Human Action Segmentation and Recognition with a High Dimensional Single Camera System

By

Jonathan Edward Hunter

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

May 2009

Nashville, Tennessee

Approved:

Professor Don Mitchell Wilkes

Professor Kazuhiko Kawamura

Professor Richard Alan Peters II

Professor Daniel Levin

Professor Megan Saylor

TABLE OF CONTENTS

Page

ACKNOWLEDGEMENTS

First and foremost, I thank God for guiding and protecting me every step of the way in my life, for keeping me in my right mind and at peace.

I would like to state my sincere appreciation to my advisor, Dr. Wilkes. His guidance made this dissertation would not have been possible, and his wonderful personality taught me the necessary qualities to be an outstanding teacher and mentor. I would also like to thank my committee, Dr. Kawamura, Dr. Peters, Dr. Levin, and Dr. Saylor, for providing insightful and firm guidance towards my dissertation and research. I would also like to thank those that supported me and guided me through my graduate career. Thank you to Dr. Cordelia Brown, Assistant Dean Burgess Mitchell, Dean Veilette, Missetha, Dr. Ed Brown, and Dr. Tamera Rogers.

Thank you to my parents, James and Carol, for engraving in me the importance of education, the necessity to strive to be the best I can be, and the passion to help as many as I can to achieve greater accomplishments in their education. I thank my brothers, Jaime and Jason, for all the laughter and love they gave me. I thank all my family for the love, support, guidance, and free food that made this possible.

Thank you, Flo Wahidi, for always looking out for me and providing great advice to get me through school and for life after graduate school.

Thanks to the gamer crew who supplied a large part of the great memories that I will take from Nashville. Thanks Beely Bounedara, Ian Davis, and Charlie Dycus.

Thanks to Dr. Oluwole Amusan for keeping me on task and listening to all my “imaginative” scenarios. You definitely have my deepest thanks for being a great friend as we went through this graduate school thing.

Thanks to all my friends that I met at Vandy during the years. Thanks Alice Davenport, Jessilyn Chatman, Candice Muhammad, Alette Davis, Heather Holmes, Sharlene Lewis, and Steve Cotton.

Thank you to my friends who are a pillar of support for me. Thank you and congratulations to all the other doctors who I graduated with Dr. Stephen Gordon, Dr. Paul Fleming, Dr. Katherine Fleming, Dr. James Hill, Dr. Hande Keskinpala, Dr. Turker Keskinpala, Dr. Juan Rojas, Dr. Karthik Subramanian, and Dr. Anupama Subramanian. Thank you to Jon Ahlbin, Jia Bai, Anitha Balasubramanian, Chris Costello, Joe Hall, Natalie Han, Fred Hillard, Atakan Varol, and Birhan Woldegiorgis.

Thank you Brenda, Steve, Anthony, and Benita.

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

Human action segmentation is one of the many unsettled topics of perceptual studies. One definition of an action is to do a task or deed. Although this definition is simple, the description needed to include all possible forms of action would be significantly longer. In fact, it is so elusive that there is no standard explanation applicable to all cases, especially in the fields of computer science, signal processing, and robotics. This is why the area of action segmentation is a major field of study within these communities. Of course, actions can be performed by living and non-living entities alike. The study of action among living entities brings to light the concept of purpose. Many actions performed by living beings are done for a purpose whether that purpose is to relay information among living things, to perform some necessary task for self-preservation, to explore the surrounding environment, etc. The types of action vary from very fine movements such as writing to entire body activities such as walking. This purpose of the action is the "needle" to be extracted from the "haystack" of all the visual input available. Extracting purpose from actions is one of the primary tools for learning among humans. It is done everyday with seemingly minimal effort. How is this done?

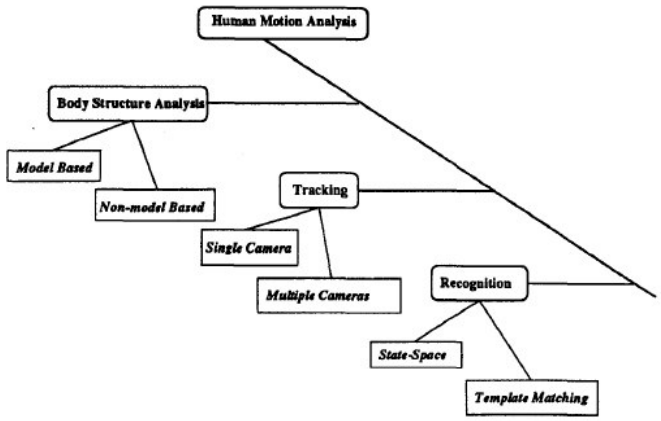


Figure 1: Research Tree for Human Motion Analysis [1]

In the review by Aggarwal and Cai [1], human motion analysis can be broken up into 3 areas, body structure analysis, tracking, and recognition as shown in Figure 1. Body structure analysis is concerned with the characteristics of the "object" to be tracked. The body structure can be represented as *a priori* knowledge in the form of a model or it may be considered to be unknown, that is to say, a "blob" that contains identifying characteristics. In tracking, there are 3 general types of systems that extract information: invasive systems that involve applying sensors directly to the person, non-invasive systems that involve a single camera, and non-invasive systems using multiple cameras. The final area, recognition, is typically addressed via template matching or state space methods. Template matching takes the motions of the tracked item and compares them to a database of training actions recorded in the same manner. State space methods identify certain static postures and actions as states and apply probability to analyze these states to provide estimations of tasks. The state space methods take advantage of Kalman filters, Hidden Markov Models (HMM) and Bayesian Networks. In the upcoming chapters, these areas with specific examples will be reviewed.

Objective of Vision System

The goal of the vision system developed for this project is to be an easy-to-use tool for training and tracking to aid in analysis of video recordings of experiments. Earlier versions of this visual system were used in papers studying the use of applying a Working Memory application to robotic motions [51]. The system is being used currently to assist analysis of intentional human motion in the Vanderbilt Psychology Department. The system is a non-invasive tool consisting of a single stationary camera using a non-model-based blob detection algorithm. The system segmentation uses an approximate nearest neighbor tree to search for though feature vectors composed of color histograms and texture measures. The user interface is designed to be fairly simple to decrease

difficulties in training and tracking. The system extracts information about the segmented frames to be further analyzed.

One of the contributions of this work are that this system will be the first system to implement our high dimensional sparse feature vector extraction method with applications to the intentional vision research done by the Vanderbilt Psychology Department. The methodology for achieving autonomous segmentation by merging the minimum spanning tree, approximate nearest neighbor tree, and normalized high dimensional sparse feature vectors provides very nice results in natural and controlled environments. Using both the supervised and autonomous systems, an extensive analysis showed that this approach is able to capture behavioral cues and is shown to correlate with the original human rated behavioral cues of the intentional vision research. Another contribution is the detailed analysis of the autonomous system algorithms.

Organization of Dissertation

The organization of this paper will provide a literature review for each of the discussed sections above. Chapter II will cover the methods of body structure analysis. Chapter III will cover the systems used for event tracking. Chapter IV will cover single camera vision techniques with respect to human motion tracking. Chapter VI will discuss the current system implemented for our research. Finally, Chapter VII will present the experiments and results.

CHAPTER II

BODY STRUCTURE ANALYSIS

Full Body Analysis

Examples of full body actions are actions such as walking, jumping, squatting, etc. In some studies, these actions are studied strictly to track and identify the action itself [8,9,11,20]. Other studies incorporate these actions combined with multiple objects [6,22,41]. The full body structure typically uses an invasive procedure to gather the specific correlated points on the person to create a model by using sensors attached directly to the person. Other non–invasive measures involve tracking visual characteristics. Methods of tracking the full body motion range from monitoring key body parts such as head detection, recording the motion throughout a task, blob detection, etc. Each of these methods carries a set of weaknesses. Motion recordings are usually sensitive to differing trajectories and temporal spacing of people when performing the same task. A large number of the visual systems implemented are sensitive to view angles and lighting conditions causing significant error in point calculations. There are a few systems devoted to view-invariance. Many of the studies involve a very controlled motion [6,25,87]. For problems involving gathering task information to mimic by robotic manipulators, the demonstrator's actions are severely limited to a specific manner of manipulating objects [5,17].

