-690.3976
329.3507	177.1005	-546.7271	295.0738	161.8669	-572.3232	309.3735	194.4886	-568.6742	259.3505	205.7071	-609.4156	199.6717	309.7747	-706.3569
324.9207	177.7486	-555.2258	301.3288	159.4013	-567.0607	306.687	196.8737	-570.8481	258.5879	206.9502	-609.9509	186.5773	328.4406	-716.1838
319.8843	183.6765	-560.0921	307.7063	154.1617	-560.5555	303.4038	200.6055	-574.4192	257.3612	208.6999	-610.652	192.2101	320.3459	-710.3919
317.5142	184.5659	-562.0669	316.6789	154.6353	-553.9218	301.1021	203.3405	-576.1632	255.7198	211.3803	-612.0378	214.3156	288.8407	-669.2404
315.9124	180.9918	-561.7505	322.7046	157.8116	-548.24	298.24	205.4356	-578.5167	253.35	215.4887	-613.8066	238.6875	253.1135	-627.1418
315.7104	179.1396	-561.5339	328.6816	158.6175	-544.3445	294.9156	204.6891	-580.9681	252.5376	217.0487	-614.7396	235.4817	252.6629	-633.9872
315.3518	178.6372	-561.4792	331.656	159.5359	-541.7892	288.7465	205.65	-585.1685	251.4336	218.4963	-615.5681	225.4	268.261	-654.0651
312.8177	178.0656	-562.8287	339.8922	164.8521	-536.0618	285.0014	205.209	-587.303	251.1486	219.0774	-614.8101	223.9906	268.7896	-664.2771
305.5739	180.8092	-567.5706	345.5449	170.7073	-530.9484	280.4403	203.5899	-590.8933	251.1069	219.1453	-613.7424	222.5668	271.6926	-655.7217
291.5477	195.8776	-579.0259	351.8263	176.2067	-526.577	275.4901	207.2529	-594.4044	249.3288	222.3833	-614.2787	221.8708	272.3926	-654.6094
280.4848	208.5111	-589.9682	356.6974	179.6066	-522.7085	272.3603	210.5481	-596.9012	248.8043	222.9551	-614.5396	226.6119	265.5515	-639.3079
266.8036	218.7178	-604.9998	364.8248	184.5547	-516.1475	267.8094	212.93	-599.2435	248.7041	223.1746	-613.7478	230.2798	258.9943	-629.4103
263.7713	215.7922	-614.4629	374.2141	191.0238	-506.382	266.0965	214.4498	-600.2898	249.0073	222.6244	-612.5962	231.3248	257.2233	-625.7957
261.4413	217.8084	-615.158	379.7664	196.4222	-501.8513	264.7445	214.5029	-600.4654	249.5638	221.8878	-610.2515	228.7515	261.3384	-630.6852
256.6823	223.6424	-616.816	390.1606	204.7086	-494.3989	266.0182	216.1435	-599.5473	249.1407	222.5199	-610.0103	228.0186	262.6219	-631.7952
251.7296	229.2248	-620.5748	395.8844	212.043	-488.6606	266.7525	216.865	-598.9044	248.1761	224.4802	-610.844	228.5798	261.1186	-630.9576

ISAC

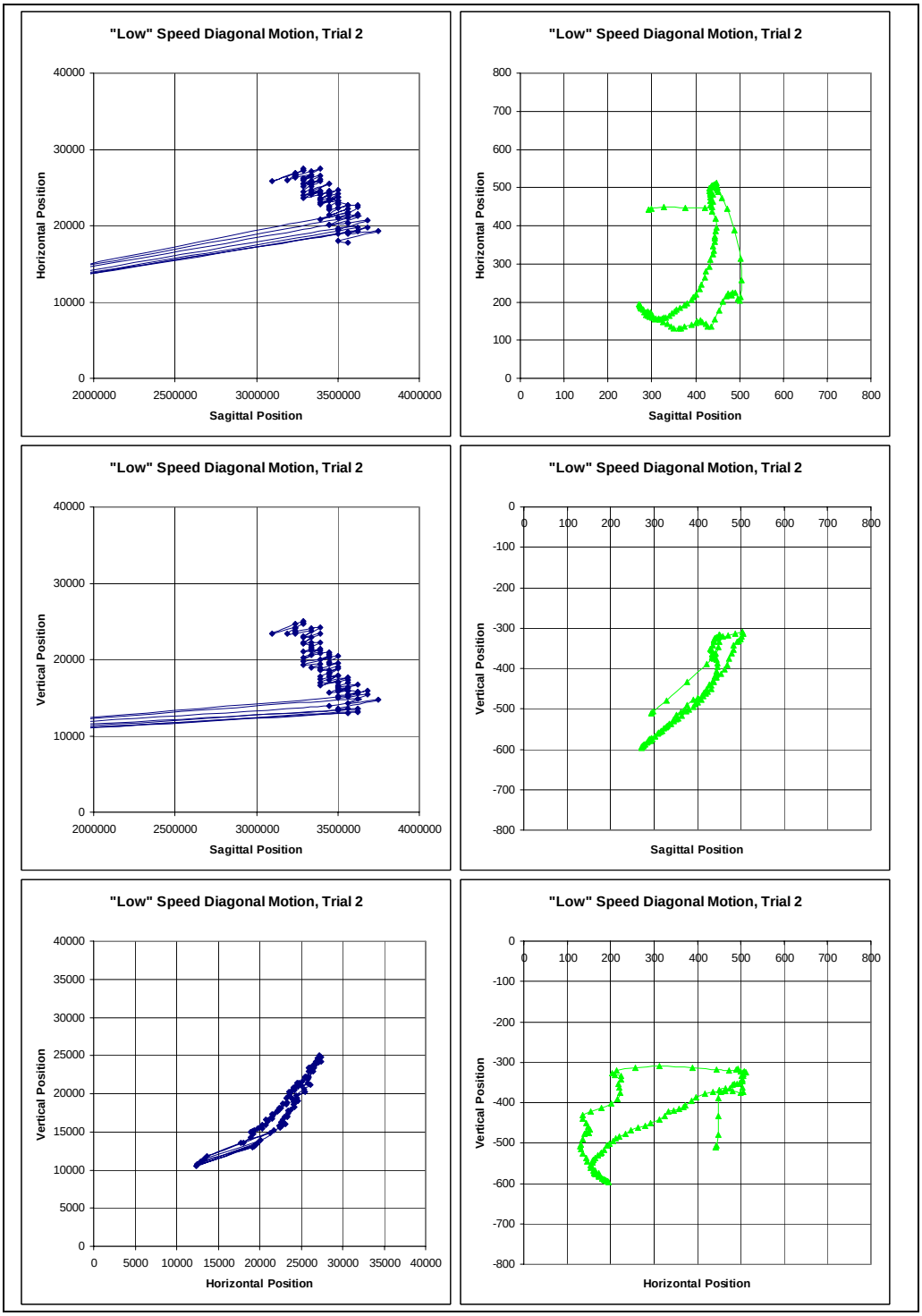
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
250.8176	229.1847	-618.2226	400.0601	221.0231	-485.0344	268.1817	215.9056	-597.8172	248.3698	224.0542	-610.7605	228.3063	261.5402	-630.934
250.9046	227.1947	-614.5044	409.1265	233.633	-476.2663	267.7223	216.7856	-597.8566	250.1234	221.5477	-609.7669	225.1428	265.9603	-632.3804
250.6354	227.1691	-613.7741	413.4587	245.9734	-468.8622	267.4899	216.9867	-598.1593	253.7548	215.4261	-607.0718	223.8516	267.9986	-633.338
250.2924	226.7648	-614.1416	420.4268	264.2335	-462.7716	260.0643	225.8179	-600.8168	255.082	212.7869	-605.0177	220.6931	272.257	-634.5385
252.9251	223.0181	-612.7307	422.8612	280.1778	-457.8778	246.7687	245.5783	-610.1641	254.9255	214.0691	-605.5884	219.6943	273.5737	-634.9533
259.1939	213.3115	-608.2973	431.116	292.3363	-451.1571	243.718	249.955	-615.0931	256.2742	211.46	-604.6293	218.5629	275.1757	-636.3012
264.6632	204.5844	-603.355	432.7889	312.1136	-441.2472	246.8283	244.6669	-610.5498	258.5871	207.8355	-602.8986	218.4386	275.2354	-636.6058
268.3123	199.8608	-600.4383	438.8594	323.9933	-432.1365	247.1927	243.4196	-609.431	261.6375	202.5022	-600.4423	218.5895	274.6578	-636.5694
269.2697	199.5254	-599.3398	441.8012	333.5583	-421.6079	246.6988	243.2797	-609.9059	262.3053	201.6146	-599.5144	218.6364	274.5837	-636.5512
273.0033	198.0581	-597.0159	439.9837	345.8207	-419.5762	247.1872	239.2877	-609.6266	262.8271	200.5697	-599.065	218.5878	274.6945	-636.7906
277.0933	192.9427	-593.6305	443.5257	358.0865	-415.4339	246.966	236.9193	-609.7123	263.9853	198.8892	-598.1552	218.9322	274.1167	-636.634
284.7146	183.3598	-586.7293	443.6582	368.2288	-410.0817	245.8973	236.7852	-610.4948	266.0675	196.8051	-596.7179	221.0092	271.2656	-635.9542
290.5851	179.3492	-580.9955	444.143	372.7847	-406.244	243.868	239.0549	-611.7184	272.851	189.9102	-591.3504	223.7442	267.4563	-634.9872
295.4553	180.7882	-577.3731	446.3066	385.6284	-395.9647	242.9663	240.3779	-612.6786	277.429	184.8567	-586.9522	226.8602	264.7076	-632.0327
302.5925	180.222	-572.3362	446.8703	396.4113	-386.3401	242.88	239.4692	-612.7677	279.3533	186.6467	-585.6205	229.7163	262.9822	-627.3978
308.6593	177.3494	-566.5905	445.0561	417.8641	-377.0977	242.9463	238.7693	-612.7444	279.7285	191.5959	-586.7945	232.676	259.0511	-625.661
317.9034	178.075	-559.735	438.4096	436.7404	-372.3922	246.4862	233.2499	-612.008	283.2191	198.0015	-585.1134	234.5106	257.6281	-624.7676
323.5127	183.0302	-554.0233	433.6723	454.3888	-373.5123	257.265	213.1694	-603.5107	284.9863	202.6266	-584.3487	235.8826	255.9315	-623.4283
326.992	188.8192	-552.9649	433.5106	465.8089	-372.2527	256.4872	215.1822	-605.1385	288.9401	206.6725	-581.9731	235.5766	257.929	-624.1827
335.2747	197.1443	-548.3048	433.7313	474.1434	-365.281	258.8519	211.8272	-604.2356	296.1273	211.552	-576.7792	235.5294	263.7376	-624.953
345.5014	207.165	-539.1417	433.6894	481.3085	-355.8438	262.3146	204.4642	-600.4567	299.905	218.6107	-574.8007	240.0891	267.7031	-630.604
354.5298	219.5384	-531.6759	432.6375	485.5836	-353.698	260.8866	207.3073	-602.6927	302.755	230.5058	-572.2426	248.1868	265.279	-629.1037
359.3032	229.9659	-527.3758	431.8717	491.8672	-353.6071	257.4941	212.3795	-623.395	303.8081	244.5112	-572.705	256.6188	265.6402	-621.9237
361.7216	236.2196	-525.221	430.6732	497.2318	-350.5009	248.6271	225.2512	-643.417	308.9682	256.6688	-567.9931	261.0039	271.2135	-619.3309
365.5668	240.4569	-521.9904	435.1262	503.8871	-342.129	244.537	237.4303	-646.0991	315.366	270.0349	-562.1089	262.4538	275.5358	-617.2933
375.1751	245.5933	-512.2075	437.2203	505.824	-332.9769	249.6449	226.2884	-640.5647	316.1382	286.3812	-560.0929	269.8428	275.422	-610.4486
385.6558	250.4402	-499.5188	439.0091	506.1546	-329.4745	254.9513	218.8325	-632.8223	317.8747	301.0757	-557.1182	281.694	277.0242	-597.5782
392.1393	259.1965	-490.8589	440.7414	506.4785	-325.5782	268.4053	214.6161	-622.9747	318.2297	317.4258	-554.1936	287.2121	279.4111	-584.2384
397.6486	276.3447	-483.7378	443.5169	507.8827	-323.1728	281.7614	199.9214	-609.5707	319.8711	327.5522	-550.057	287.8525	279.9068	-583.3903
401.0988	295.2623	-476.6233	446.3445	509.9718	-324.2716	290.4153	191.5709	-594.9608	323.9444	340.9034	-543.7668	287.362	280.1211	-586.174
403.4693	319.7669	-469.8888	446.9096	511.4772	-325.7912	305.83	188.1541	-582.2304	324.9345	348.594	-539.9702	293.5732	278.4266	-582.0206
400.3304	353.8209	-463.8776				310.8301	191.4889	-577.3106	323.9707	359.0049	-538.7647	300.3764	273.9761	-576.5777
396.5364	383.0741	-462.1654				328.9322	207.4233	-565.9913	326.1326	367.3104	-533.5388	308.206	274.1738	-570.2919
396.6598	417.5662	-457.5243				352.899	223.7563	-545.3058	327.6461	376.481	-529.293	311.0108	278.7792	-567.0127
412.9972	452.2718	-436.4231				354.5716	240.1186	-536.4947	326.6466	388.6702	-525.4263	317.1787	289.4129	-561.8827
422.173	471.5189	-409.9331				352.0022	249.2052	-542.1684	331.0961	395.8378	-522.0876	321.497	302.4402	-554.9897
432.9947	480.7205	-372.6292				351.5005	252.9636	-543.1789	336.0467	403.8068	-513.8899	324.1078	313.3438	-551.5141
438.914	482.8963	-345.1057				355.8299	255.411	-537.9021	336.4156	409.4875	-511.0533	327.2296	321.0204	-547.7227
443.8824	485.8603	-326.3195				356.7327	253.6582	-536.8282	338.2072	423.4989	-509.1038	332.2106	329.4956	-540.9732
442.0531	487.8865	-320.7062				359.7731	252.2536	-533.1865	338.0166	439.9218	-504.1772	341.2383	338.2491	-529.9145
436.7504	492.4641	-324.7717				364.7917	252.2223	-527.9459	331.5051	455.7275	-500.8743	347.1047	353.2401	-523.0952
425.3649	496.9741	-340.198				367.9702	254.7921	-523.7291	326.2628	469.0807	-497.9278	368.6325	378.0951	-498.324
412.3556	498.4997	-364.7897				369.4814	257.7929	-521.9288	325.139	472.361	-496.7202	385.5939	404.5447	-452.8981
406.2357	502.6611	-390.3624				371.6257	266.4133	-519.0711	328.2489	472.3351	-493.4817	381.1617	410.6217	-439.856
403.764	504.3981	-395.124				380.5454	282.1681	-507.6263	336.0343	473.0842	-483.886	380.4905	410.614	-463.2652
405.4104	505.4097	-384.4542				387.8623	304.2991	-494.1902	342.0339	477.8898	-472.972	376.4505	409.6536	-482.876
404.404	503.5362	-379.2316				386.34	330.05	-489.853	341.1813	482.1733	-469.145	369.139	408.3702	-493.24
403.9236	503.9883	-380.2727				381.6927	352.199	-488.8134	339.1125	488.4306	-468.159	367.9056	410.5009	-493.4815
404.8549	505.8519	-378.267				378.9196	367.9822	-486.4554	338.4239	496.07	-462.5348	367.3269	410.6962	-493.3143
407.1518	506.5292	-369.2074				380.2364	375.77	-481.4199	340.6536	499.0556	-456.3228	365.877	409.8096	-493.7299
407.9081	506.4629	-363.0948				384.6012	372.5204	-477.0503	344.8274	499.9042	-449.9005	369.0429	406.9979	-490.5148
407.0992	506.347	-364.4739				391.9291	368.667	-469.0464	346.4642	499.3945	-448.4293	367.4952	407.3728	-490.5217
406.9214	506.3215	-366.6657				398.3397	364.1815	-463.4745	345.8461	499.3031	-449.6004	365.8325	411.0499	-490.1124
407.1154	506.3493	-366.1151				404.5237	359.0414	-458.8763	345.8401	499.3022	-449.5668	366.4685	414.0898	-486.859
407.0504	506.2527	-366.16				416.6841	354.4273	-448.8356	345.7126	499.2834	-449.9248	369.2413	417.609	-480.9794
406.995	506.2448	-366.5472				421.8312	359.8623	-440.8086	346.5014	499.4754	-449.6212	364.5776	431.4694	-475.7654
407.1217	506.2629	-366.352				424.7785	359.7793	-436.4489	350.3587	500.8857	-444.4569	353.4766	450.0415	-475.6624
407.1336	506.2646	-366.384				423.3642	361.0223	-439.5376	355.8158	501.7056	-436.9032	345.0567	462.5064	-476.7795
407.1714	506.27	-366.8451				420.2263	374.0722	-442.9589	364.9844	503.6386	-423.8901	344.3201	463.3082	-477.0584
407.5347	506.322	-367.4733				414.8884	387.0697	-444.287	370.5978	502.9623	-415.2225	345.75	461.2118	-476.919
407.6752	506.3421	-367.8582				412.0924	393.6428	-444.194	370.5891	502.9611	-415.1894	349.1727	457.4993	-476.0342
407.7686	506.3554	-368.1149				408.9872	395.1222	-444.7941	369.4618	503.9966	-416.0029	345.8598	461.8451	-476.7347
409.0778	507.3325	-367.5351				408.6241	399.0001	-444.9788	370.4368	504.3046	-414.4397	343.2998	465.6283	-478.8193
414.0674	509.5669	-360.1665				406.7607	406.3411	-443.4865	370.9054	503.8119	-414.3796	359.709	468.0889	-464.1992
415.8142	508.7541	-351.5836				401.1464	422.0607	-445.2078	374.6211	504.6124	-408.5848	369.9108	471.2716	-444.2052
416.152	508.5355	-351.5405				400.8335	449.2356	-437.7242	379.8981	505.4063	-398.5582	370.0578	475.8401	-438.0921
415.4431	509.4137	-353.5386				399.9286	471.0869	-410.1878				358.2957	489.7587	-442.6535
						397.6205	476.9547	-396.2117				348.9105	502.8031	-443.9116
						397.8546	476.7277	-401.4462				344.3383	508.9367	-443.7019
						399.67	475.8824	-406.9749				346.8011	506.7498	-444.7753
						398.4046	478.3239	-406.5496				356.5744	508.305	-436.2881
						397.5785	480.8794	-405.8163				382.926	515.8216	-402.7907
						399.3293	485.3305	-402.5943				394.992	508.7877	-381.884
						399.9551	489.5599	-398.3216				399.1449	507.0107	-372.4591
						400.4716	493.9016	-393.0333				395.9674	509.1085	-376.0343
						403.874	500.1063	-387.4705				395.3423	510.2982	-377.0292
						405.3603	502.8888	-380.9378				399.3049	508.4985	-370.5653
						406.4073	503.8987	-378.1284				400.2521	507.864	-368.2828
						406.6546	504.8023	-376.8699				402.7483	510.9335	-362.5807
						406.7521	505.4255	-376.0833				407.6157	510.9842	-355.0292
						407.8735	506.0207	-376.48				408.7408	493.4026	-384.0637
						408.9362	506.3471	-375.0904				392.1567	437.5695	-478.822
						409.5021	506.6034	-372.6408				375.068	387.2505	-608.0613
						410.7338	506.7795	-372.4181						
						411.6631	506.9124	-369.9981						
						412.1078	507.1527	-368.3837						
						411.995	507.4898	-368.0892						
						411.8209	507.4647	-368.7104						
						412.0405	507.4964	-368.2179						

ISAC

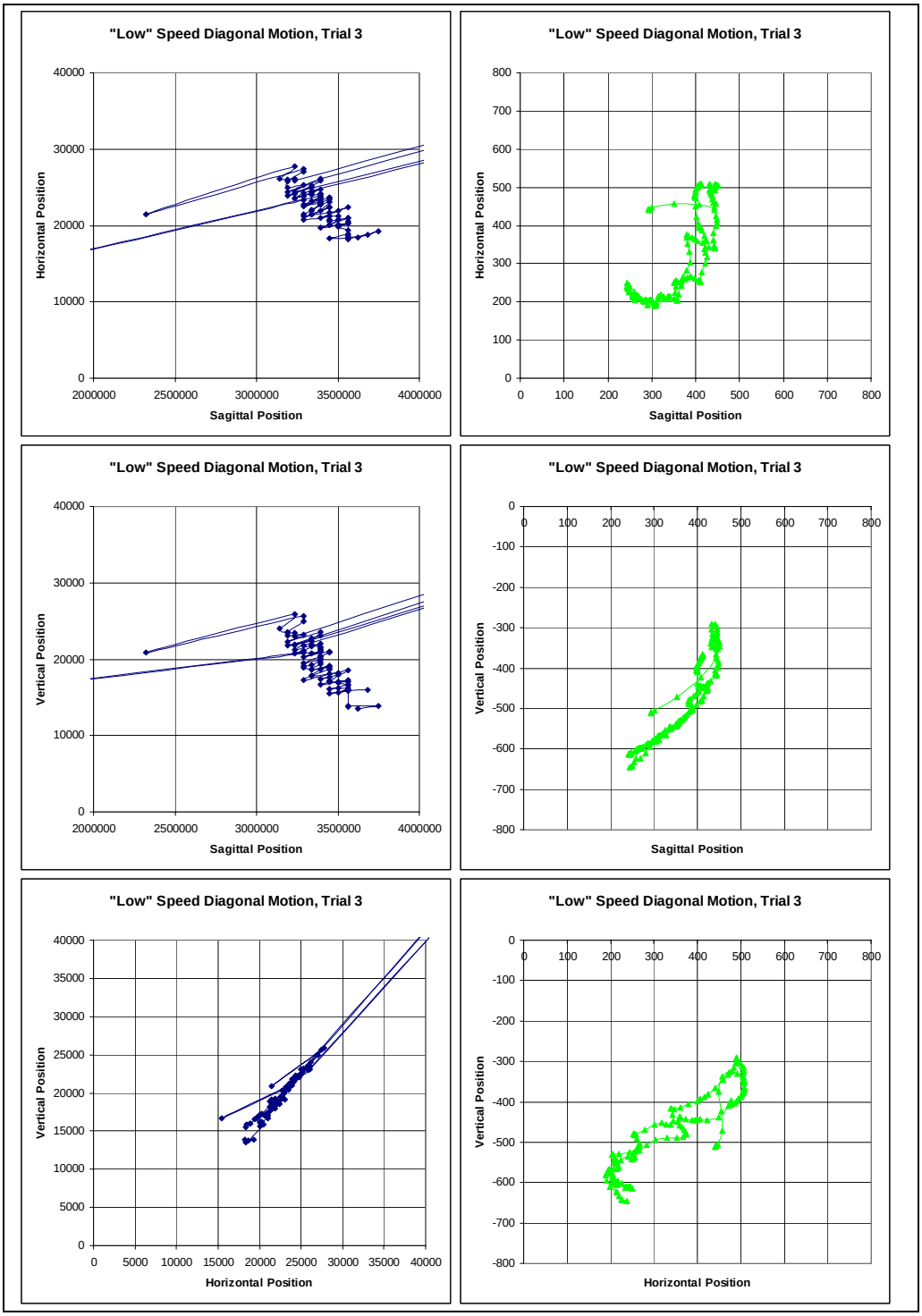
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
						412.5355	507.5678	-366.6568						
						412.1746	507.781	-366.6301						
						411.7798	508.1657	-366.9802						



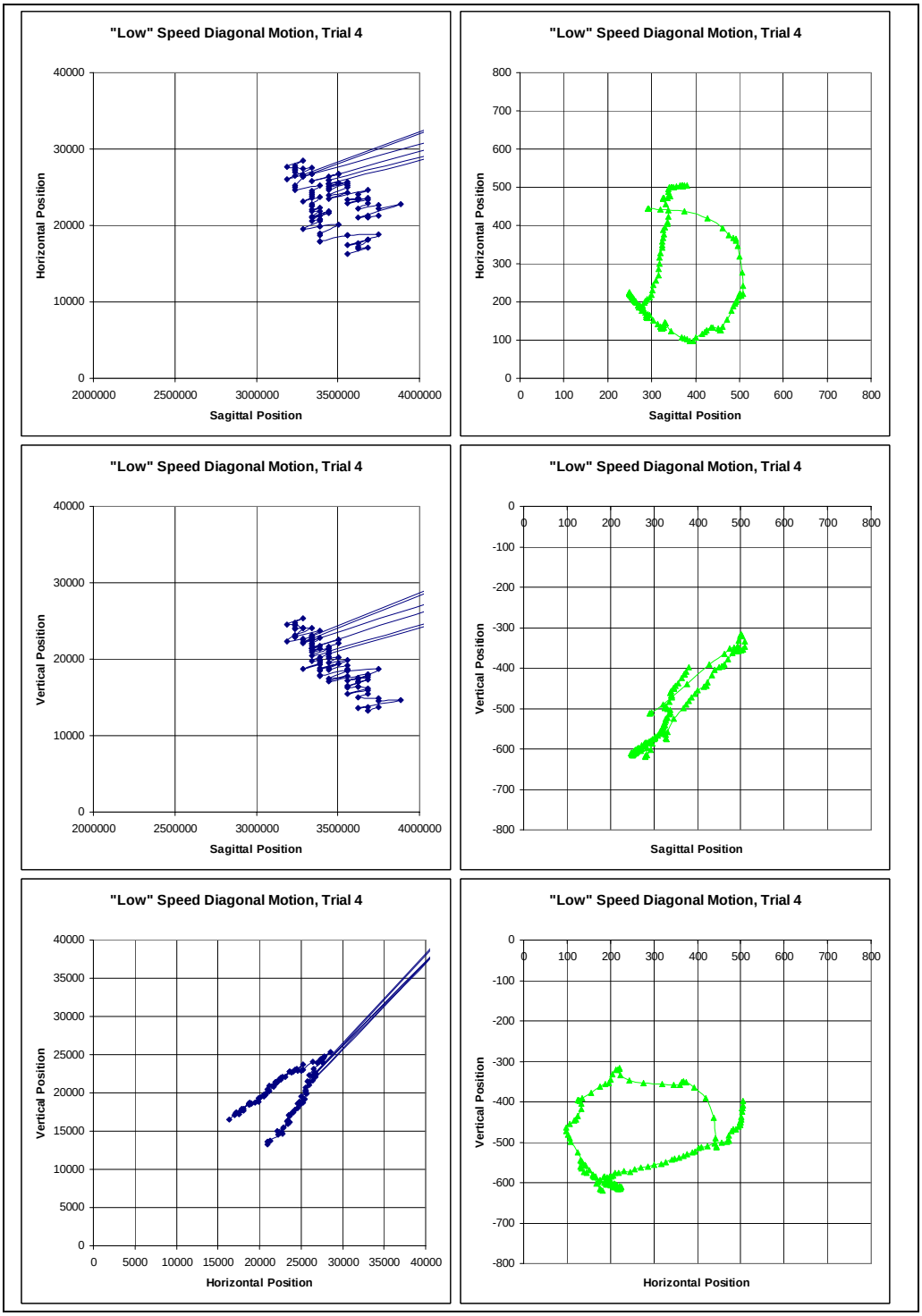
"Low" speed diagonal motion, trial 1; human motion is shown on the left, that of ISAC on the right



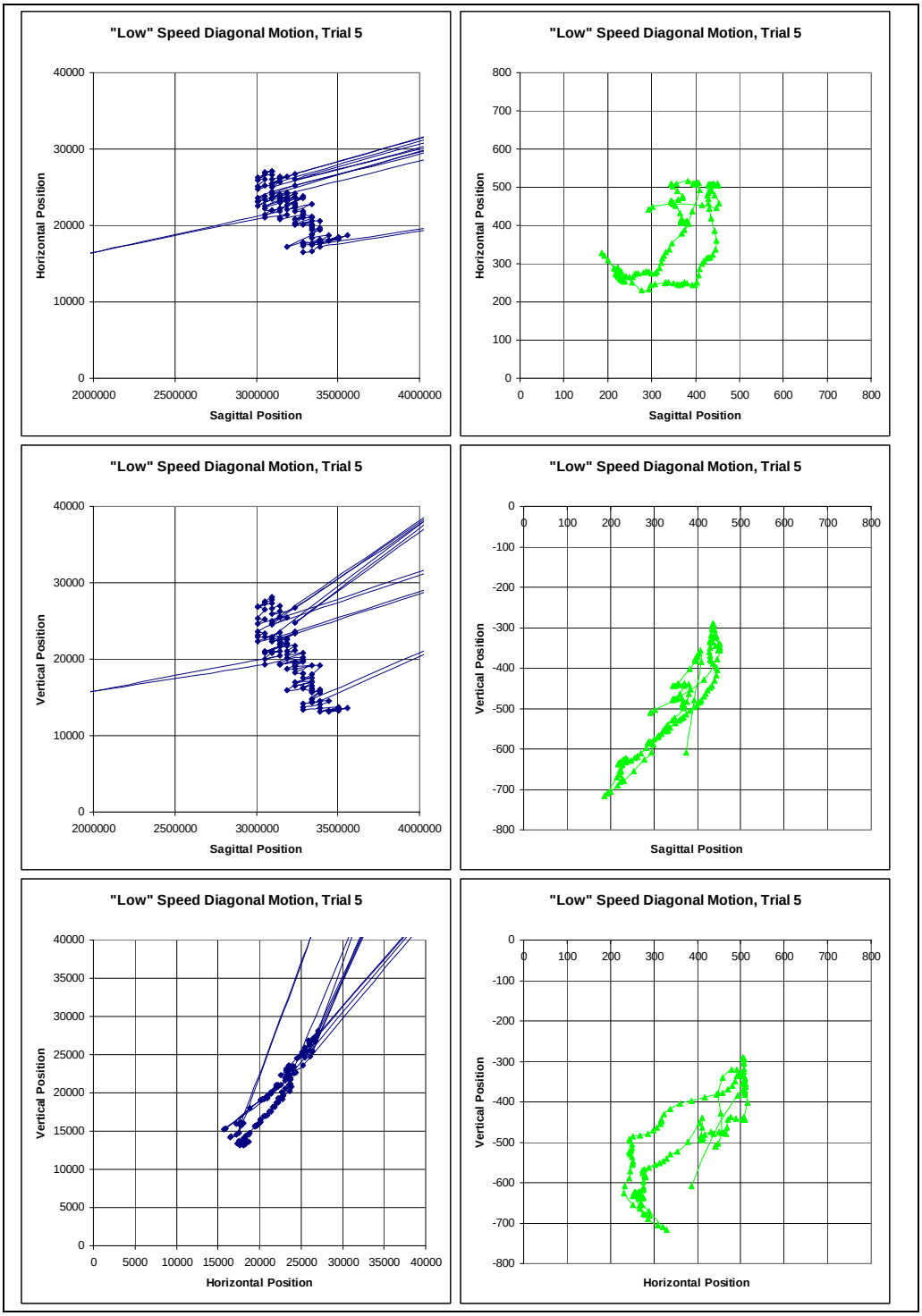
"Low" speed diagonal motion, trial 2; human motion is shown on the left, that of ISAC on the right