Body structure analysis uses a variety of body parts to extract information. There has been quite a bit of work done with tracking the entire body as well as tracking specific parts such as the hands or the head. It makes sense that the main cues for action are often related to the manipulators (hands) and the visual sensors (eyes). For example, gaze may be estimated by head direction when analyzing human actions. By using features or models of these parts, the body can be tracked across successive images. The necessary steps for extracting information are identification of temporal

boundaries of action, identification of relevant characteristics, extraction of temporal activities of objects, and extraction of purpose. A significant amount of work has been focused on each of these areas. The first area to focus on is the tracking of full body motions. The full human body is modeled according to two main methods: the use of skeleton models (2D or 3D) and the use of motion templates/optical flow.

Full body motions are gestures of the human body such as walking, sitting, etc. These activities are studied by analyzing the entire body for the structural changes as a task is performed. The skeleton models consist of significant points on the human body and are connected by either lines or shapes depending on the 2D or 3D modeling method. Some use invasive techniques to collect the points, usually involving the subject wearing sensors or indicators on their person that are used to provide precise tracking [86]. Some use natural visual cues of interest in the environment to detect the significant points [6,11]. The methods for visual detection vary from using, often complex, multi-camera analysis to using simpler single camera view dependant analysis. Some of the techniques for different camera setups will be discussed later in the paper. Most of the visual 3D modeling requires multiple cameras filming the subject. The cameras are usually stationary. The precise position, orientation, and pixel size of the cameras are known with respect each camera. By correlating the frame by frame position of the markers from the multiple cameras, 3D positions are extracted from the video sequence. Using these sequences of positions, shape models can be applied to the spaces between points and various data can be extracted about the motion. By placing sensors on subjects, significantly less noise is involved with the data than with data gathered from strictly visual means. A large constraint in using markers is that the environment must be modified creating a more unnatural scenario.

Applying 2D models to extracted silhouettes from images is another commonly used method [40,50]. Aggarwal et al. [87] used motion to initialize a model over the silhouette of a person, and

using forward kinematics, is able to continue tracking the silhouette by applying a cost function based on the amount of motion necessary to overlay the model (see Figure 2).

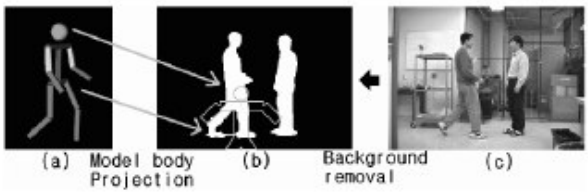


Figure 2: Overview of extracting silhouette and overlaying the model [87]

Aggarwal et al. used this tracking to extract information about the position and velocity of a person's appendages to determine if dangerous activity is occurring (such as fighting). This work was continued [25] by applying the data to finite state automata to detect sequences of states for the human models to detect motions such as pushing (see Figure 3).

Figure 3: A sequence of tracked model states indicating a push [25]

The states to be detected as part of the action of pushing include the rising of the hands and a fast forward motion of the arms for the attacker, followed by the negative motion of the victim. They applied automata to detect actions such as walking, kicking, pointing, pushing, handshake, etc. In another study done by Peursum et al. [6], they use a 2D quick star skeleton method on the silhouette of a person at differing camera angles to extrapolate a 3D model over the person. The star skeleton places points at the extremities of the person and connects those points to either the upper or lower points of the center spine (shown in Figure 4).

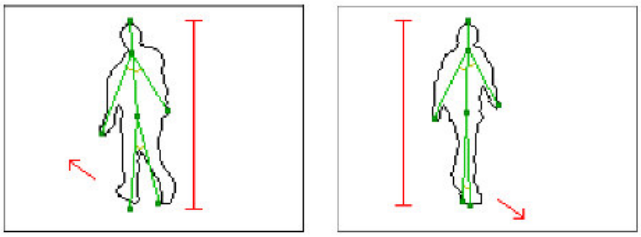


Figure 4: 2D star skeletons applied to different views of the same moment [6]

Using the motions of the skeleton, the values are analyzed by a left-right Hidden Markov Model (HMM) to determine sequences of activities and actions. An interesting result from their research is using a flat HMM instead of a hierarchical HMM to identify actions did not work well, but the flat HMM did detect activity segments well. A study using a similar method of skeleton overlay from only one camera viewpoint determined that it was possible to track the skeleton across multiple frames and required an estimation of angle velocity when joints became occluded by the rest of the body [89].

Another method of full body tracking is motion sequences and motion images [4, 54, 73]. Bobick and Davis [4] use motion energy images (MEI) and motion history images (MHI) to extract information about full body motions. A motion energy image is an image where the motion values as an action is performed are stored in a collective image over time. It is assumed that the person can be separated from the background during the making of these images (Figure 5).

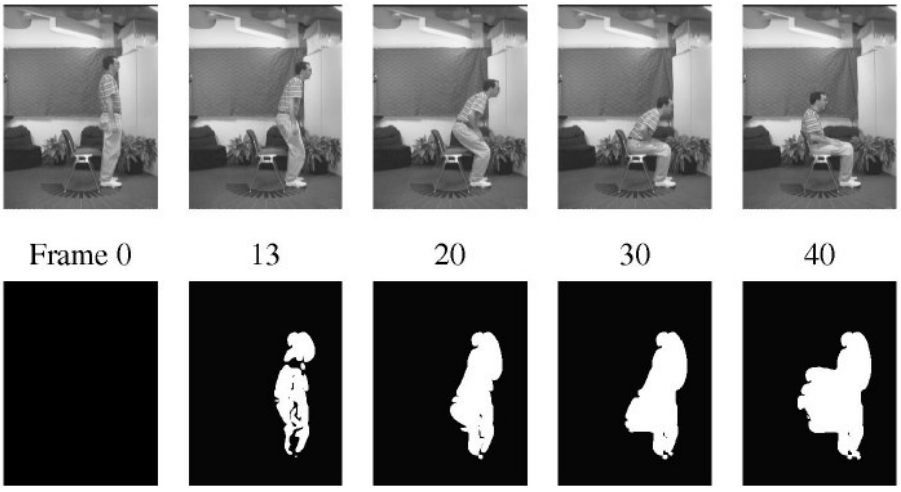


Figure 5: (top row) A series of images of "sitting", (bottom row) Corresponding MEI [4]

Though information reguarding the motion in that region of the image is retained, the pattern of the motion through time is lost. Since the MEI answers the question of "where" the motion is, the MHI was used to answer the question of "how" the motion moves. By simply adding a decay operator for the motion values over time, the MHI can represent when the motion occurred in the time sequence (see Figure 6).

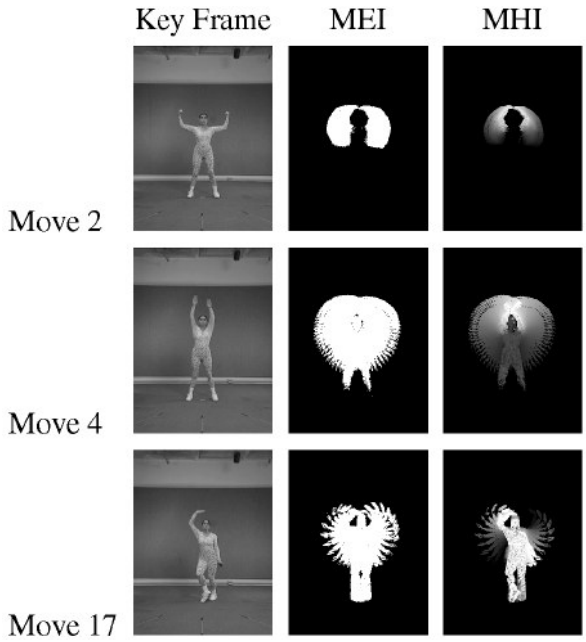


Figure 6: Motions and the corresponding MEI and MHI [4]

To compare the MEI and MHI images, the shapes portrayed by the captured motions are calculated. The Hu moments are calculated to give a shape feature vector for each of the images. Moments, when used with images, are weighted averages of the pixel intensities over a certain area of an image. Moments are defined in terms of Riemann Integrals.