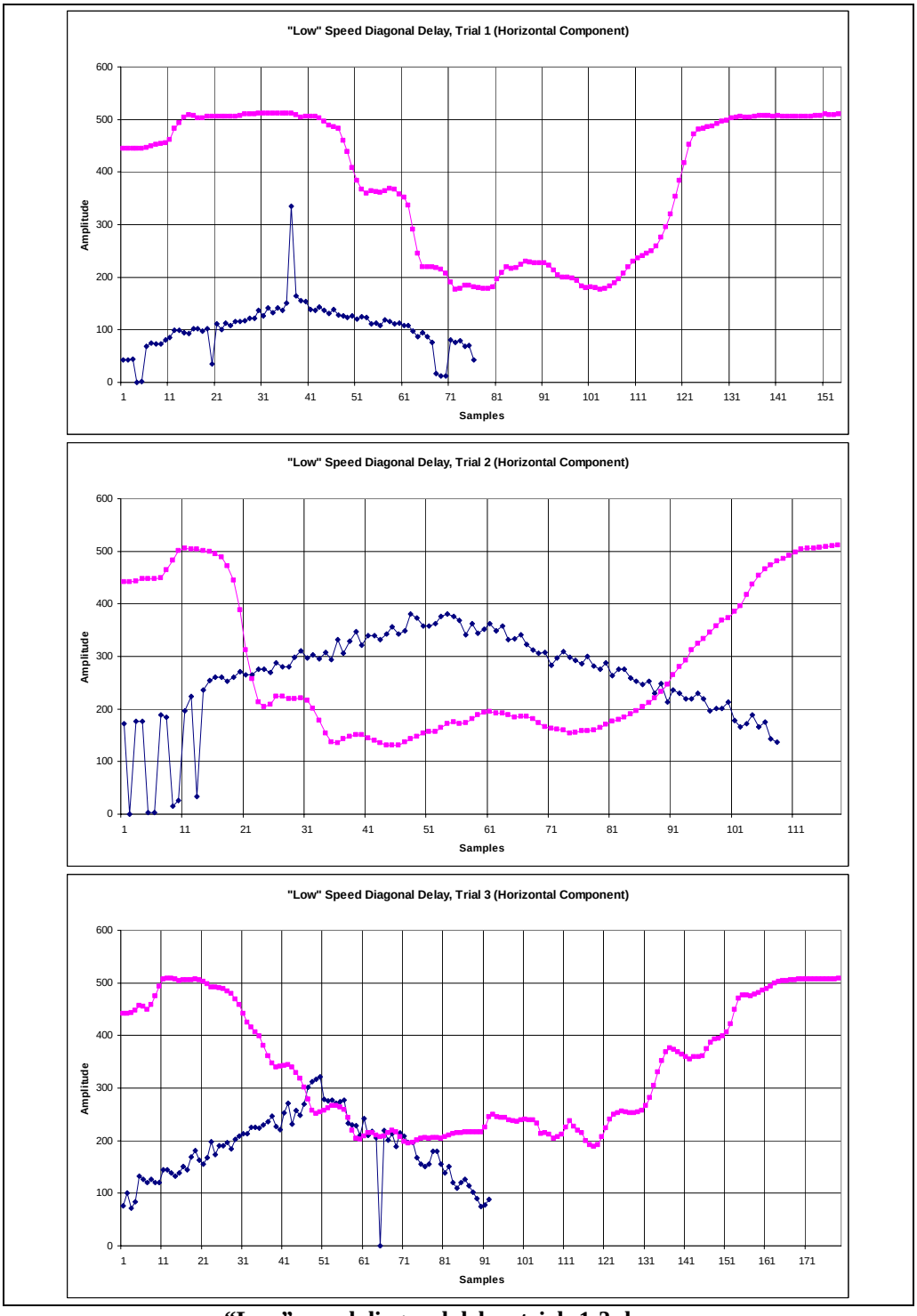
"Low" speed diagonal motion, trial 3; human motion is shown on the left, that of ISAC on the right



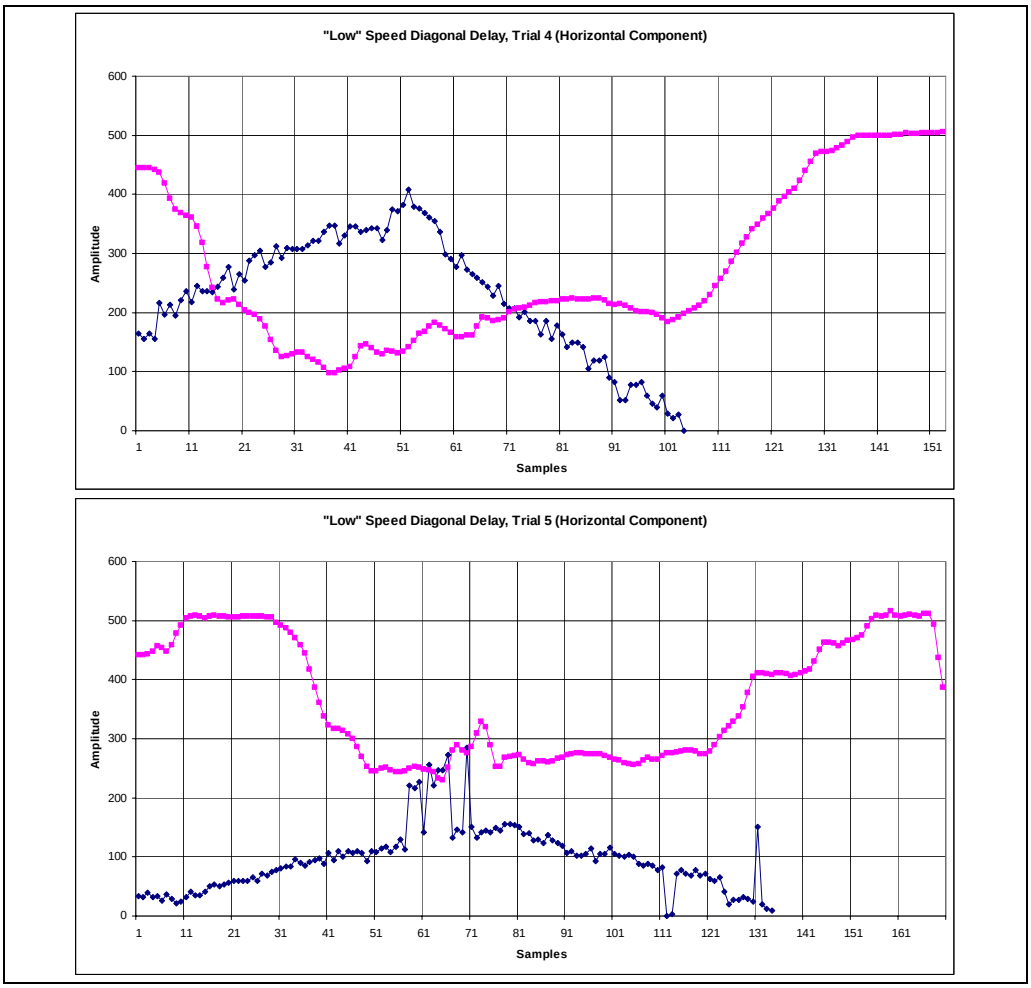
"Low" speed diagonal motion, trial 4; human motion is shown on the left, that of ISAC on the right



"Low" speed diagonal motion, trial 5; human motion is shown on the left, that of ISAC on the right



"Low" speed diagonal delay, trials 1-3; human motion is shown on the on bottom initially, that of ISAC on the on top initially



"Low" speed diagonal delay, trials 4-5; human motion is shown on the on bottom initially, that of ISAC on the on top initially

Diagonal motion at "Medium" speed

Human

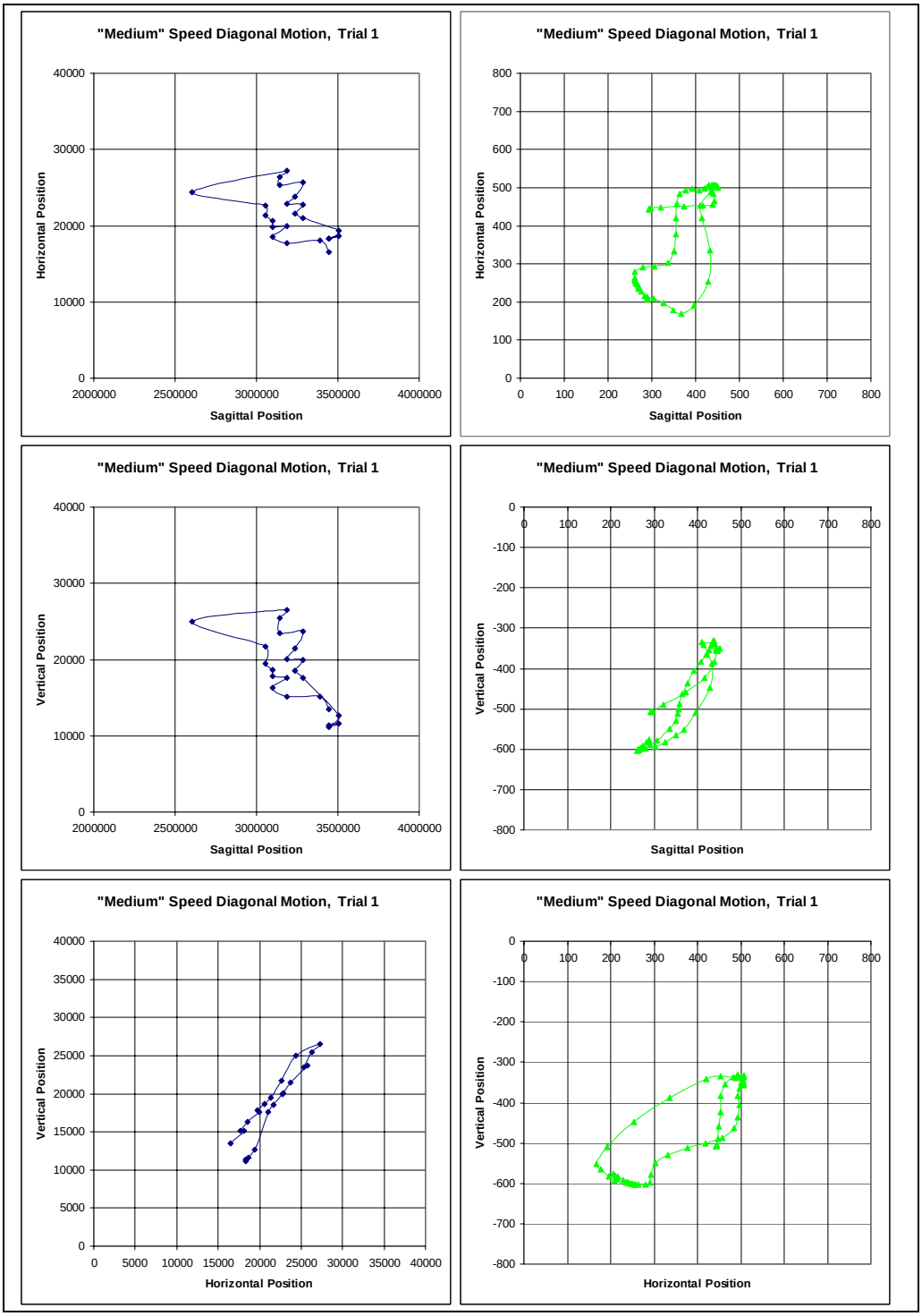
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
3446371	18290.32	11177.42	3885000	21509.09	13618.18	3502869	18475.41	11934.43	3885000	21763.64	15781.82	4273500	22120	15960
3502869	18704.92	11590.16	11246053	53421.05	76263.16	3621610	19101.7	11864.41	3885000	21254.54	15781.82	4273500	21840	15820
3446371	18290.32	11403.23	3684052	22206.9	16655.17	1765909	14404.96	12438.02	3815625	21000	15625	3815625	24500	20500
3502869	19393.44	12622.95	3748684	23456.14	17561.4	3502869	22377.05	15147.54	3748684	21982.46	15964.91	3956945	25925.93	21388.89
3287308	21000	17553.85	3748684	23947.37	17807.02	3502869	23295.08	16065.57	3684052	21965.52	16172.41	3885000	26090.91	21509.09
3237500	21636.36	18560.61	3748684	25421.05	19526.32	3502869	24213.12	16868.85	3815625	23750	17500	3748684	25175.44	21122.81
3287308	22723.08	19923.08	3561250	25666.67	20766.67	3502869	24672.13	18360.66	3815625	24250	17625	3885000	26854.54	22781.82
3189179	22880.6	20059.7	3561250	25900	21116.67	3446371	25064.52	19080.64	3815625	24750	18250	21367500	120400	136500
3237500	23757.58	21424.24	3502869	26508.2	22606.56	3446371	25967.74	20209.68	3684052	24620.69	18344.83	3815625	27750	23875
3287308	25738.46	23692.31	3502869	27426.23	24327.87	3287308	29400	29076.92	3621610	24322.03	18389.83	3621610	27881.36	25271.19
3142280	25323.53	23470.59	3338672	27781.25	26140.63	2739423	25666.67	25666.67	3748684	25912.28	19649.12	3338672	31062.5	30953.13
3142280	26352.94	25426.47	3391667	25444.45	24222.22	2704747	25253.16	25341.77	3815625	26750	20750	3621610	27406.78	25745.76
3189179	27268.66	26537.31	3446371	24612.9	21903.23	3237500	23969.7	24500	3684052	26310.35	20879.31	23741666	133000	159444.4
2605793	24414.63	24926.83	3391667	22777.78	19888.89	3287308	23584.62	23692.31	3502869	25819.67	20770.49	3561250	25200	22983.33
3052500	22600	21700	3502869	23065.57	19622.95	3287308	22938.46	23046.15	3621610	27169.49	22186.44	3502869	24213.12	21803.28
3052500	21400	19500	3621610	23372.88	19694.92	3287308	21861.54	22076.92	3502869	26967.21	22606.56	3502869	22836.07	20885.25
3096739	20594.2	18666.67	3502869	21918.03	18131.15	3391667	20777.78	20555.55	3561250	28000	23566.67	3502869	21688.53	20196.72
3096739	19782.61	17855.07	3502869	21229.51	17901.64	3621610	16966.1	16610.17	3502869	28344.26	24213.12	3748684	20508.77	19280.7
3189179	19955.22	17552.24	3621610	18389.83	15067.8	3748684	16578.95	16947.37	3621610	30016.95	25864.41	3885000	20490.91	19090.91
3096739	18565.22	16333.33	3815625	17250	13375	3748684	15842.11	16333.33	3502869	29721.31	25819.67	4109135	16961.54	15480.77
3189179	17656.72	15149.25				3885000	14127.27	14636.36	3502869	34540.98	31213.12			
3391667	18111.11	15111.11				3815625	12750	14000	3446371	35000	32403.23			
3446371	16483.87	13435.48							3391667	33000	31000			
									3338672	31281.25	29531.25			
									3338672	28437.5	27015.63			
									3391667	27666.67	25444.45			
									3621610	27406.78	34762.71			
									3446371	25967.74	23032.26			
									3391667	24777.78	21777.78			
									3338672	23843.75	20781.25			
									3502869	24442.62	21114.75			
									3561250	24266.67	20883.33			
									3502869	23524.59	19852.46			
									3621610	23610.17	20050.85			
									3502869	21918.03	18819.67			
									3621610	21711.87	18033.9			
									3815625	22750	18375			
									3748684	21000	17070.18			
									3748684	20508.77	16578.95			
									3684052	19793.1	15689.66			
									3815625	20000	15875			
									3621610	19101.7	14474.58			
									3621610	18864.41	14355.93			
									1415066	11635.76	12377.48			
									3684052	18827.59	13758.62			
									3885000	19727.27	14509.09			
									3748684	18543.86	13754.39			
									3748684	18052.63	13754.39			
									3815625	17750	13625			
									3748684	17315.79	13631.58			

ISAC

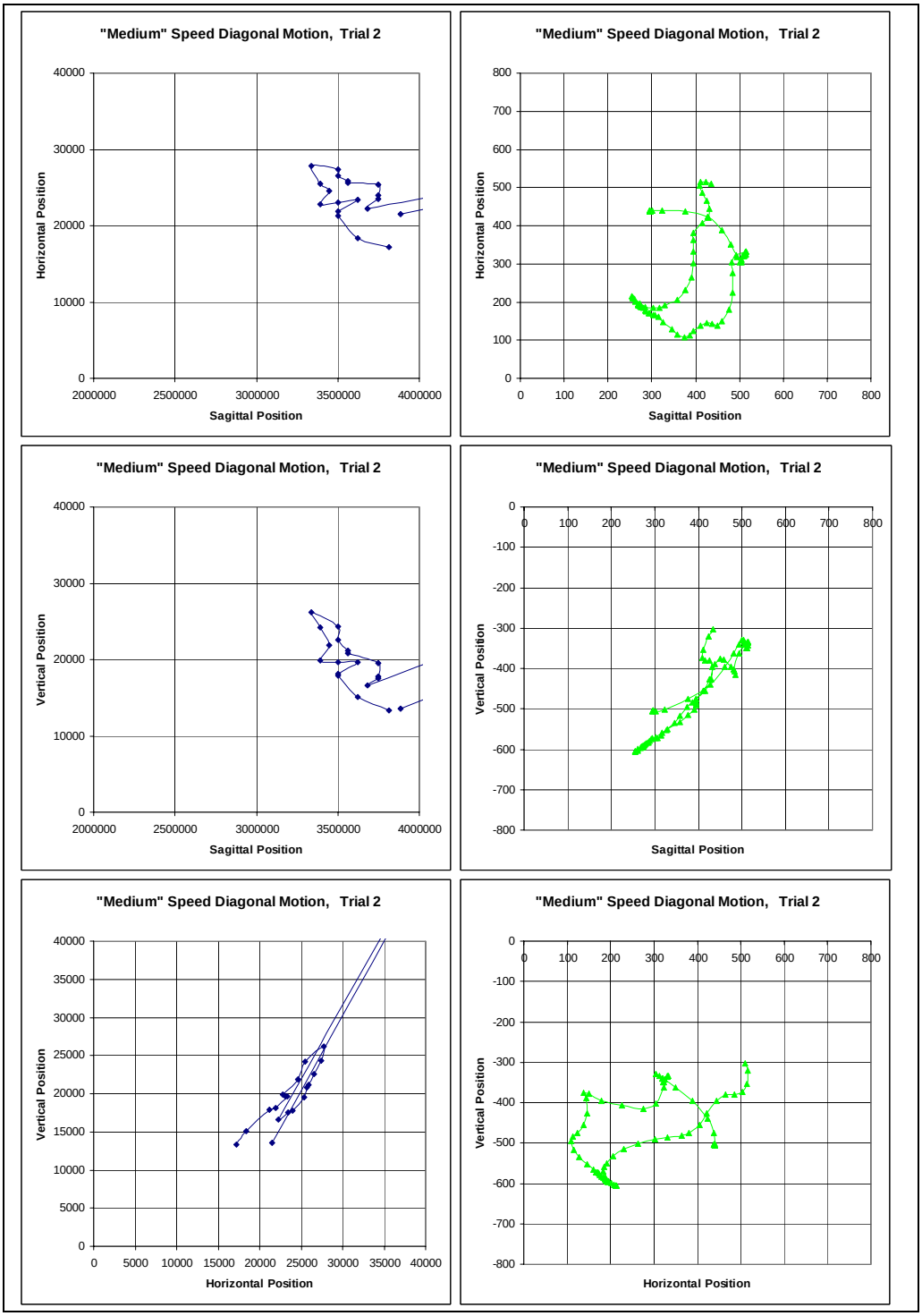
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
293.5664	442.6101	-508.4686	296.2512	438.4392	-504.1063	294.1514	442.0874	-507.727	292.9685	442.9027	-510.7487	292.0979	443.368	-511.8436
293.5664	442.6101	-508.4686	296.0521	438.5587	-504.2207	294.1891	442.0297	-507.6928	293.0062	442.8451	-510.6803	292.1363	443.3106	-511.8094
293.629	442.6093	-508.4114	296.0547	438.5586	-504.289	294.1598	442.2592	-507.5782	293.083	442.73	-510.6461	292.2138	443.1958	-511.8094
293.629	442.6093	-508.4114	295.5742	438.7422	-504.7344	297.4304	444.4831	-505.1694	292.9808	442.7882	-510.6691	292.1751	443.2532	-511.8094
297.3442	444.7165	-505.3503	294.9948	439.2701	-505.2361	319.2486	447.319	-485.4627	292.7144	442.9054	-510.8405	292.0727	443.3112	-511.8323
320.823	448.2744	-488.8753	295.4449	439.3169	-506.5446	372.4301	450.6994	-450.2931	292.637	443.0204	-510.8405	294.1949	443.5215	-510.2497
373.3454	450.4155	-458.2038	301.6518	440.2279	-506.3791	417.6001	451.0198	-418.315	292.8266	443.0184	-510.7033	324.4509	442.6953	-491.09
416.8377	452.9077	-424.2038	323.5106	439.2417	-500.9965	442.352	451.7536	-385.4525	292.56	443.1354	-510.8746	373.2961	418.564	-445.8123
439.4867	454.1522	-384.243	377.3765	437.4492	-474.5474	445.3022	461.8809	-360.4362	292.5218	443.1928	-510.9088	412.9887	378.8921	-414.624
443.3444	463.9409	-354.9597	427.1095	423.766	-440.5813	441.2661	482.6378	-343.9068	292.3087	443.8802	-510.7145	439.1629	322.6768	-400.533
439.9023	482.7071	-336.7134	460.4836	389.4652	-395.8575	436.5691	497.2746	-336.4438	291.6105	444.1704	-511.2397	450.9019	292.1265	-403.4189
437.6463	493.0124	-331.0117	480.5091	349.8585	-363.3311	435.0688	501.858	-337.4559	291.9909	444.168	-511.0339	468.3867	278.4069	-420.3882
436.6495	502.0553	-337.9081	494.1175	319.0176	-340.9917	438.98	495.4397	-340.8334	291.867	444.1688	-511.319	468.9967	292.7232	-427.335
443.2959	506.4431	-349.3741	499.8608	304.1226	-329.7884	448.7268	478.3865	-341.6651	291.7419	444.1696	-511.5357	468.4473	294.7807	-422.0574
445.9124	506.7896	-356.4147	504.4181	304.3751	-329.3578	459.075	431.4837	-345.7752	291.9335	444.1684	-511.5353	471.92	296.1656	-414.7501
448.0142	504.4179	-355.5952	505.35	311.855	-333.8879	467.0654	337.2936	-358.4099	291.6225	444.627	-511.6037	473.7886	300.0111	-392.5521
451.4624	500.5857	-349.8621	506.3182	321.6366	-340.6349	473.7713	205.4433	-381.4857	291.4026	444.6851	-512.6979	461.7048	294.7701	-376.8368
434.8066	488.4727	-338.7045	509.9682	321.4638	-348.347	443.8766	102.2058	-405.2571	291.5313	444.6846	-512.823	436.7845	291.4408	-383.5385
409.5196	452.7058	-334.7682	512.6713	322.1265	-349.2891	395.3063	44.2408	-450.3149	291.5947	444.6843	-512.7659	411.8608	274.7531	-432.4462
414.5173	420.5365	-340.4613	514.2927	325.6043	-342.9575	394.0432	61.80943	-457.2617	291.2775	444.6857	-513.0513	390.5109	216.389	-505.731
432.646	335.9575	-386.7069	514.6613	331.8346	-336.291	380.2798	71.25699	-464.9893	290.9616	444.6871	-513.6441	365.4342	152.1416	-539.0562
428.9108	253.7911	-448.6006	514.2462	332.7079	-334	379.8491	72.79509	-468.6602	290.58	444.6889	-514.0542	341.1957	121.5606	-563.7797
396.4534	190.7888	-510.4008	514.065	332.0701	-332.6952	376.3283	76.54432	-474.5711	290.6055	444.7457	-514.2707	324.6194	135.8828	-582.4823
368.1523	167.9366	-552.5044	512.9603	330.1953	-334.9767	370.6204	82.79389	-485.4676	290.9876	444.744	-513.9289	305.8712	161.6011	-601.8876
349.6613	178.1338	-565.8861	492.7382	321.8999	-363.0277	366.7031	85.54649	-492.6714	290.924	444.7443	-514.0201	292.7785	180.0314	-614.4944
326.2474	196.0758	-581.7514	481.4674	305.0348	-402.2711	363.4419	84.60708	-495.6533	290.8605	444.7446	-514.1112	294.1286	177.5268	-599.3143
304.0052	208.4675	-594.8731	484.3483	276.0131	-415.6788	362.0131	83.01152	-495.7801	290.9241	444.7443	-514.0543	296.6038	171.0549	-573.9177
289.9451	214.2752	-590.3708	483.5458	225.6725	-407.1939	359.6433	82.75475	-495.2608	290.9245	444.7443	-514.1568	293.6638	171.8914	-563.8609
289.6431	209.7681	-579.6503	475.8547	179.362	-395.8575	356.2219	84.85682	-496.7997	290.455	445.4297	-514.1568	284.2831	182.661	-581.2988
289.4235	205.6746	-577.1236	459.0352	149.747	-378.608	354.9067	86.39855	-498.7271	290.2885	445.487	-514.3049	270.8736	200.3013	-598.2864
283.313	216.2244	-582.8152	450.0617	138.0818	-375.1141	353.8349	87.36606	-499.9161	290.416	445.4868	-514.2252	265.4609	208.1644	-602.6766
274.6032	228.6904	-591.9036	437.6552	143.6668	-388.7498	352.8554	86.83367	-499.3666	290.416	445.4868	-514.2252	265.7791	204.0261	-603.6817
269.274	235.6482	-597.9404	425.5117	145.1435	-426.2186	351.6688	86.95306	-499.741	290.416	445.4868	-514.2252	265.3046	200.5035	-603.2655
270.5855	237.5218	-597.0878	410.9593	138.2081	-455.8567	350.5837	86.35972	-500.2722	290.4165	445.4867	-514.4303	262.5887	203.8337	-603.5026
268.8644	240.9391	-597.4397	394.2406	124.3401	-476.081	348.792	86.70442	-502.5672	290.3773	445.5438	-514.4303	258.6945	208.2906	-606.4043
266.3396	245.1735	-600.1004	386.407	113.3064	-483.8028	348.502	86.32977	-502.5933	290.3773	445.5438	-514.4303	257.7887	209.8136	-607.1272
264.2141	249.1983	-601.5962	374.3478	108.7295	-495.9725	348.1821	85.8842	-502.7217	290.3773	445.5438	-514.4303	250.1929	219.8887	-610.4707
263.3231	253.5352	-601.7871	358.8524	115.4805	-516.6834	347.2463	87.16493	-503.6702	292.6156	446	-513.7813	242.5272	233.0278	-618.58
261.7877	256.065	-602.8185	346.033	128.4406	-535.021	346.899	87.22439	-505.2107	316.8192	446.4359	-499.9752	236.898	242.3077	-623.7393
261.1319	257.9553	-603.1832	326.2797	146.8567	-552.2878	346.5854	86.88962	-505.9831	354.7164	442.5247	-461.2684	234.5742	246.0037	-627.1488
261.4049	263.7468	-603.071	314.7497	160.8796	-564.9774	346.2259	86.62407	-506.0331	399.5198	423.8415	-417.3562	231.702	250.4695	-628.7813
261.3565	280.1964	-602.0661	307.0114	166.6207	-571.5035	345.8895	87.01491	-506.9823	427.7263	375.2801	-401.399	227.1434	257.9645	-631.5203
280.2118	291.0798	-598.4268	302.9918	167.1026	-571.7693	345.974	86.54253	-507.1766	446.033	343.4519	-409.915	224.3897	262.3877	-635.5479
306.3771	293.5838	-578.7436	294.5093	170.8997	-573.4482	345.9832	86.18286	-506.9854	464.8587	332.1849	-429.249	220.8078	268.1748	-642.6763
336.6939	303.6679	-550.11	292.2356	171.851	-575.3681	345.9198	86.16325	-507.2542	464.3414	335.6184	-438.1789	217.8775	273.3817	-645.8911
350.8854	332.7918	-528.864	286.9982	174.7324	-579.6227	345.9863	86.29355	-507.596	464.1158	344.003	-433.0666	218.7301	271.7504	-645.5295
355.6454	377.7735	-511.7725	284.2916	178.5186	-582.668	345.9846	86.41139	-507.7964	461.3423	349.2113	-424.7975	220.4288	268.7594	-644.5549
355.9032	418.9932	-500.8058	280.2917	183.838	-585.6474	345.9242	86.27183	-508.0357	459.7068	335.666	-413.3289	220.8832	269.1007	-645.2603
357.8601	457.5678	-486.6066	277.2381	187.4737	-588.6784	345.8583	86.25595	-508.0994	460.5126	308.3848	-388.578	230.4057	255.8751	-647.4238
364.2117	483.933	-462.9732	276.5109	188.2275	-589.4318	340.4191	89.83224	-512.6473	453.192	263.2277	-374.6172	243.3376	237.5112	-640.2684
377.2792	493.5225	-436.6273	275.2724	187.8415	-589.6755	320.6206	114.3949	-536.7314	444.5175	188.6594	-377.1285	265.1778	224.0557	-627.1688
392.5405	497.2994	-405.6235	273.3601	189.5381	-591.2837	297.6027	157.8633	-578.0249	435.338	121.9367	-398.3881	281.3459	228.714	-614.067
407.2187	493.1454	-384.2743	271.7704	191.0445	-592.5429	282.5725	182.3138	-602.6264	407.8916	87.50006	-425.7498	299.1906	246.5415	-596.9342
420.9226	497.0753	-365.9628	270.7573	191.8718	-593.3644	281.0365	184.4028	-600.3519	392.3338	81.67438	-454.9785	311.2776	273.7448	-582.4673
426.175	502.4597	-354.3284	270.2853	192.1303	-593.7979	284.5516	178.4293	-584.9433	384.7473	98.3295	-478.2574	324.9608	300.6117	-569.2627
431.6094	506.0868	-344.295	269.9808	191.4246	-593.6869	282.6255	183.568	-582.2754	371.8565	107.8215	-500.6552	338.6333	337.5998	-549.9411
437.7445	507.5659	-336.1709	269.0397	192.2331	-594.3956	276.5457	202.845	-589.9207	365.7445	113.9335	-513.0516	339.2133	369.8297	-535.8244
439.9615	507.1211	-333.4992	267.7464	193.9662	-595.5887	273.2416	218.4662	-594.8461	351.3265	122.5797	-527.8702	336.6973	400.9316	-525.7611
			261.6137	200.9004	-598.6854	275.7375	227.2103	-593.8448	340.9129	133.7449	-540.8131	332.4901	422.0639	-518.0675
			255.5614	211.1484	-604.5079	277.473	241.4737	-592.9294	320.7876	147.4899	-553.7576	332.1164	433.9267	-512.2515
			254.3984	214.5017	-606.1576	284.5822	260.0516	-589.0265	305.1871	164.4594	-569.6335	338.3823	441.4305	-501.3596
			255.8355	211.6821	-604.9185	299.605	280.8535	-576.5888	296.3949	175.0415	-578.8182	344.422	440.3421	-493.3637
			255.8508	212.5733	-604.634	309.1451	303.4716	-565.589	287.1575	181.9527	-584.6547	351.3578	440.0224	-485.4755
			261.1343	208.6824	-603.5756	315.4091	331.2054	-555.0237	281.3836	189.1089	-589.6538	354.6151	443.219	-480.1303
			272.7201	195.7592	-594.0298	310.5439	359.6573	-550.8435	273.0587	197.9755	-594.7014	370.6404	460.8762	-459.9722
			285.1198	187.8211	-583.8649	304.6478	387.4938	-548.5102	267.3358	205.1545	-599.2304	382.6139	477.383	-416.8826
			303.1665	183.8731	-569.8156	299.3203	414.4243	-543.9881	261.8485	210.8218	-602.9893	392.0384	486.9166	-388.624
			316.939	184.5455	-558.5441	314.4335	436.0287	-528.1268	258.8494	212.9936	-604.8402	398.3338	490.1384	-379.7378
			329.8512	191.1298	-549.8351	339.3962	455.143	-490.4464	252.3814	218.6485	-607.809	400.0066	490.6541	-387.4521
			357.6412	205.069	-533.3853	361.347	460.0712	-459.9383	244.0264	232.0552	-614.3921	406.3814	498.0191	-399.9456
			375.6319	231.0124	-515.0248	371.2635	465.6202	-443.7049	238.9615	240.8506	-621.1543	410.0616	502.3916	-407.6867
			390.7754	263.9984	-501.0365	376.4564	473.9463	-443.8594	232.9068	249.7529	-631.2184			
			395.6415	301.6519	-491.3044	377.436	487.4082	-441.4272	228.0293	257.3016	-641.9292			
			394.6523	331.5957	-486.6462	377.945	502.09	-436.8847	223.0241	265.1504	-645.7034			
			393.9801	363.0311	-481.5557	386.2934	514.0296	-419.4022	219.8235	270.2479	-647.6144			
			395.1445	381.3299	-476.2406	399.361	518.0743	-388.3753	217.4313	274.1423	-648.5095			
			415.24	406.1214	-456.1492	411.1057	514.9042	-364.2371	215.0054	278.0276	-649.2542			
			429.5419	421.9878	-426.512	419.0506	514.6425	-344.7483	213.4593	280.6086	-649.8356			
			430.8687	443.2948	-394.5055	421.2634	516.2688	-334.145	213.87	279.9712	-645.2925			
			425.839	465.6332	-380.7978				214.9541	278.0793	-641.7572			
			415.7278	485.6154	-379.7311				215.44	277.2548	-640.874			
			408.8988	504.6789	-372.9173				215.9076	276.5575	-639.6432			
			412.1924	513.7382	-352.611				219.1971	271.1976	-638.2344			
			422.9071	515.2514	-319.6632				222.5039	265.0225	-632.956			
			434.7595	510.6801	-302.2698				223.5007	264.2233	-634.2094			
									228.9432	256.946	-635.8521			
									244.0924	234.0307	-629.074			