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q \rho(x,y)\,dx\,dy \text{ , where p,q = 0,1,2,…} \quad (1)$$

$\rho(x,y)$ is the pixel intensity at position (x,y). From these moments, it is useful to derive central moments. Central moments are moments that are translation invariant (not dependant on the location of the area in the image). Central moments are defined as:

$$u_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x-\bar{x})^p (y-\bar{y})^q \rho(x,y)d(x-\bar{x})d(y-\bar{y}) \text{, where p,q = 0,1,2,…} \quad (2)$$

By subtracting the mean $\bar{x}$ and $\bar{y}$ values from each pixel comprising the shape of the object, the shape is normalized to a common location no matter what location it originated from in the image. For the purposes of general object detection via shape analysis, moments must be invariant to translation, rotation, scaling and mirror shapes. The equations used for the first 4 orders of Hu moments were calculated:

To achieve translation invariance:

$$\mu_{00} = m_{00} \equiv \mu \quad (3)$$
$$\mu_{10} = 0 \quad (4)$$
$$\mu_{01} = 0 \quad (5)$$
$$\mu_{20} = m_{20} - \mu\bar{x}^2 \quad (6)$$
$$\mu_{11} = m_{11} - \mu\bar{x}\bar{y} \quad (7)$$
$$\mu_{02} = m_{02} - \mu\bar{y}^2 \quad (8)$$
$$\mu_{30} = m_{30} - 3m_{20}\bar{x} + 2\mu\bar{x}^3 \quad (9)$$
$$\mu_{21} = m_{21} - m_{20}\bar{y} - 2m_{11}\bar{x} + 2\mu\bar{x}^2\bar{y} \quad (10)$$
$$\mu_{12} = m_{12} - m_{02}\bar{x} - 2m_{11}\bar{y} + 2\mu\bar{x}\bar{y}^2 \quad (11)$$
$$\mu_{03} = m_{03} - 3m_{02}\bar{y} + 2\mu\bar{y}^3 \quad (12)$$

To achieve scaling invariance:

$$\eta_{pq} = \frac{\mu_{pq}}{(\mu_{00})^{\gamma}} \text{ where } \gamma = (p+q)/2+1 \text{ and } p+q \geq 2 \quad (13)$$

To achieve orientation invariance the 7 Hu moments [76]:

$$v_1 = \eta_{20} + \eta_{02} \quad (14)$$
$$v_2 = (\eta_{20} + \eta_{02})^2 + 4\eta_{11}^2 \quad (15)$$
$$v_3 = (\eta_{30} + 3\eta_{12})^2 + (3\eta_{21} + \eta_{03}) \quad (16)$$
$$v_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \quad (17)$$
$$v_5 = (\eta_{30} + 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})] + 3(\eta_{21} + \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \quad (18)$$
$$v_6 = (\eta_{20} + \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \quad (19)$$
$$v_7 = 3(\eta_{21} + \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - (\eta_{30} + 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \quad (20)$$

To identify test movements, the Mahalanobis distance is calculated between the test feature vector and the known training feature vectors. This distance measure takes into account the distance of a sample from a group as well as the group's standard deviations. Mahalanobis distance is the distance of a sample from the center mass of a group divided by the width of the ellipsoid in the direction of the sample (Note: the ellipsoid is the best estimate model of the group samples). Since this method is sensitive to varying time spans over which the action is taking place, varying the values of the time coefficient can speed up or slow down any action to help synchronize the image with the training motions. The time coefficient is the constant that controls the speed of intensity decay for the images as frames pass. Using only a single camera, they test the recognition of the samples receiving a result of 12 correct classifications out of 18 trials. They extend the experiment by using 2 cameras that view the task at varying angles between 0-90 degrees to test for accurate detection of the task. The difference for the camera angles were known and the distance from the person had to be the same. This method improved the results to 15 correct classifications out of 18 trials. One of the problems noted for the classification across other subjects was the speed of the action was significantly slower than the original aerobics instructor in which the system was trained on and the conjunction of the subject wearing low frequency clothing made the segmentation algorithm not detect the complete motion.

Partial Body Analysis

The partial body analysis is focused on the body parts that give the most information about the task observed. When observing humans, the obvious parts to analyze would be the head/eyes and the hands for many upper-body tasks. If the main actions are running, jumping, squatting, etc., the best body parts to focus on may be the legs. Aloimonos et. al. used the MoCap System to monitor the angles of the knee, hip, and ankle joints for the right and left leg [20]. They go on to use a set of symbols to represent the 6 possible combinations of joint velocity and joint acceleration (R for

negative velocity and acceleration; B for positive velocity and acceleration, Y for negative velocity, positive acceleration etc). They also apply a positive integer value for the angular velocity with each symbol. By sequencing these symbols, actions are represented by a set of symbols as can be seen in Figure 7 below.

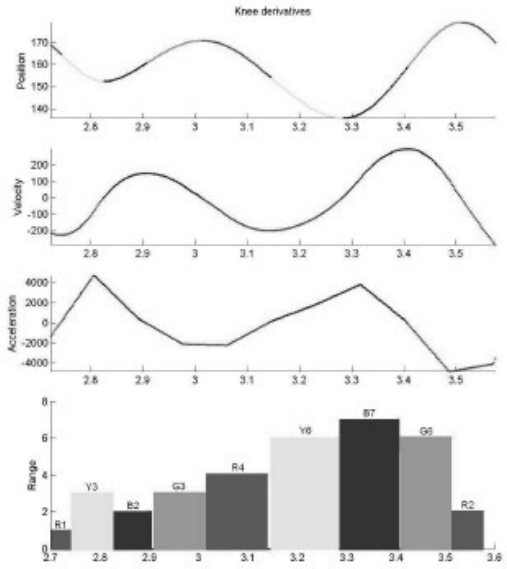


Figure 7: Knee angle derivatives during a jogging activity [20]

Research shows trends toward using the head to detect and identify tasks. Kojima et. al. [5] showed that tracking the head and estimation of the face direction could be used to identify directions and activities within the lab environment. Using a single stationary camera trained on the empty room as the background, the person is detected by differences with background. The head is determined by chromatically training on an average of the face and hair tones. An edge mask is used to filter the head region. The edge mapping of the head is Gaussian blurred and compared to a database of sample head directions. On the assumption that the layout of the room is known before hand, actions are determined by the proximity and directions of the head as well as the trajectory throughout time.

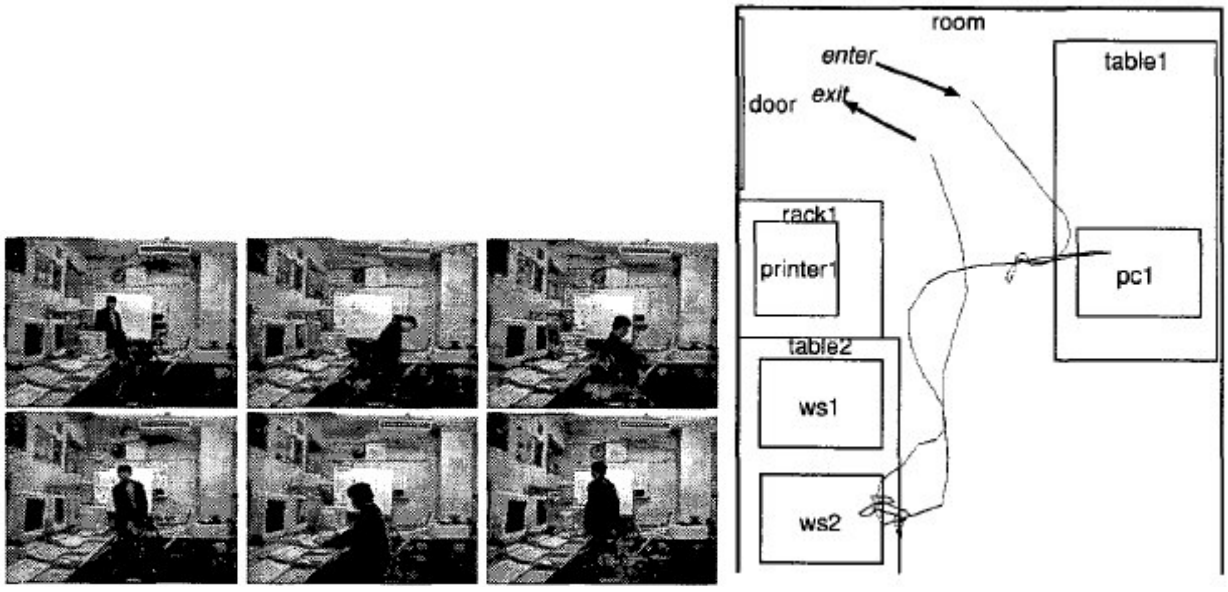


Figure 8: Sample input sequence of action in the room [5]