ISAC

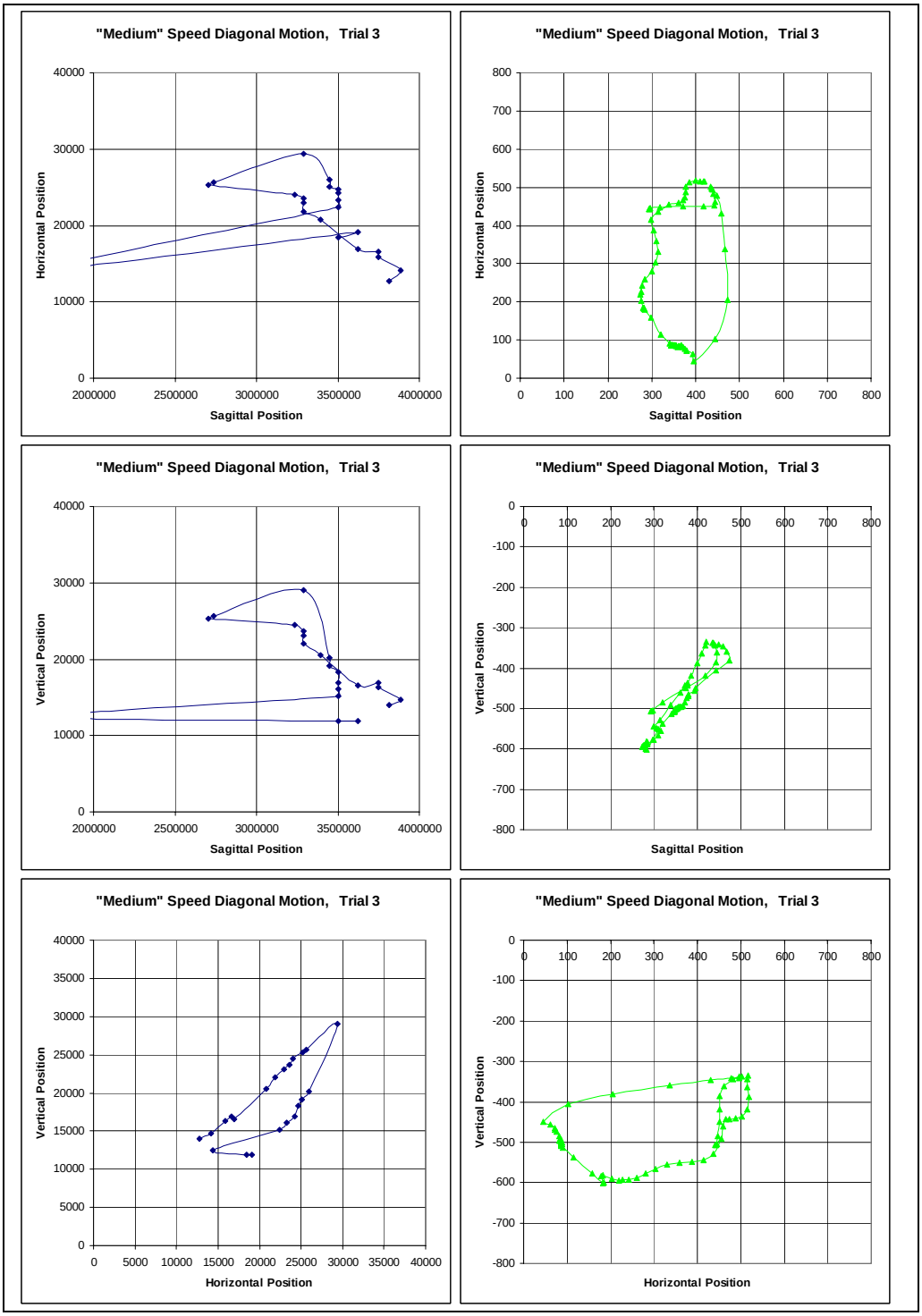
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
									259.2017	211.1545	-617.8195			
									275.614	196.3718	-606.5462			
									287.338	196.3823	-595.9733			
									302.9544	196.621	-585.0741			
									319.3375	194.9651	-571.1857			
									340.7541	200.6218	-554.7107			
									363.4986	219.7014	-534.0722			
									375.953	240.7207	-519.6508			
									385.2822	262.5102	-508.5212			
									392.6926	280.2137	-500.5615			
									403.7764	301.5141	-486.4291			
									420.0649	329.5124	-456.8284			
									430.3247	364.3457	-425.5531			
									428.1884	407.9315	-401.6998			
									420.9019	444.9255	-392.1547			
									420.9957	479.1714	-390.228			
									422.5489	499.0467	-388.0081			
									425.3357	504.6069	-376.4609			
									434.6456	502.8143	-355.8754			
									442.0511	501.5195	-335.9504			
									444.6728	501.84	-319.5441			
									444.0547	503.5411	-316.0125			



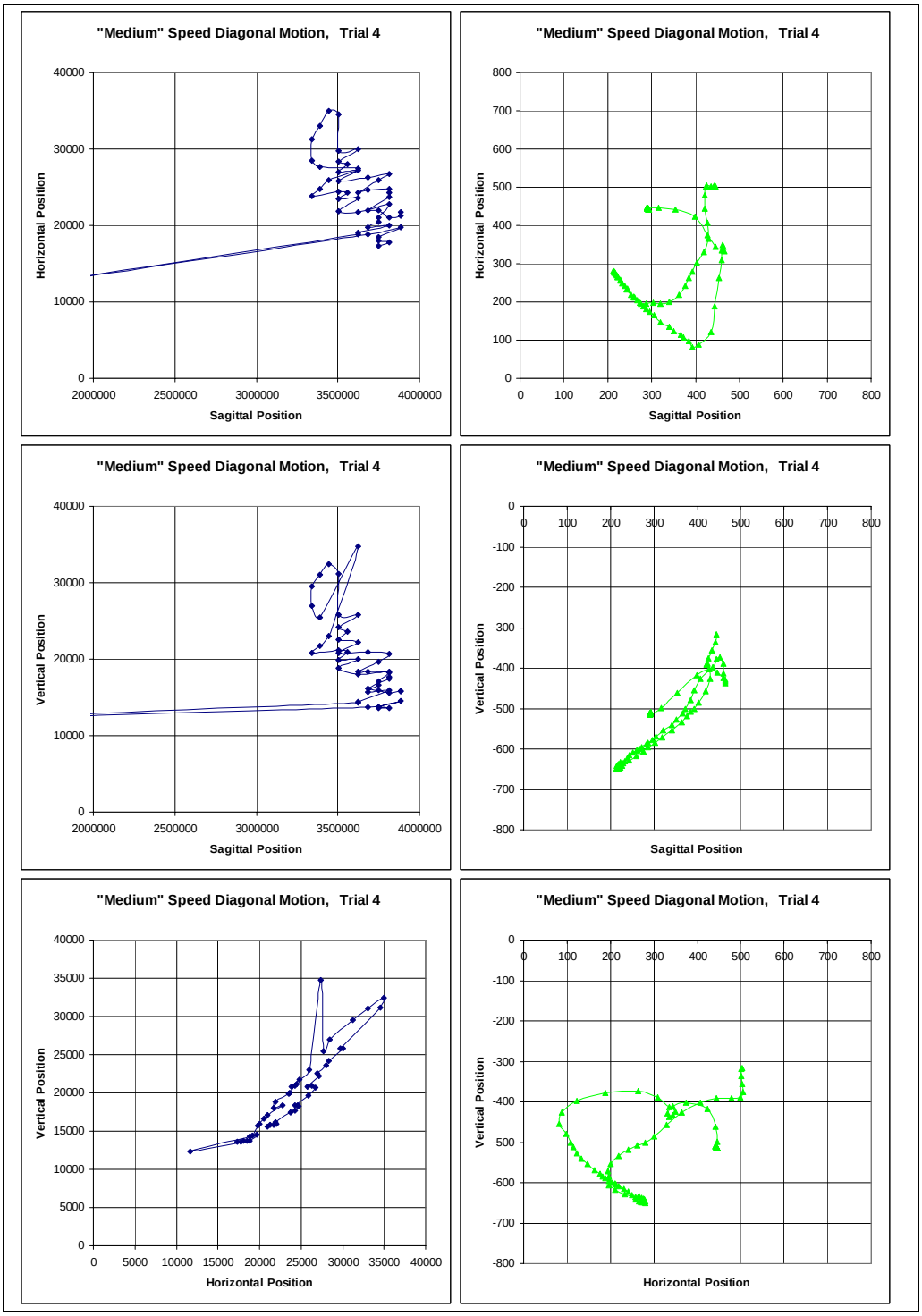
"Medium" speed diagonal motion, trial 1; human motion is shown on the left, that of ISAC on the right



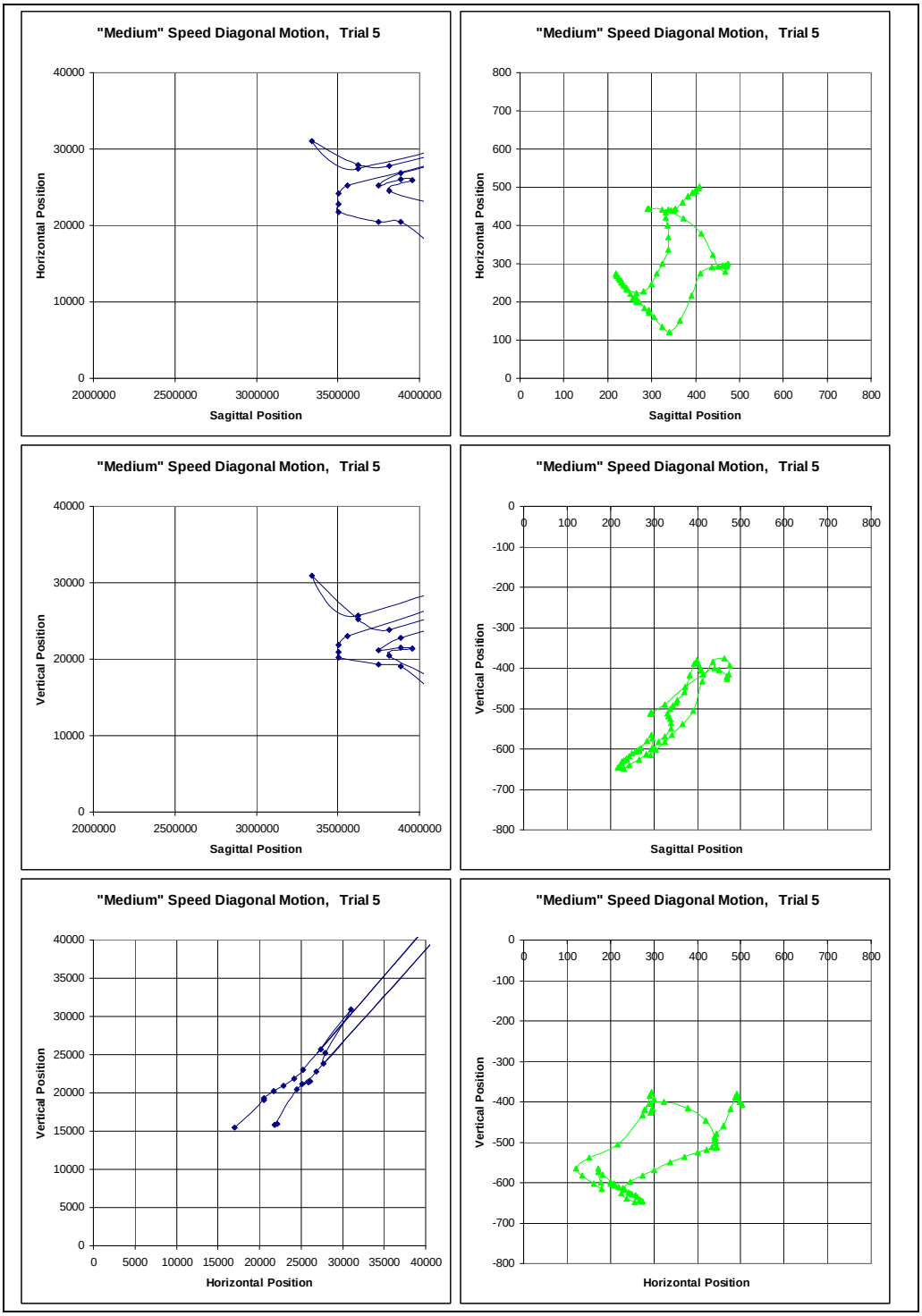
"Medium" speed diagonal motion, trial 2; human motion is shown on the left, that of ISAC on the right



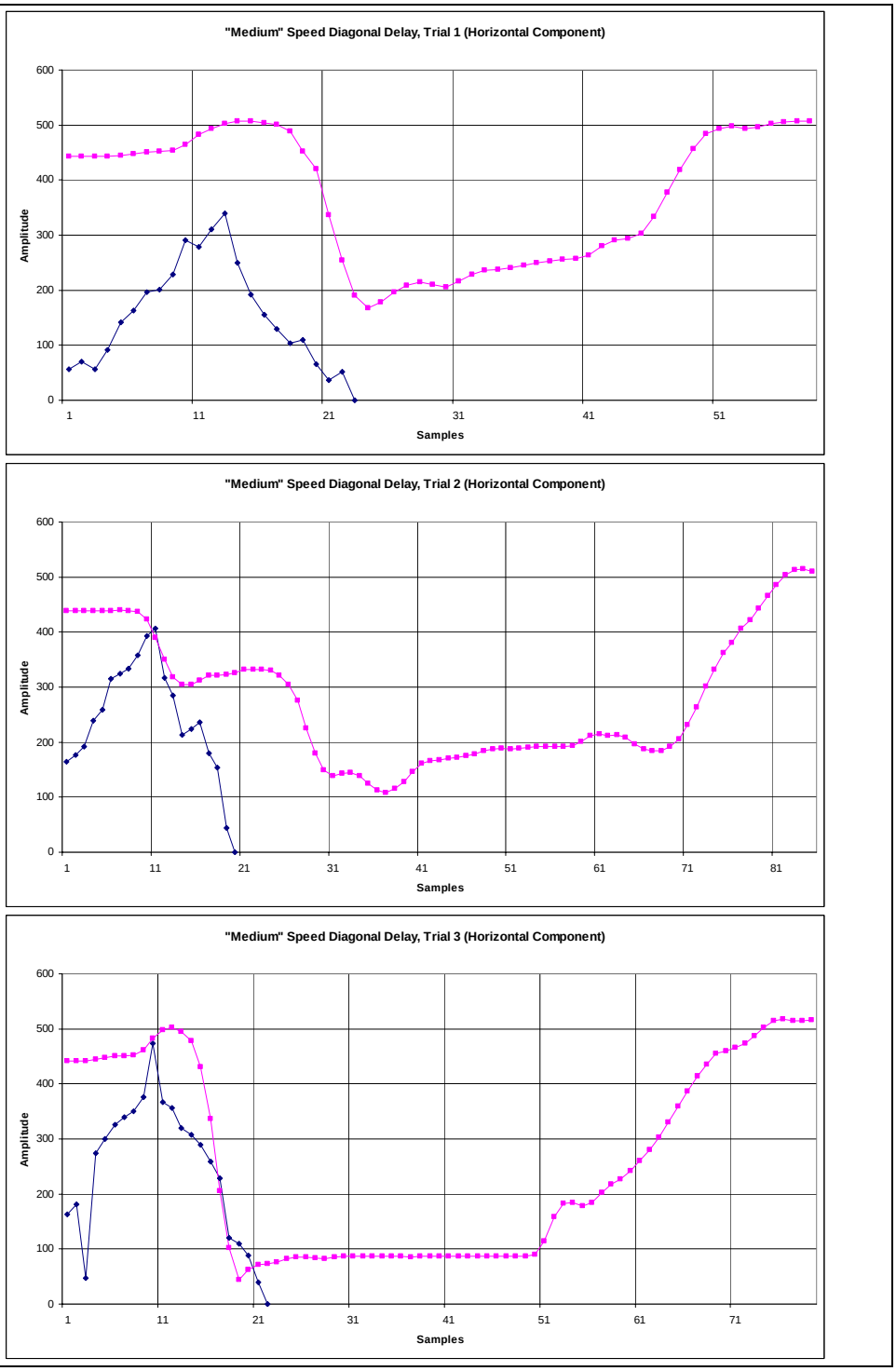
"Medium" speed diagonal motion, trial 3; human motion is shown on the left, that of ISAC on the right