The sample image shows a person entering the room, moving and working on pc1 (personal computer 1) , moving and working on ws2 (workstation 2) and leaving the room. The video sequence is segmented by monitoring the head. If the head is moving, the head is facing in primarily the same direction. If the head is still, the head keeps approximately the same position. By taking the change in distance across a segment for the only moving agent (the human), and using it in the following sigmoid function,

$$f(x) = \frac{1}{1 + Ae^{-Bx}} \tag{21}$$

, where A and B are empirically selected constants, x is the change in distance. The f(x) values range from [0,1]. A number of features are extracted for each segment as shown in Table 1(a) below. A list of verbs are shown in Table 1(b) below.

Table 1(a) Conceptual Features

feature	range	meaning
features about motion		
time_elapse	real [0,1]	time elapsing or not
move	real [0,1]	object moves
move_near	real [0,1]	move toward object
move_apart	real [0,1]	move away from object
move_up	real [0,1]	move to up
move_down	real [0,1]	move to down
turn_toward	real [0,1]	turn to the direction of object
...	...	.....
features about state		
face	real [0,1]	face to object
by	real [0,1]	locate near object
exist	integer 0/1	existing
...	...	.....
features about characteristics		
prop_through	integer 0/1	able to be passed through
prop_operable	integer 0/1	able to be operated

Table 1(b) Verbs

verb	conceptual features
approach	$exist(a)^0$, $exist(a)^1$, $exist(o)^0$, $exist(o)^1$, $face(a, o)^0$, $by(a,o)^1$, $move\_near(a, o)$
go away	$exist(a)^0$, $exist(a)^1$, $exist(o)^0$, $exist(o)^1$, $by(a, o)^0$, $neg(by(a, o)^1)$, $move\_apart(a, o)$
stand up	$exist(a)^0$, $exist(a)^1$, $move\_up(a)$
sit down	$exist(a)^0$, $exist(a)^1$, $move\_down(a)$
enter	$neg(exist(a)^0)$, $exist(a)^1$, $by(a, o)^1$, $prop\_through(o)$
exit	$exist(a)^0$, $neg(exist(a)^1)$, $by(a, o)^0$, $prop\_through(o)$
operate	$exist(a)^0$, $exist(a)^1$, $time\_elapse(a)$, $face(a, o)^0$, $neg(move(a))$, $prop\_operable(o)$
...	.....

a: agent, o: object, $neg(p) \equiv 1 - p$

The verbs are represented by a set of feature changes from the beginning to the end of the segment. Using the approach verb as an example, the existence of the agent needs to transition from 0 to 1, the existence of the object needs to transition from 0 to 1, and the person transitions from not facing the object to being next to the object. By connecting the agent, object, verbs and, time, a higher level language is created to describe the activity in the room.

Madabhushi & Aggarwal showed that, by tracking the centroid of the head alone, actions such as standing, walking, sitting, etc. could be recognized [3]. A single person would be in the video performing these actions. Of 41 sequences, 34 were identified correctly giving an 84% success rate. The main point to note in this study is the simplicity of the methods used. The head was determined by the upper portion of the motion in the image and tracked by taking the nearest group in the sequence of images. Using the velocity of the head over successive frames as the feature vector, they were able to get the 84% success rate for action recognition. All the studies agree that focusing strictly on the head limited the actions available to be detected. Both studies used relatively simple methods for tracking the head and simple feature vectors to describe the actions of the videos.

Other research focused attention on the hands such as Kuniyoshi & Inoue using a multi-camera system to track the hand and objects for recognition of action sequences [16]. Their setup used stereo vision to observe a human performing a building task. They make some key assumptions such as the blocks not being occluded, a single hand is used at all times, and the action is carried out in a smooth consistent manner with no mistakes in assembly process. The viewing area contains only the necessary objects and is intruded upon only by the hand of the demonstrator. The blocks must be picked up by the hand in a pincer configuration with the forefinger and thumb making connection on the sides of the block without occluding the block from the view of the cameras. By tracking the hand and the objects, they created assembly classes of transfer, local motion, approach, depart, and fine motion. To identify these classes a set of sequences were observed (i.e., transfer consisted of empty hand reaching object (near, hold-false), reaching with object (near, hold-true), withdraw hand from object (other)). Using this information, they could recognize assembly tasks in real time. They continued their work for task mimicking using a robotic manipulator [17].

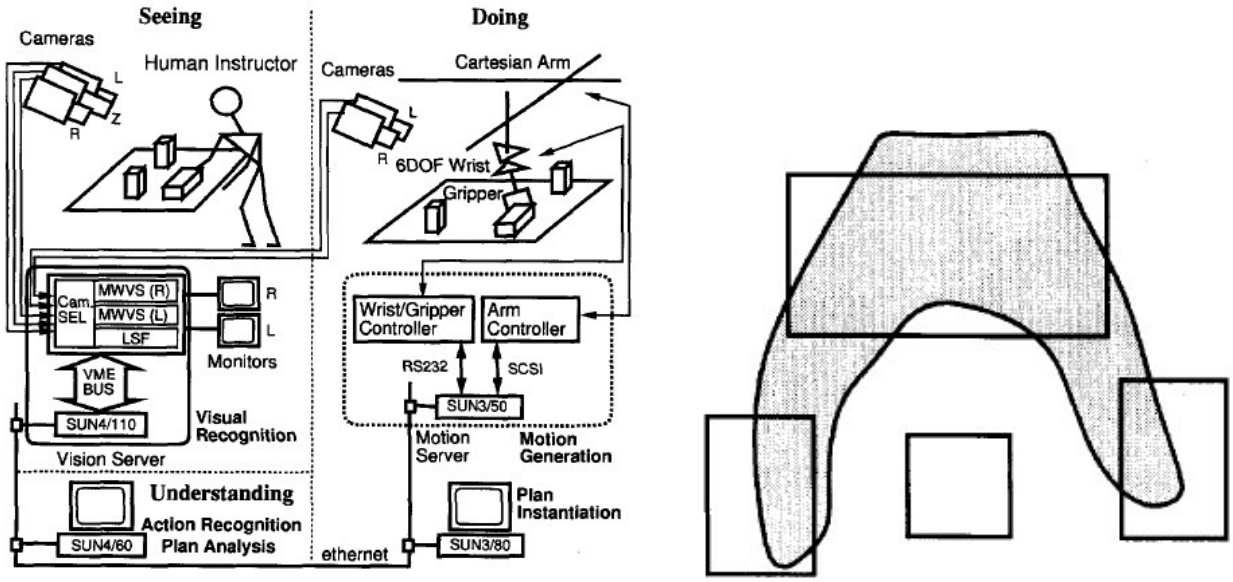


Figure 9: (left) Overview of system; (right) Hand position for picking up object w.r.t. camera [17]

Rao et. al. used hand tracking to extract the two dimensional motion of the hand when doing a task to extract significant moments (or dynamic instances), such as pauses and direction changes, to

identify whole tasks such as opening a cabinet, picking up and putting down objects, etc [10]. They determine dynamic instances by recording the hand position and calculating the curvature and direction via equation 22:

$$\kappa(t) = \|\mathbf{r}'(t) - \mathbf{r}''(t)\| / \|\mathbf{r}'(t)\|^3 \quad (22)$$

where $\mathbf{r}(t) = [x(t), y(t), t]$, x(t) is the horizontal coordinate of the hand centroid, y(t) is the vertical coordinate of the hand centroid, and t is time. By assigning a value of "+" for clock-wise and "-" for counter clock-wise, they correlate the number of peaks in the curvature data (dynamic instances) and the sign direction of the curves to create a feature vector for the movement (an example can be seen in Figure [10].