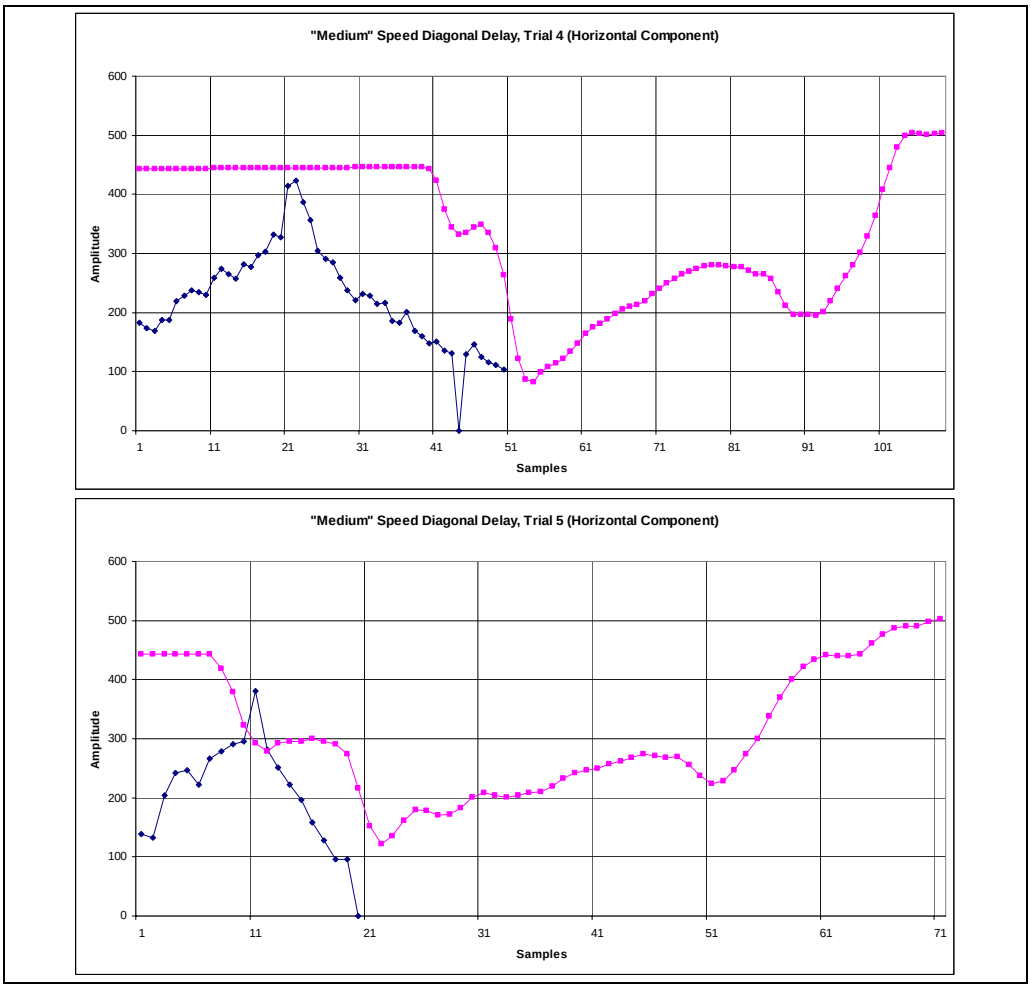
"Medium" speed diagonal motion, trial 4; human motion is shown on the left, that of ISAC on the right



"Medium" speed diagonal motion, trial 5; human motion is shown on the left, that of ISAC on the right



"Medium" speed diagonal delay, trials 1-3; human motion is shown on the on bottom initially, that of ISAC on the on top initially



"Medium" speed diagonal delay, trials 4-5; human motion is shown on the on bottom initially, that of ISAC on the on top initially

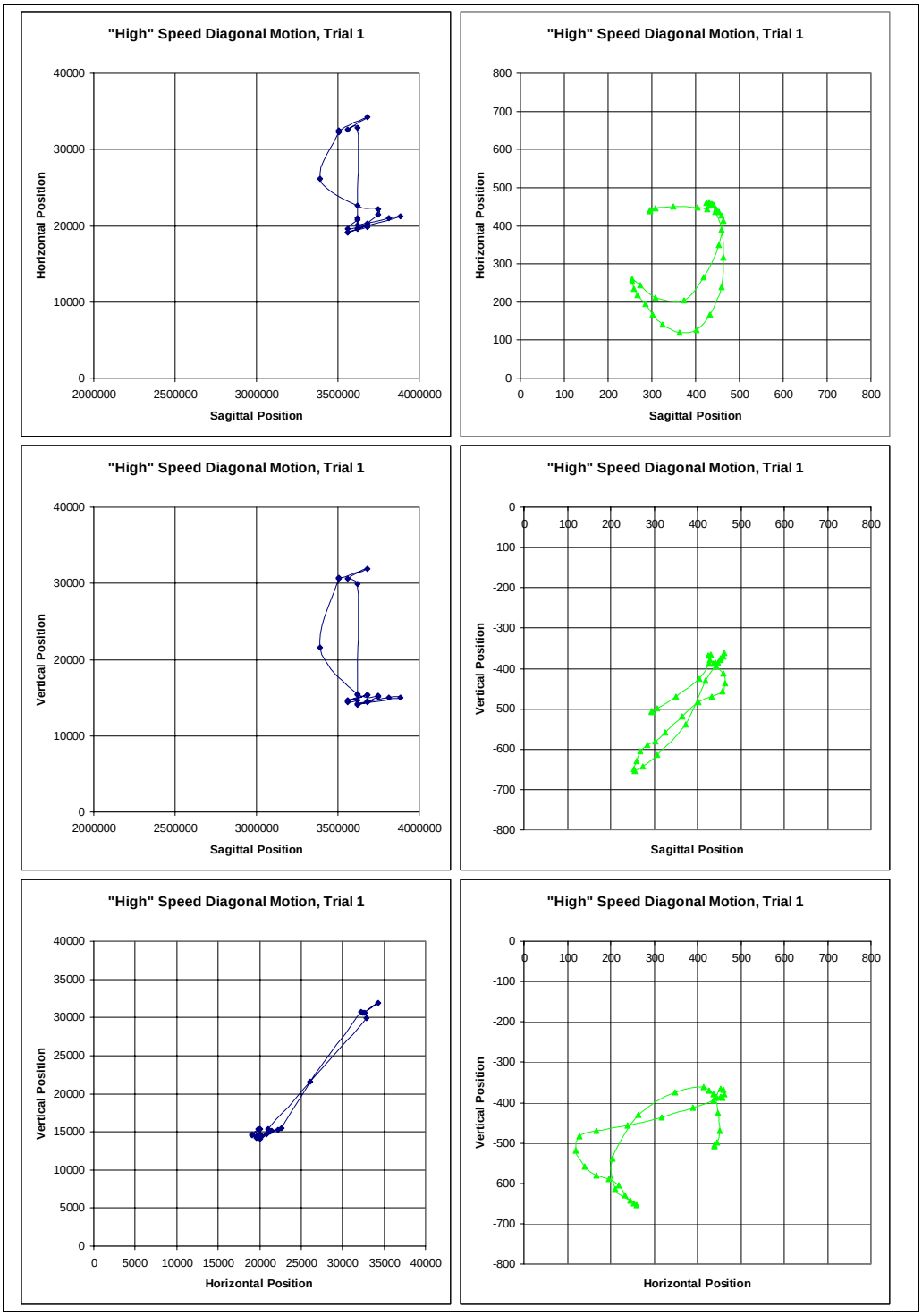
Diagonal motion at "High" speed

Human

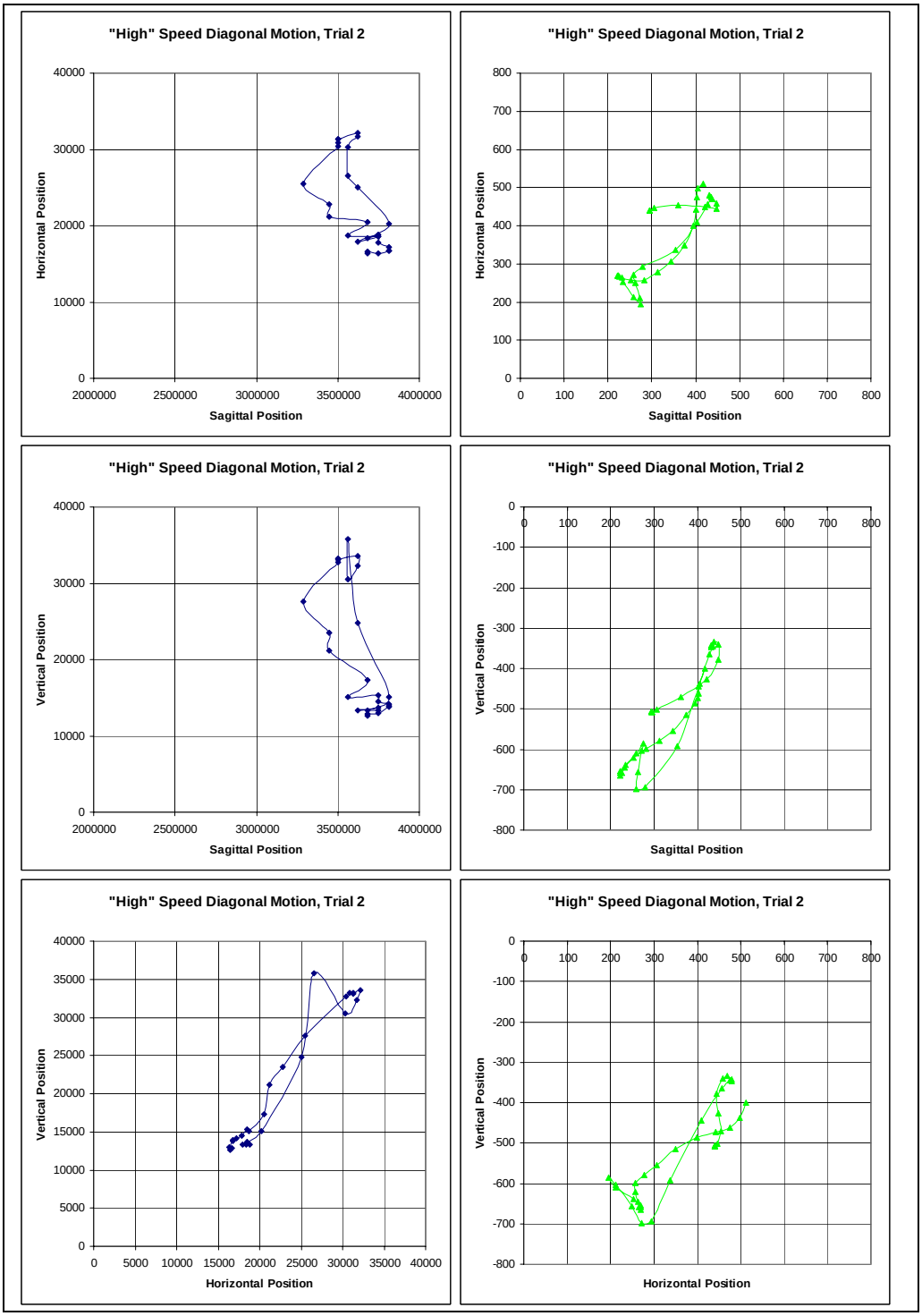
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
3684052	19793.1	15327.59	3684052	18344.83	13275.86	3561250	17266.67	12950	4189706	21000	13176.47	3885000	19218.18	13745.45
3561250	19133.33	14700	3748684	18789.47	13385.96	3561250	17500	12483.33	4031604	20207.55	12679.25	3956945	19444.45	14000
3561250	19133.33	14583.33	3748684	18543.86	13631.58	3502869	17098.36	12508.2	3956945	20222.22	12444.44	3956945	19444.45	13740.74
3684052	20034.48	15327.59	3621610	17915.25	13288.14	3561250	17500	12833.33	3956945	28259.26	25407.41	3885000	19218.18	13490.91
3561250	19600	14466.67	3815625	20250	15125	3446371	17387.1	12532.26	3561250	27066.67	30100	3956945	19962.96	13870.37
3621610	20762.71	14711.86	3621610	25033.9	24796.61	3502869	18016.39	13081.97	3815625	22000	20375	4031604	20735.85	14528.3
3621610	21000	15423.73	3561250	26600	35816.67	3621610	18864.41	14237.29	3815625	20250	18875	3815625	28500	25500
3621610	32864.41	29898.3	3561250	30333.33	30566.67	3684052	20275.86	16172.41	3815625	18750	16625	3748684	29105.26	27140.35
3561250	32666.67	30566.67	3621610	31677.97	32271.19	3621610	24559.32	21949.15	3885000	18200	16163.64	3748684	32052.63	35736.84
3684052	34275.86	31862.07	3621610	32152.54	33576.27	2513824	26105.88	27341.18	3956945	18925.93	15685.19	3621610	30254.24	34050.85
3502869	32475.41	30639.34	3502869	31327.87	33049.18	3287308	30261.54	33169.23	3885000	18709.09	15018.18	3956945	22296.3	22555.55
3502869	32245.9	30754.1	3502869	31327.87	33163.93	3287308	29615.38	33061.54	4031604	19679.25	15320.75	3748684	19280.7	19403.51
3391667	26111.11	21555.55	3502869	30868.85	33163.93	3684052	22689.65	22689.65	3885000	19218.18	14509.09	3815625	17750	16375
3621610	22661.02	15542.37	3502869	30409.84	32704.92	3338672	19468.75	18484.38	3885000	19727.27	14636.36	3956945	17370.37	15944.44
3748684	22228.07	15228.07	3287308	25523.08	27569.23	3502869	17786.88	16983.61	3956945	20222.22	14777.78	4031604	15452.83	14660.38
3748684	21491.23	15105.26	3446371	22806.45	23483.87	3684052	15689.66	13879.31	4031604	20471.7	15056.6			
3684052	20275.86	14482.76	3446371	21225.81	21225.81									
3684052	20034.48	14603.45	3684052	20517.24	17258.62									
3621610	19576.27	14237.29	3561250	18666.67	15050									
3885000	21254.54	15018.18	3748684	18543.86	15350.88									
3815625	21000	15000	3748684	17807.02	14491.23									
3621610	20050.85	14118.64	3815625	17250	14125									
			3815625	16750	13875									
			3815625	16750	13750									
			3748684	16333.33	13017.54									
			3684052	16655.17	12913.79									
			3684052	16413.79	12672.41									