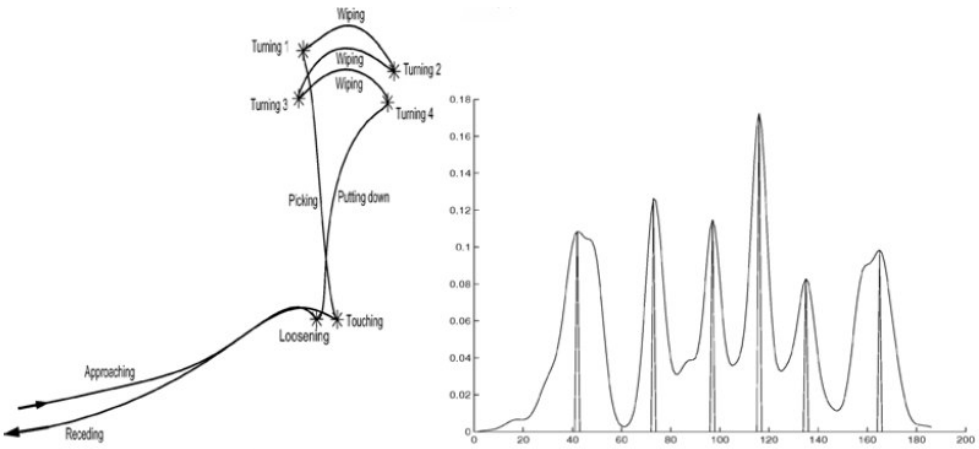


Figure 10: (left) Hand Movement of Erasing Board, (right) Curvature Plot of Movement [10]

Data comparison occurs if to sets of actions have the same number of dynamic instances and the same sequence of signs for those instances. The feature vectors for the dynamic instances are the 2D coordinates of the hand at those moments. They use an affine projection model to project the 2D motion into a 3D space. The affine model assumes that the depth of the 3D action is small compared to the distance to the camera. This assumption allows for 2D points to be changed to 3D through linear transformation. A 3D action observed by a camera will be projected onto a 2D image plane as can be exemplified by Figure 11.

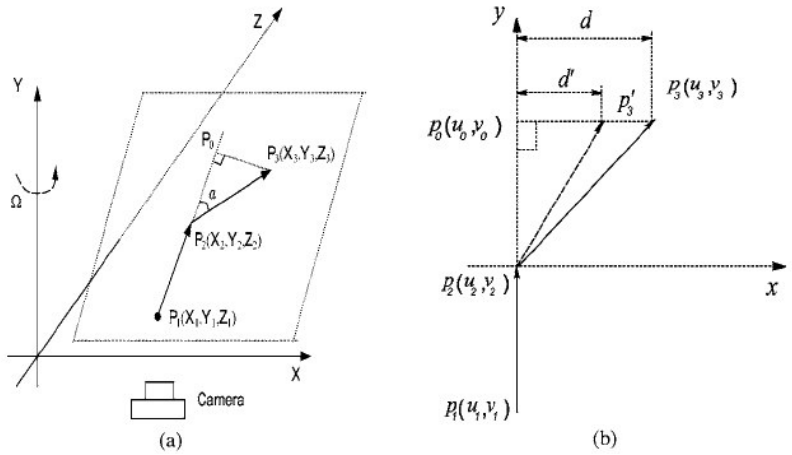


Figure 11: (a) Sample vector representation in 3D, (b) 2D projection of 3D vector [10]

In Figure 11, there are 3 points P_1, P_2, and P_3. The camera faces orthogonal to the x-y plane in Figure 11a. Assuming the vector $\overrightarrow{P_1P_2}$ is vertical (parallel to the y-axis), no matter how the camera pans around the motion vectors with respect to the y-axis the vector $\overrightarrow{P_1P_2}$ will remain unchanged in the 2D projection $\overrightarrow{p_1p_2}$ of Figure 10b. The limitation lies with vector $\overrightarrow{P_2P_3}$. In Figure 11b, as the camera pans around the action with respect to the y-axis, the 2D projection changes from the solid vector $\overrightarrow{p_2p_3}$ to the dashed vector $\overrightarrow{p_2p_3}$. Notice the overall direction change from point $p_1(u_1,v_1)$ to $p_2(u_2,v_2)$ to $p_3(u_3,v_3)$ remains clockwise even as the angle of pan (Ω) about the y-axis changes as long as the angle of pan (Ω) remains within $(-90^o,90^o)$ of the position where $\overrightarrow{p_2p_3}$ lies on the x-y plane. The following equation accounts for the change in the x coordinates for the 3D point as the camera pans.

$X' = X\cos(\Omega) - Z\cos(\Omega)$, where X' changed x-coordinate as the camera pans

Given the affine camera model used is:

$$u' = f\frac{X'}{D} \tag{23}$$

,where f is the focal length, D is the distance from the camera to P_2. The d' value (i.e. the distance between the projections of P_3 and P_0) space shown in Figure 11b is calculated by the following:

$$d' = f \frac{(X_3 - X_0)\cos(\Omega)}{D} \quad (24)$$

The same logic follows for the tilt angle around the x-axis (φ). By determining the curvature and dynamic instants of multiple camera angles of the same task, the 2D coordinates (matrix M) can be converted to 3D coordinates (matrix S) if the projection matrix (P) is known. The equation for the 3D to 2D conversion is a linear transformation:

$$M = P * S \quad (25)$$

,where M is a 2 x n matrix consisting of n 2D points of an action, S is the 3 x n shape matrix consisting of n 3D points of an action, and P is the 2 x 3 projection matrix.

Each action is viewed from k camera angles, thus each camera view has an M matrix creating k M matrices. Using the distances between the 3D calculation instances calculated from the different camera angles, they identify the motions and new actions (if the distances are far from all trained motion). The experiment used 47 different actions performed by 7 different people. They determined effectiveness by judging the best 3 choices for each action. Of all the actions, only 5 had 3 false matches and another 5 had partially incorrect matches. The comparisons were based on a single instance of the action.

Some of the best methods for segmentation and recognition of human tasks lie in observing the head movements/gaze in conjunction with hand motions such as in the research of Yu & Ballard [18] where by simply studying the head motion, eye motion and hand activity led to segmentation and recognition of the tasks. A system layout showing the motions of the head, eye, and hand extracted from an image sequence to train a set of Hidden Markov Models for action recognition (see Figure 12).

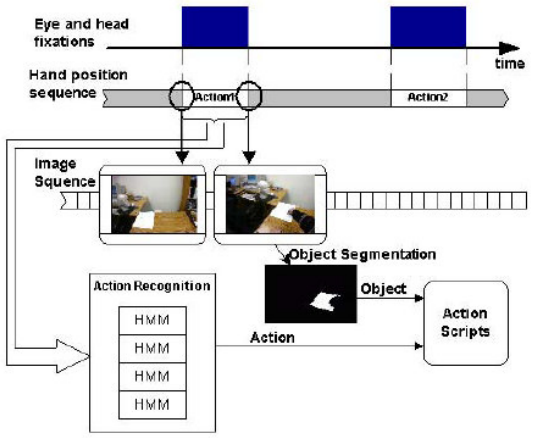


Figure 12: Overall system diagram [18]

The motions were gathered using an invasive hand sensor and headset that tracks the head/eye motion. From the head motions throughout a task, they were able to extract when the head was fixated by low motion values as seen in Figure 13. A similar threshold method was used to determine eye fixation from low eye motion values also seen in Figure 13. After determining the moments where the head and eye were fixated at the same time, action segments were extracted during these time periods. The data from the actions were the 3D hand position and rotation for both hands during the times of simultaneous fixation.

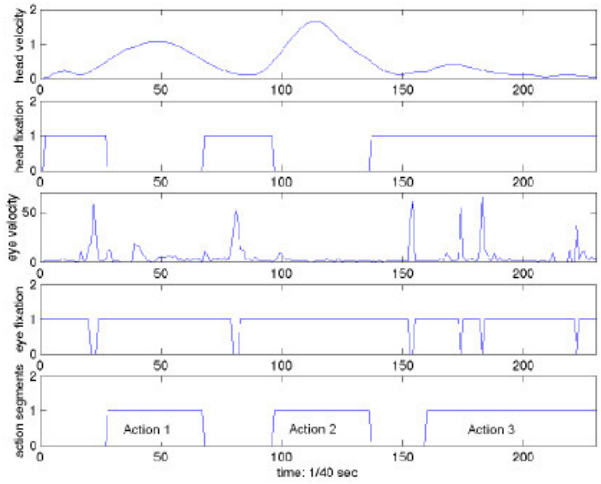


Figure 13: Sample of action segmentation via head/eye and hand movements [18]

By training a Hidden Markov Model on the data gathered through the fixation periods, an overall segmentation accuracy of 83.9% was achieved with 91.6% recognition accuracy on actions such as picking, placing, lining up, stapling and folding. The field of machine vision is continuing to examine more robust methods of detecting and recognizing actions.

CHAPTER III

EVENT TRACKING

Some of the most popular methods of studying human motion segmentation use finite state automata, Hidden Markov Models (HMM), Bayesian Networks and Kalman filters. The main use of these techniques in human motion segmentation is to provide recognition for actions performed. In general, these techniques receive the information from a system (in the form of vectors) and form states depending on how these vectors correlate with each other. By following a sequence of states, an action can be described. After training the system, a new sequence of vectors can be read in and interpreted by the system. Depending on the resulting estimated sequence of states, the new sequence of vectors is identified as a particular action. It is essential to understand more about these methods so that they can be applied in the most effective manner. The definition of the Hidden Markov Model will be referenced from a compilation of sources [30, 94-100]. After defining the models, methods of use for human motion segmentation and recognition will be covered.

Finite State Machines

Finite State Machines (FSM) are models of systems that contain states and transitions that mathematically represent the activity [102, 103]. Though finite state machines are most commonly used to represent system operations, human motion analysis can use them to represent the series of actions and identify tasks. The parts of a finite state machine are as follows:

States: S_i :where i = 1 to N (Number of States)

Transitions: T_{ij} :where T is the connectivity from state i to state j

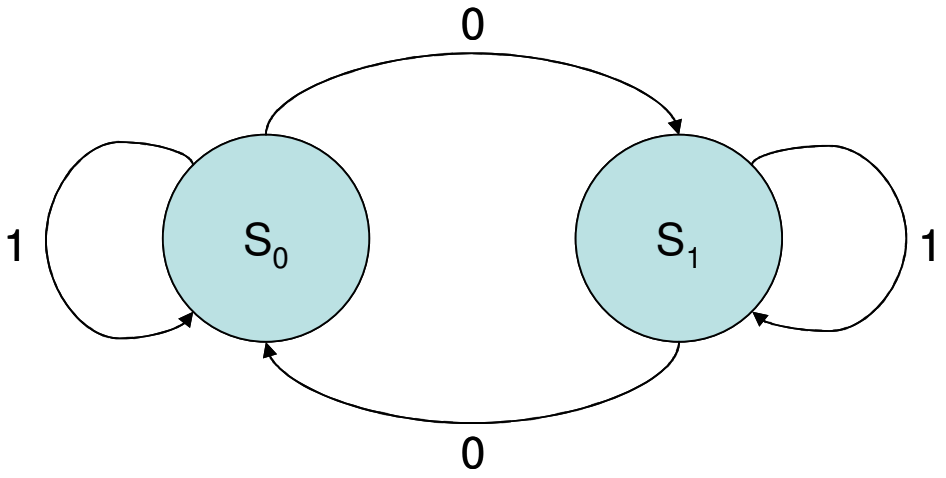
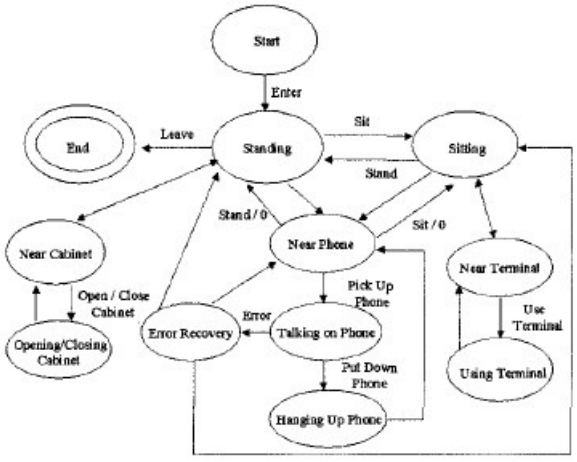


Figure 14: Example FSM

In the example FSM shown in Figure14 , it can be seen that starting in state 0 if a 0 input is received, the system transitions to the other state. If the system receives an input 1, then it remains in the current state. Actions represent the behavior of the system throughout the model. Exiting Actions are actions that occur when the system leaves a particular state. Entering Actions are actions that occur when a system is entering a state. Of course, finite state machines can be vastly more complicated than that shown in Figure 14 depending on the task to be modeled. In the papers using finite state machines, the models are used to represent the state and activities of a person.

In Ayers and Shah's work [78], they use a finite state machine to represent the location and activity of a person in a particular room. By simply detecting the skin tones of a person and the objects in a room, they are able to create a state machine to represent the actions in the room. Using the state machine shown in Figure 14, the student's activities are identified in the lab. The system is trained on the original positions of the objects and skin tones.



Actions	Recognition Conditions
Enter	significant skin region detected for τ frames
Stand or Sit	y-position of a person's tracking box increases or decreases significantly
Pick Up Object	the object initial location has changed after the person and object have moved away
Put Down Object	person moves away from object being tracked
Open or Close	increase or decrease in the amount of "scene change" detected in object
Use Terminal	person sitting near terminal and "scene change" is detected in mouse, but not behind it
Pick Up Phone	"scene change" in phone detected, phone has been tracked until it stopped moving in the y-direction
Put Down Phone	phone has been tracked back to its original location
Leave	person's tracking box is close to edge of the entrance area through which people enter

Figure 15: (left) Action Finite State Machine, (right) Transition descriptions [78]

The system is able to identify actions of different people. Actions are determined by tracking the position of the person in correlation with the position of key objects (e.g., "Pick Up Object" is determined by the proximity of the person to the object and the change in the objects position). All of the transitions are explained in Figure 15. The activities of the person can be described from the states of the FSM. A key limitation to the model is the amount of work necessary to change the finite the state machine to detect a wider range of actions.

Hidden Markov Models

To define HMMs, it is easier to first define Markov chains [95]. Markov chains are models that consist of states and transitions across those states. Markov chains are very similar to weighted finite state machines but the difference being they have a set of observations and a transition matrix that are used to determine the probability of that set occurring given a specific model. Let's define the following:

States: S_i :where i = 1 – N (Number of States)

Observations: $O = \{V_1, V_2, \ldots, V_T\}$:where T is the number of observations in a sequence

Transition Matrix: A :where each value a_{ij} is the probability of transitioning from S_i to S_j

Initial Probability: π_i :where each value is a probability of starting in state S_i

In Rabiner's paper [30], he uses a sample 5 state Markov chain to aid in the description (Figure 16). For the transition matrix to be correct an assumption must be made called the Markov assumption.