ISAC

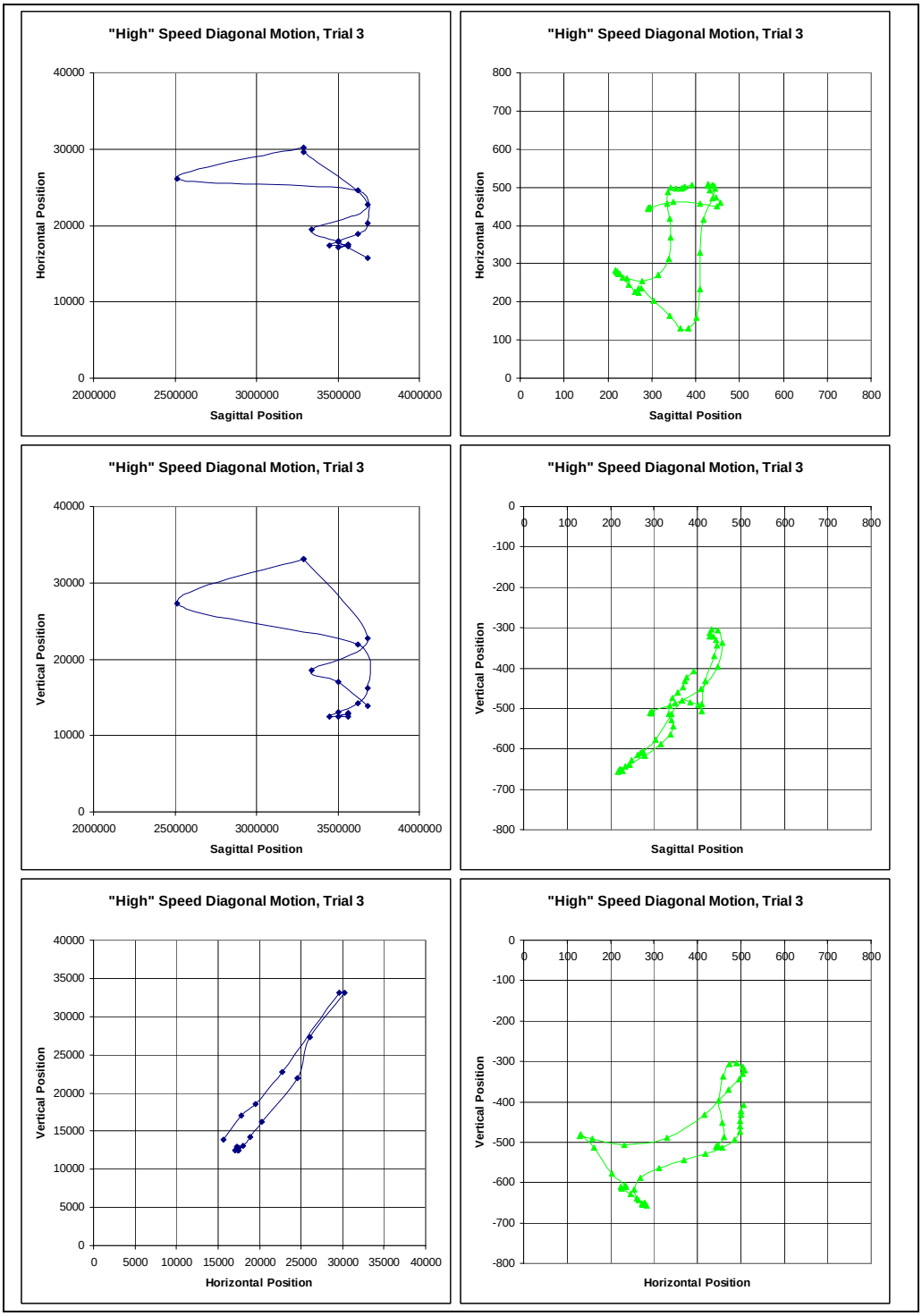
Trial 1			Trial 2			Trial 3			Trial 4			Trial 5		
x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
295.5905	439.4853	-506.6917	295.1789	439.6659	-507.4902	292.0528	445.3692	-510.2329	294.6539	441.2781	-508.9462	294.1818	441.3429	-509.6202
295.8382	439.4798	-506.4285	295.0152	439.7266	-507.5703	292.3555	444.7381	-510.2449	294.9428	441.2161	-508.717	294.0025	441.5173	-509.609
295.73	440.1125	-506.142	294.9686	440.6428	-507.1581	292.4445	444.0508	-510.52	294.7121	441.2772	-508.6839	293.5603	442.5529	-509.3802
307.4762	445.8792	-499.0999	305.6647	445.7598	-500.9324	292.2914	444.7384	-510.2338	294.4859	441.3381	-508.8556	305.3064	448.1044	-503.1202
349.9137	451.2479	-470.2597	361.3203	454.5765	-471.3138	295.8804	447.9227	-507.4663	294.6111	441.3361	-508.7411	360.0923	455.4088	-467.9758
404.5933	447.9899	-424.6376	420.9146	448.8164	-426.8023	348.8038	460.7115	-485.9522	294.5869	441.2792	-508.7984	418.4419	451.1126	-419.6363
426.2181	444.3266	-388.0783	448.7047	444.3263	-377.4957	409.1919	457.9872	-451.7504	304.2278	442.4271	-506.7332	444.1195	446.6984	-371.2546
430.3144	452.9597	-366.3146	448.2599	458.588	-340.3129	447.9056	449.9682	-396.4761	362.141	450.9284	-490.6567	443.1769	457.0354	-342.4798
424.9815	459.2941	-367.4615	437.7964	469.4409	-333.5634	456.8189	459.6456	-337.4939	423.7625	438.289	-452.7301	439.0552	463.2412	-341.0944
429.6286	461.4123	-378.1239	431.0314	478.6254	-342.8051	447.6672	473.8545	-306.3056	458.2685	423.5359	-390.9108	440.501	464.7984	-350.1806
440.8224	454.0022	-386.0317	432.1122	479.6778	-347.7813	432.0724	491.259	-303.3296	469.7427	423.3297	-334.5412	445.9149	458.789	-348.2404
447.8179	442.4865	-385.6234	435.0634	475.4665	-345.8402	428.2107	504.1732	-313.554	462.6483	434.8751	-309.1241	451.2236	455.5321	-337.3702
453.6334	436.5056	-377.9341	427.4395	456.9858	-364.3523	429.0861	507.7556	-321.98	455.171	443.0528	-307.0196	460.8221	441.5724	-326.5305
459.7266	427.8125	-369.1287	402.2313	409.0722	-443.5828	437.7671	506.6394	-322.1226	454.7745	450.6572	-317.653	455.435	411.6204	-343.1994
462.4788	413.8218	-362.1193	353.4806	337.3139	-592.9659	442.1287	504.6985	-330.3743	457.229	451.4948	-323.8515	425.3346	329.281	-437.9635
453.0039	348.598	-373.7033	279.2314	293.3787	-694.9792	444.8507	496.4561	-344.4404	470.8815	451.6585	-324.8687	375.7595	249.4968	-557.9268
418.4374	264.5559	-430.2802	258.2222	272.4609	-699.0044	439.503	470.8624	-369.9984	479.5694	448.1693	-337.9393	320.4679	212.6515	-609.3287
372.8262	203.2276	-537.4741	263.529	249.616	-655.5213	417.5616	416.0398	-431.4453	475.2401	436.607	-358.0221	308.6394	206.3444	-604.4959
307.6662	211.1503	-613.7996	272.3875	210.9522	-603.3203	410.228	329.7237	-489.2494	435.7571	385.8754	-414.2244	293.5665	202.5896	-607.375
273.2762	244.7435	-642.3775	275.4545	195.3022	-585.4609	409.8735	232.692	-507.7486	391.2767	311.859	-501.5786	278.0306	200.5637	-611.5226
254.9839	260.5703	-652.6607	259.3818	211.9801	-610.6366	402.2029	158.5503	-492.2644	365.3292	239.1654	-562.3151	270.1998	210.7309	-601.707
254.2786	253.2577	-649.0252	234.0207	252.5542	-638.4846	383.8473	130.0173	-484.7955	335.428	177.6994	-577.8077	265.0738	217.2394	-596.8143
259.3815	233.5669	-630.3575	223.0087	269.4878	-653.8208	365.2119	131.4579	-481.4763	327.82	149.0482	-569.0116	260.7582	221.5043	-604.5286
268.3391	218.832	-605.4846	221.5197	270.1395	-665.6867	341.3637	162.6854	-513.9842	304.435	152.3426	-577.8094	250.8431	232.4251	-610.9649
285.312	195.0083	-589.3009	223.0303	268.7926	-659.3009	304.5034	202.1338	-577.1477	288.592	174.2008	-591.1669	244.8336	240.6438	-615.0629
302.6979	166.4567	-581.0835	225.3566	266.2407	-657.8883	275.6613	234.4507	-606.9259	277.3349	190.542	-600.5695	243.5216	241.5114	-620.5883
325.0282	139.7003	-558.0151	232.4208	264.1247	-644.8282	270.3471	236.4762	-610.4683	275.4859	190.8609	-599.5124	236.4721	247.4394	-630.6122
364.1334	118.5721	-517.4875	253.0931	256.4377	-620.4633	269.1646	223.5442	-611.4956	274.9668	188.2447	-591.4395	229.1976	259.1032	-641.9708
402.3339	127.7596	-483.4787	282.7557	257.698	-599.8579	262.0987	225.265	-615.7793	270.9624	192.8753	-591.0421	221.3204	271.1348	-654.0961
432.9961	167.108	-470.8094	313.4881	277.8143	-579.1193	247.2836	245.1566	-627.876	265.2847	200.0705	-596.6561	216.6857	277.9909	-660.07
458.4758	239.1623	-457.0312	343.7061	305.4525	-553.9383	233.3783	264.1721	-644.6008	259.5926	207.2851	-601.2185	212.4649	284.3874	-661.5387
463.8267	316.9064	-437.0782	374.0421	349.466	-514.959	226.7259	272.8156	-654.3044	252.9733	217.225	-606.2369	211.5776	285.231	-662.6501
459.4157	389.6867	-411.1406	394.5397	398.8954	-485.5252	224.676	273.5458	-652.9568	246.569	227.6323	-614.975	209.6522	288.0137	-663.8186
444.1937	437.3097	-394.1552	401.8814	441.4329	-473.6542	224.3305	272.4418	-652.5522	238.2621	239.8891	-623.9605	207.7304	292.9187	-665.506
436.6975	456.8749	-388.6037	403.147	475.2305	-461.5905	219.9396	276.5379	-653.9345	234.1551	246.3207	-626.2329	212.6643	289.555	-658.7537
			405.6503	497.2418	-438.201	216.7425	283.0694	-655.6658	232.1148	249.1718	-627.4118	236.5424	268.9107	-648.8812
			416.9713	511.0737	-400.2375	221.1743	279.8954	-650.2246	231.4281	253.4413	-628.0525	270.6563	262.5826	-620.6392
						244.0264	261.4945	-639.0265	237.8468	252.6269	-627.9105	298.8799	287.7881	-596.396
						278.1052	253.6351	-617.5858	271.2016	231.5333	-621.7211	321.4316	331.4005	-576.2954
						315.1893	269.6301	-588.8448	316.4628	231.268	-587.0214	335.1001	379.2962	-552.5787
						338.3485	312.0355	-563.5889	356.0927	260.3802	-549.3553	344.3825	422.0356	-526.6869
						343.7502	368.8575	-544.9438	374.566	319.8835	-522.8732	361.831	464.0295	-497.7508
						340.3759	418.5939	-528.3759	372.8022	380.5631	-507.5486	371.1553	489.709	-474.5612
						334.1228	457.4775	-513.5438	367.2846	440.5373	-489.8646	387.833	496.4561	-446.2606
						336.5127	486.8246	-493.6634	363.0481	484.8812	-461.4882	404.081	498.5037	-415.9266
						342.9354	498.4993	-473.5553	370.3623	506.8741	-428.8849	413.5734	498.5629	-394.6881
						354.7724	497.7016	-460.6022	386.8518	511.6673	-396.3521	416.792	500.8997	-383.484
						367.4022	497.5758	-446.3598	400.3752	507.1059	-375.9532			
						371.7733	499.5254	-432.651	410.2636	502.8552	-361.0127			
						374.8659	501.4071	-423.2538						
						391.907	506.7895	-406.6759						



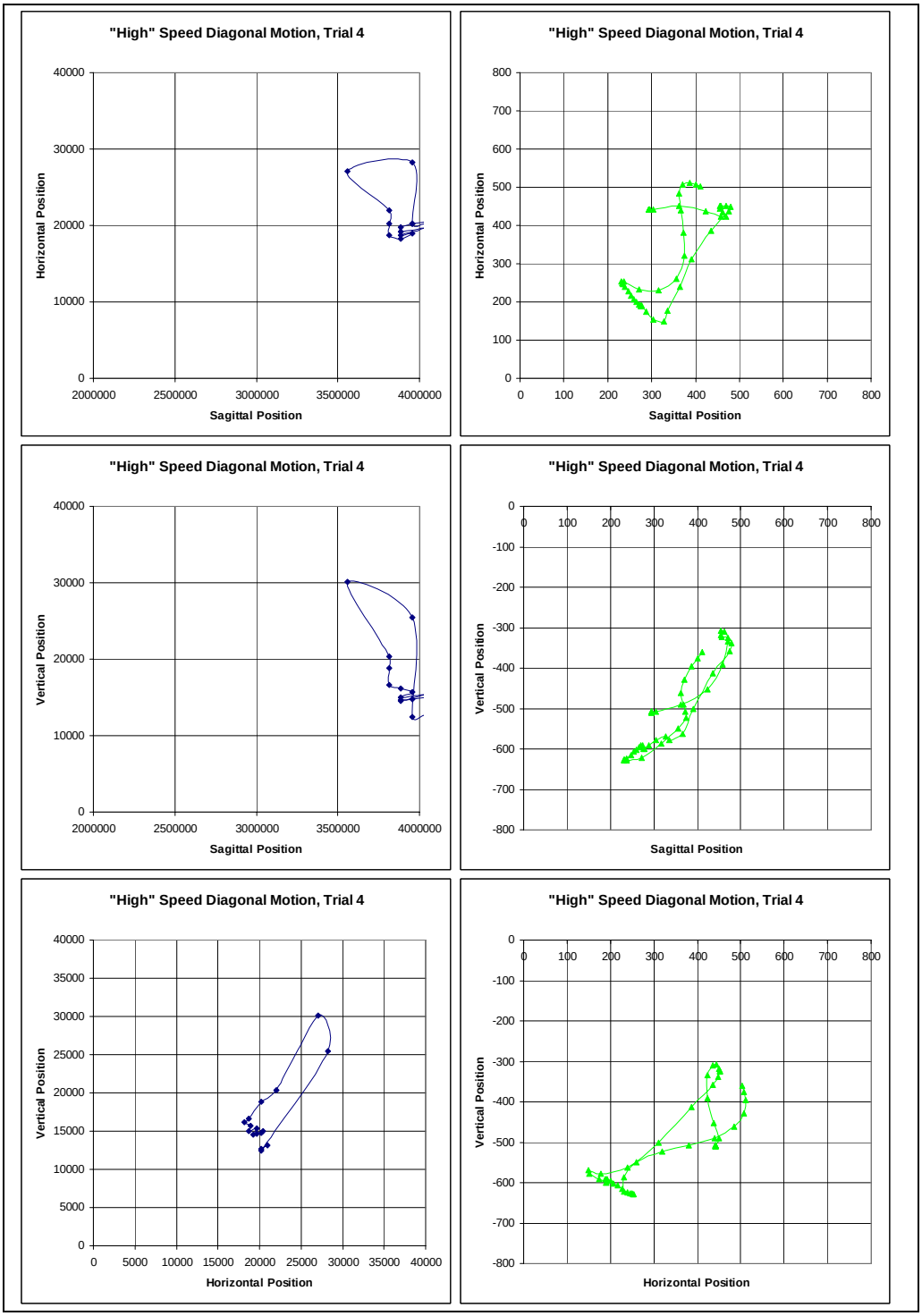
"High" speed diagonal motion, trial 1; human motion is shown on the left, that of ISAC on the right



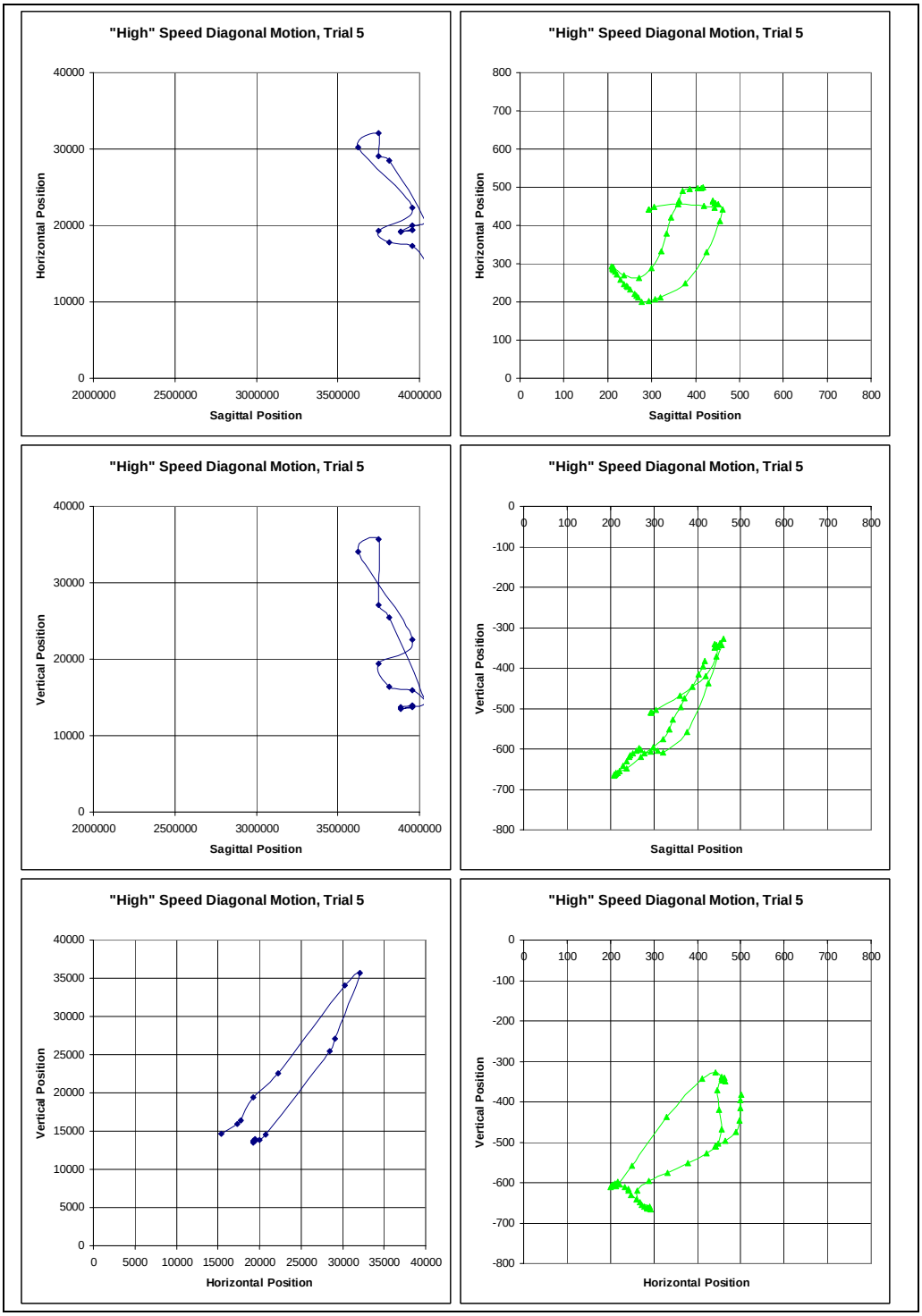
"High" speed diagonal motion, trial 2; human motion is shown on the left, that of ISAC on the right