Markov Assumption: $$P(s_i|s_1 \ldots s_{i-1}) = P(s_i|s_{i-1}) \quad (26)$$

The Markov assumption says that the probability of transitioning from S_{i-1} to S_i is only dependant on the previous state S_{i-1} and not any of the states before S_{i-1}. This is saying that the transition probabilities do not change due to transitioning from states prior to the current state.

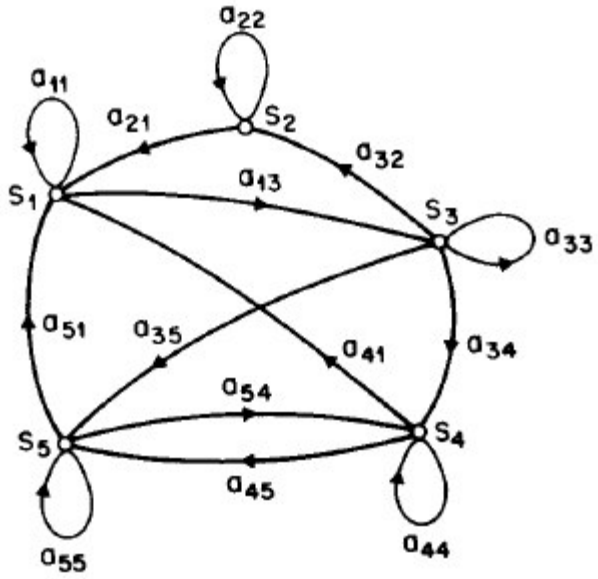


Figure 16: Sample 5 state Markov chain [30]

In a Markov chain, all the information from the states, transition matrix, initial probabilities, and observations are provided. The Markov chain can be given a sequence of observations and the probability of that sequence occurring may be calculated from the transition matrix and the initial probability. The observations V directly correspond to the state S. Notice that observations for a Markov chain are now defined as states. This definition is accurate because all the information about

the model is available so an observation actually indicates the state that the system is currently in. For the model above, if O = { V_5, V_4, V_1, V_3, V_2, V_2, V_1 } corresponds to O = { S_5, S_4, S_1, S_3, S_2, S_2, S_1 }, then $P(O|model) = \pi_5 \times a_{54} \times a_{41} \times a_{13} \times a_{32} \times a_{22} \times a_{21}$. The Markov chain is useful for finding probabilities of sequences but relies on the unlikely situation where all information about the model is available. Addressing this shortcoming is where the strength of the Hidden Markov Model lies.

The Hidden Markov Model has the same variables as the Markov chain. The only difference is the observations $O = \{O_1, O_2, \ldots, O_T\}$ where T is the number of observations in a sequence. Each O_t is drawn from a list of symbols $V = \{v_1, v_2, \ldots, v_W\}$ where W is the number of different symbols that are available to the model. The reason the observation definition change is because the model is now "hidden". Since the observations no longer give exactly which state the system resides in, one additional component is needed:

Emission Matrix: B : where each value b_{iw} is the $P(v_w| S_i)$

With the emission matrix, the symbols from observations have a probabalistic link to the states of the hidden model. The assumption that the observation only relies on the current state and not previous states must be made for the Emission matrix to hold true. Given appropriate values of N (number of states), W (number of symbols), A (transition matrix), and B (emission matrix), the Hidden Markov Model can produce a sequence of observations. The HMM requires that N and W be set so the model can be represented as λ (A, B, π). Eisner [100] explained a sample problem of a HMM would be: "You are climatologists in the year 2799, studying the history of global warming. You can't find any records of Baltimore weather, but you do find my diary, in which I assiduously recorded how much ice cream I ate each day. What can you tell about the weather from this information?"

Given:

V = {1, 2, 3} for the number of ice cream eaten that day

S = {Cold, Hot} for the weather

$$B = \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.1 & 0.2 & 0.7 \end{bmatrix}$$ where the rows are the states and the columns are the symbols.

(i.e., $b_{23} = 0.7$ which is the probability of 3 eaten ice creams on hot days)

$$A = \begin{bmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{bmatrix}$$

The state estimation for the following plotted observation sequence would be:

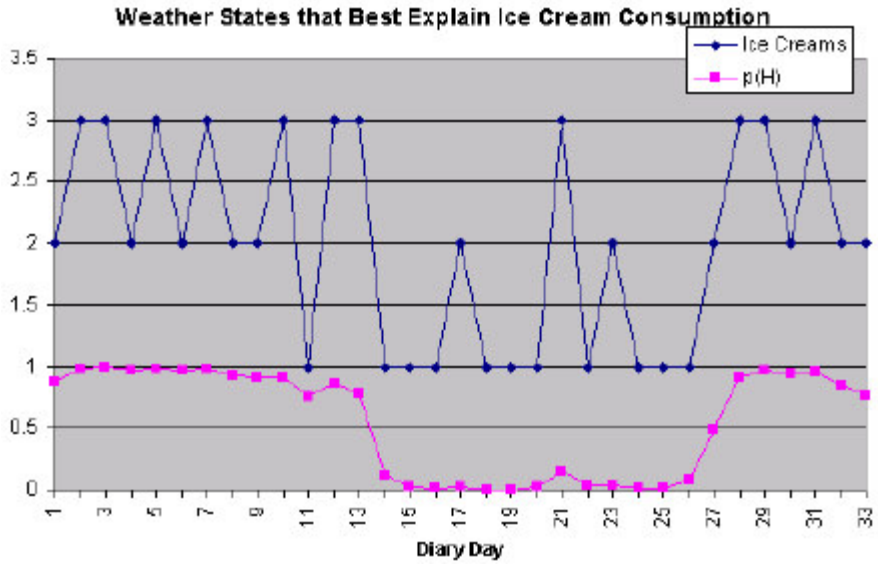


Figure 17: Results of the Sample Weather Problem [100]

There are 3 main problems to be solved with HMMs:

- Given an O and λ, what is the probability of observing the given observation sequence?
- Given an O and λ, what is best estimate of the corresponding state sequence?
- Given an O and S, how can we change λ to maximize P(O|λ)?

The first problem is a likelihood problem. By solving this problem, a method is developed that allows for comparisons among models. This is useful for comparing unclassified observations with known model estimates to classify the observation sequence. This problem is solved by taking the sum of the probabilities of all possible states sequences that can represent the given observation

sequence (a solution provided by the forward algorithm [30, 94, 96]). The general steps of the algorithm are as follows:

$\alpha_{t-1}(i)$ the **previous forward path probability** from the previous time step

$b_j(o_t)$ the **state observation likelihood** of the observation symbol *ot* given the current state *j*

1. Initialization

$$\alpha_1(j) = a_{0j} b_j(o_1) \quad 1 \le j \le N \tag{27}$$

2. Recursion:

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t) \quad 1 \le j \le N, 1 \le t \le T \tag{28}$$

3. Termination:

$$P(O|\lambda) = \alpha_T(s_F) = \sum_{i=1}^{N} \alpha_T(i) a_{iF} \tag{29}$$

The second problem is a decoding problem. By solving this problem, a way of analyzing the structure of the model is available. This may be used to learn about optimal state sequences for recognition. This problem is solved by a very similar algorithm to the forward algorithm called the Viterbi algorithm [30, 94, 96]. The Viterbi algorithm takes the maximum probability for all the possible state sequences that represent the given observation sequence to determine the sequence of states. The most probable state sequence can be determined by tracking the states that create the maximum probability. The general steps of the Viterbi algorithm are:

$v_{t-1}(i)$ the **previous Viterbi path probability** from the previous time step

$b_j(o_t)$ the **state observation likelihood** of the observation symbol o_t given the current state *j*

1. Initialization:

$$v_1(j) = a_{0j} b_j(o_1) \; 1 \le j \le N \tag{30}$$

$$b_{t1}(j) = 0 \tag{31}$$

2. Recursion:

$$v_t(j) = \underset{i=1}{\overset{N}{MAX}}\, v_{t-1}(i)a_{ij}b_j(o_t) \quad 1 \le j \le N,\ 1 < t \le T \tag{32}$$

$$bt_t(j) = \underset{i=1}{\overset{N}{ARGMAX}}\, v_{t-1}(i)a_{ij}b_j(o_t) \quad 1 \le j \le N,\ 1 < t \le T \tag{33}$$