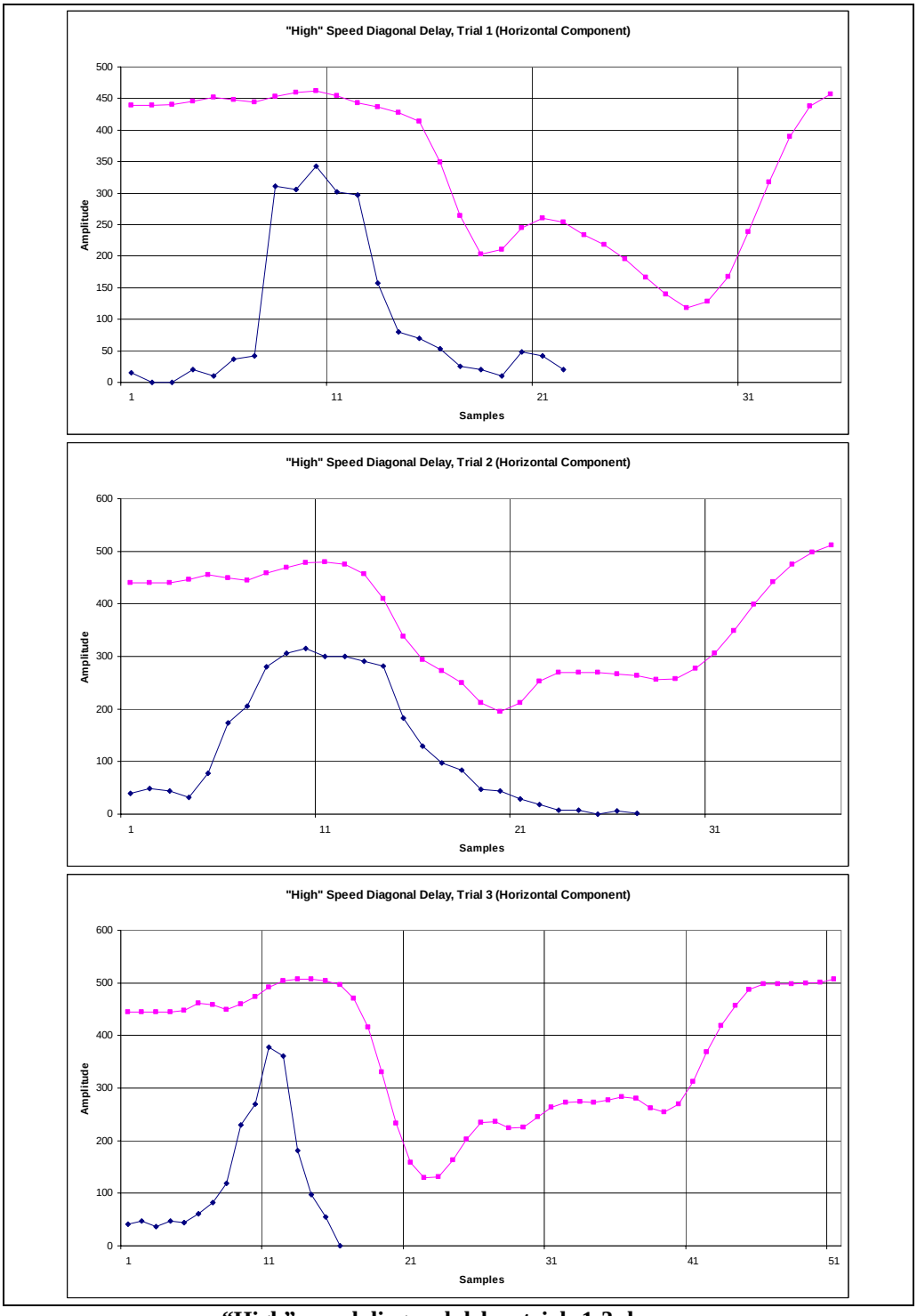
"High" speed diagonal motion, trial 3; human motion is shown on the left, that of ISAC on the right



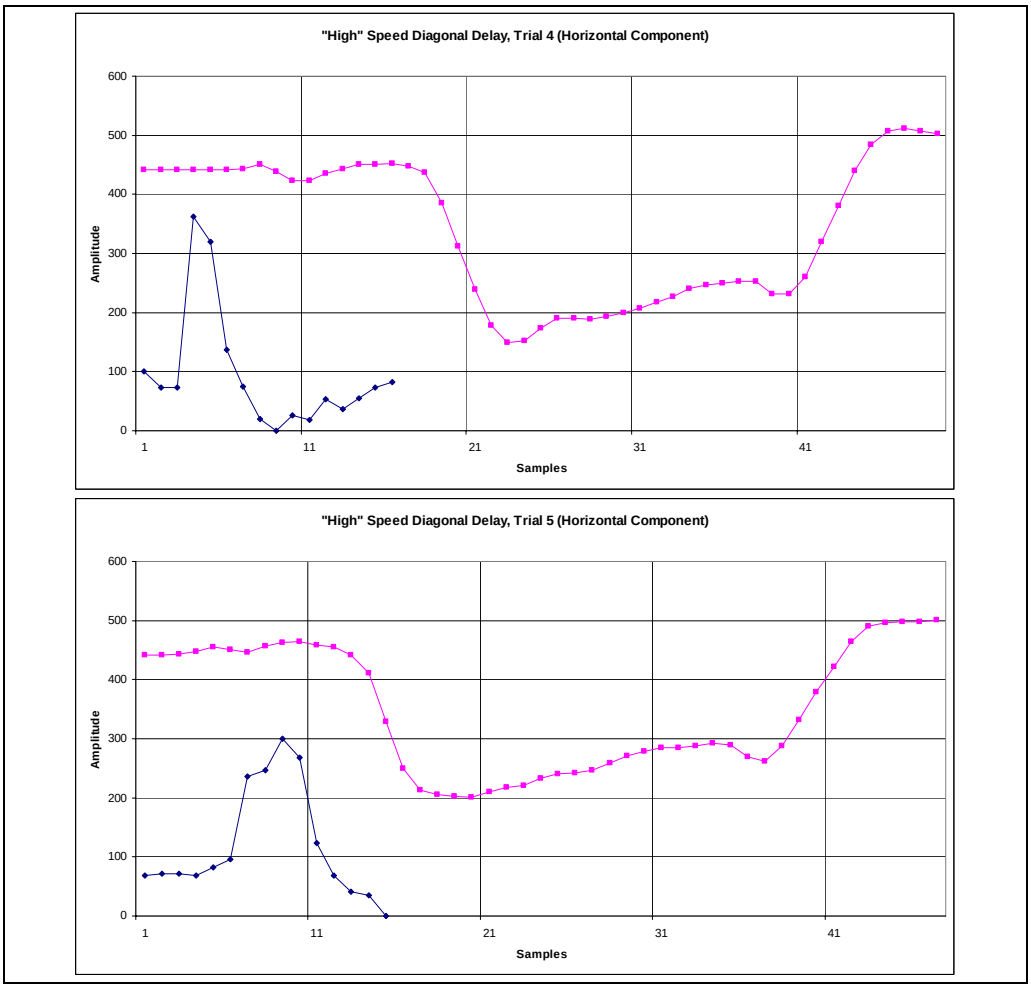
"High" speed diagonal motion, trial 4; human motion is shown on the left, that of ISAC on the right



"High" speed diagonal motion, trial 5; human motion is shown on the left, that of ISAC on the right



"High" speed diagonal delay, trials 1-3; human motion is shown on the on bottom initially, that of ISAC on the on top initially



"High" speed diagonal delay, trials 4-5; human motion is shown on the on bottom initially, that of ISAC on the on top initially

BIBLIOGRAPHY

[1] K. Čapek, *R.U.R.*, in *Toward the Radical Center: A Karel Čapek Reader*, North Haven, CT: Catbird Press, 1990.

[2] I. Asimov, *I, Robot*, New York, NY: Gnome Press, 1950.

[3] A. Vivas and V. Mosquera, "Predictive Functional Control of a PUMA Robot," in *Proc. ACSE 05* Conference, pp. 19-21, Dec. 2005.

[4] Motoman.com, "Motoman Industrial Robots." 19 Feb. 2009, http://www.motoman.com/products/robots/default.php.

[5] O. Khatib, L. Sentis, J. Park, and J. Warren, "Whole Body Dynamic Behavior and Control of Human-like Robots," in *International Journal of Humanoid Robotics*, vol. 1, no. 1, pp. 29-43, 2004.

[6] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka, "The Development of Honda Humanoid Robot," in *Proc. IEEE International Conference on Robotics and Automation 1998*, vol. 2, pp. 1321-1326, 16-20 May 1998.

[7] C. S. Kim, W. H. Seo, S. H. Han, and O. Khatib, "Fuzzy Logic Control of a Robot Manipulator Based on Visual Servoing," in *Proc. IEEE-ISIE 2001*, vol. 3, pp. 1597-1602, 12-16 June 2001.

[8] B. Ulutas, E. Erdemir, and K. Kawamura, "Application of a Hybrid Controller with Non-Contact Impedance to a Humanoid Robot," in *Proc. IEEE International Workshop on Variable Structure Systems 2008*, pp. 378-383, 8-10 June 2008.

[9] S. J. Huang and C. F. Hu, "Neural Network Controller for Robotic Motion Control," in *The International Journal of Advanced Manufacturing Technology*, vol. 12, no. 6, pp. 450-454, Nov. 1996.

[10] R. Zhu and Y. Yamane, "A Robust Control Scheme for Robot Manipulator by Using a Genetic Algorithm," in *Proc. ISCIE International Symposium on Stochastic Systems Theory and its Applications 2001*, vol. 33, pp. 261-266, 2001.

[11] S. Schaal, "Is Imitation Learning the Route to Humanoid Robots?" in *Trends in Cognitive Sciences*, vol. 3, pp. 233-242, 1999.

[12] S. Schaal, "Learning Robot Control," in *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, ed., 2nd ed., Cambridge, MA: MIT Press, 2002, pp. 983-987.

[13] A. Billard, "Learning Motor Skills by Imitation: A Biologically Inspired Robotic Model," in *Cybernetics and Systems: An International Journal*, vol. 32, pp. 155-193, 2001.

[14] Vanderbilt.edu, "CRL ISAC," 14 Feb. 2009, http://eecs.vanderbilt.edu/CIS/CRL/isac.shtml.

[15] Sony Electronics Inc., *Color Video Camera Module*, Park Ridge, NJ: Sony Corporation.

[16] S. Begley, "Gesture Recognition and Mimicking in a Humanoid Robot," MSEE Thesis, Vanderbilt University, 2008.

[17] Directed Perception, *Model PTU-D46*, Burlingame, CA: Directed Perception, 2009.

[18] DPerception.com, "Directed Perception: PTC-D46-17 Model," 4 Feb. 2009, http://www.dperception.com/ptu_D46_view.html.

[19] Imagenation Corporation, *PXC200 Precision Color Frame Grabber*, Beaverton, OR: Imagenation Corporation, 1997.

[20] Imagenation.com, "PXC Frame Grabber Family," 4 Feb. 2009, http://www.imagenation.com/pxcfamily_img.html.

[21] R. H. Gaylord, "Fluid Actuated Motor System and Stroking Device," United States Patent 2 844 126, 1958.

[22] G. K. Klute and B. Hannaford, "Accounting for Elastic Energy Storage in McKibben Artificial Muscle Actuators," in *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 122, no. 2, pp. 386-388, 2000.

[23] T. Tsuji, S. Miyata, T. Hashimoto, and H. Kobayashi, "Controller Design for Robot with Pneumatic Artificial Muscles," in *SICE-ICASE 2006*, pp. 5419-5422, 18-21 Oct. 2006.

[24] ShadowRobot.com, "Shadow Robot Company: Air Muscles Overview," 6 Feb. 2009, http://www.shadowrobot.com/airmuscles.

[25] SMC, *Series ITV*, Indianapolis, IN: SMC Corporation of America.

[26] "Encoder Basics," *Danaher Precision Systems*, 19 Feb. 2009, http://www.neat.com/products/electronics/pdf/EncoderBasics.pdf.

[27] VitalSystem.com, "Motion Control PCI Card," 7 Feb. 2009, http://www.vitalsystem.com/web/motion/motionLite.php.

[28] VitalSystem.com, "Motion Control Breakout Boards," 7 Feb. 2009, http://www.vitalsystem.com/web/common/breakout.php.

[29] VitalSystem.com, "Termination Board pn 7525 for MotEnc-Lite," 7 Feb. 2009, http://www.vitalsystem.com/web/motion/um7525.pdf.

[30] SourceForge.net, "SourceForge.net: Open Computer Vision Library," 2 Feb. 2009, http://sourceforge.net/projects/opencvlibrary.

[31] G. Bradski and A. Kaehler, *Learning OpenCV*, Sebastopol, CA: O'Reilly Media, Inc., Sept. 2008.

[32] WillowGarage.com, "CvReference – OpenCV Wiki," 3 Feb. 2009, http://opencv.willowgarage.com/wiki/CvReference.

[33] WillowGarage.com, "CxCore – OpenCV Wiki," 3 Feb. 2009, http://opencv.willowgarage.com/wiki/CxCore.

[34] WillowGarage.com, "HighGui – OpenCV Wiki," 3 Feb. 2009, http://opencv.willowgarage.com/wiki/HighGui.

[35] WillowGarage.com, "MachineLearning – OpenCV Wiki," 3 Feb. 2009, http://opencv.willowgarage.com/wiki/MachineLearning.

[36] Intel.com, "Intel® Integrated Performance Primitives 6.0," 4 Feb. 2009, http://www.intel.com/cd/software/products/asmo-na/eng/302910.htm.

[37] A. Haar, "Zur Theorie der orthogonalen Funktionensysteme," in *Mathematische Annalen*, vol. 69, pp 331-371, 1910.

[38] P. Viola and M. J. Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features," in *Proc. IEEE-CVPR 2001*, vol. 1, pp. 511-518, 2001.

[39] R. Lienhart and J. Maydt, "An Extended Set of Haar-like Features for Rapid Object Detection," in *Proc. IEEE-ICIP 2002*, vol. 1, pp. 900–903, 22-25 Sept. 2002.

[40] N. Seo, "Tutorial: OpenCV haartraining (Rapid Object Detection With A Cascade of Boosted Classifiers Based on Haar-like Features)," 19 Feb. 2009, http://note.sonots.com/SciSoftware/haartraining.html.

[41] J. Wachs, H. Stern, Y. Edan, M. Gillam, C. Feied, M. Smith, J. Handler, "A Real-time Hand Gesture Interface for Medical Visualization Applications," in *Applications of Soft Computing: Recent Trends*, A. Tiwari, J. Knowles, E. Avineri, K. Dahal, and R. Roy, eds., Verlag, Germany: Springer, 2006, pp. 153-163.

[42] HandSpeak.com, "HandSpeak: ASL A," 10 Feb. 2009, http://www.handspeak.com/spell/index.php?abc=asla.

[43] M. J. Donahoo and K. L. Calvert, *TCP/IP Sockets in C*, San Francisco, CA: Morgan Kaufmann, 2000.

[44] Baylor.edu, "Practical TCP/IP Sockets in C," 16 Feb. 2009, http://cs.baylor.edu/~donahoo/practical/CSockets/c++.html.

[45] MovesInstitute.org, "HandVu: Hand Gesture Recognition," 5 Feb. 2009, http://www.movesinstitute.org/~kolsch/HandVu/HandVu.html.

[46] P. Ratanaswasd, W. Dodd, K. Kawamura, and D. C. Noelle, "Modular Behavior Control for a Cognitive Robot," in *Proc. ICAR 2005*, pp. 713-718, 18-20 July 2005.