3. Termination:

The best score:
$$P^* = v_t(qF) = \underset{i=1}{\overset{N}{MAX}}\, v_T(i) * a_{i,F} \tag{34}$$

The start of backtrace:
$$S_T^* = bt_T(qF) = \underset{i=1}{\overset{N}{ARGMAX}}\, v_T(i) * a_{i,F} \tag{35}$$

The third problem is a training problem. By solving this problem, a way of training a HMM to a set of observation sequences is available. This problem is solved by an algorithm called the Baum-Welch algorithm [30, 94, 96] which is a variation of the expectation maximization algorithm [29]. The algorithm starts with an initial estimate of the model. The algorithm calculates γ (the probability of being in a state S_j at time t) and ξ (the probability of making the transition from S_i to S_j and observing O(t+1)), then uses these values to adjust the model. The cycle continues until convergence (there is a chance that the algorithm will converge to a local minimum instead of a global minimum. The algorithm is given below:

4. Initialization

$$\beta_1(i) = a_{i,F}, \quad 1 \le i \le N \tag{36}$$

5. Recursion:

$$\beta_t(j) = \sum_{i=1}^{N} \alpha_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \quad 1 \le j \le N, 1 \le t \le T \tag{37}$$

6. Termination:

$$P(O|\lambda) = \alpha_T(s_F) = \beta_1(0) = \sum_{i=1}^{N} \alpha_{0j} b_j(o_1) \beta_1(j) \tag{38}$$

The next topic of interest is observing how HMMs are used in human motion analysis. Lee and Kim [93] tracked gestures of the hand and further refined the HMM's recognition capabilities by adding a likelihood threshold for the actions. They gathered information about the hand using a feature vector that consists of the horizontal and vertical movement between frames. These feature vectors are given a directional code from 0-15 (Figure 18).

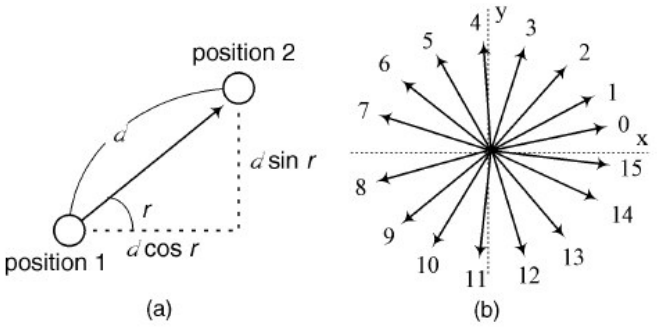


Figure 18: (a) Feature extraction, (b) Directional code values [93]

Using these directional codes as observation values, they set the number of states (N) of the HMM to be 5-10 depending on the complexity of the motion. By training a HMM for each gesture and a HMM for the thresholds, the gesture end points can identified. The threshold HMM is a combination of all the states from each of the separate gesture HMMs. The output observation probabilities and the self-transition probabilities are the same as the gesture HMMs but the outgoing transition probabilities are weakened by the following equation:

$$a_{ij} = \frac{1 - a_{ij}}{N - 1}, \text{ for all j, i}\neq\text{j where N is the number of states} \tag{39}$$

All the states in the threshold HMM are interconnected giving the threshold model the ability to link subpatterns of different gestures. Keeping the output observation and self-transition probabilities the

same will allow the threshold HMM to have a higher likelihood during subpatterns. The reduced output transition probabilities force the threshold HMM to have a smaller likelihood after a complete sequence of a gesture than the specific gesture-trained HMM. By examining where the gesture-trained HMM overtakes the threshold HMM, a candidate endpoint emerges (Figure 19).

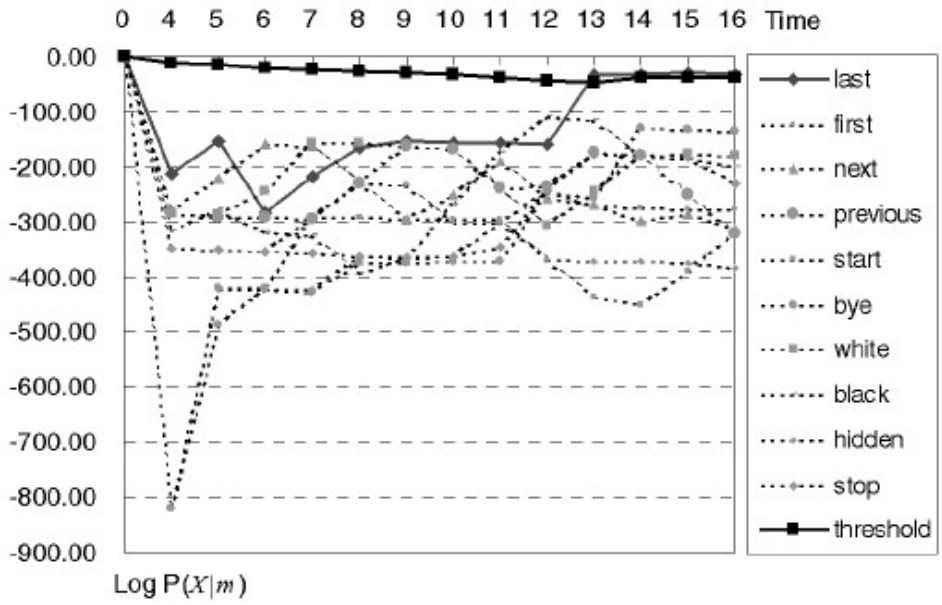


Figure 19: Likelihood plot of gestures and threshold vs time [93]

In Figure 19, it can be seen that possible candidate endpoints emerge for the gesture "last" at time 13-16. Each candidate endpoint can be used as a reference point to run the Viterbi algorithm to backtrace and find the most likely starting point in the sequence. By taking the sequences with the highest likelihood with the gesture-trained HMM, they are able to segment continuous human motion. The system had an overall detection rate of 93.81% as shown in Table 2.

Table 2: Detection Results with Reduced Threshold Model [93]

Command	Number of Gestures	Results					
		Insert	Delete	Substitute	Correct	Detection (%)	Reliability (%)
Last	41	1	1	2	38	92.68	90.68
First	48	0	3	2	43	89.58	89.58
Next	57	1	0	1	56	98.25	96.55
Previous	41	1	4	1	36	87.80	85.71
Start	28	0	3	0	25	89.29	89.29
Bye	35	0	2	1	32	91.43	91.43
White	45	0	0	0	45	100.00	100.00
Black	49	0	3	1	45	91.84	91.84
Hidden	39	0	1	1	37	94.87	94.87
Stop	37	0	0	0	37	100.00	100.00
Total	**420**	**3**	**17**	**9**	**394**	**93.81**	**93.14**

Deng and Tsui [42] follow up Lee and Kim's work with another way to identify gestures from continuous motion containing gesture and non-gesture motions. They argue that the Viterbi algorithm only works if the observation sequence consists only of the start and stop of a specific gesture (pencil trajectory of writing a number). In continuous motion, the non-gesture motion will cause the Viterbi algorithm to miss gestures because of the extra observations of the non-gestures. A solution for this is to have a sliding window that takes each observation and sets it as the beginning of a sequence to determine the gestures. If given an input sequence, $O^t = \{o_1, o_2, \ldots, o_t\}$ and the sequence is broken up into multiple sequences $O^t_\tau=\{o_\tau,o_{\tau+1}, \ldots,o_t\}$, the probability for an observation section given the gesture model is

$$P(O_1^t \cup O_2^t \cup ... \cup O_{t-1}^t \mid \lambda_g) = \sum_{\tau}^{t-1} P(O_\tau^t \mid \lambda_g) \tag{40}$$

where λ_g is the predefined gesture HMM model and t is the time step at which the gesture ends. The evaluation of this equation is the same as the sliding evaluation. The forward algorithm calculates this

value (equation 40). An alternative to equation 40 is to use the probability of an observation O and a starting state I (equation 41)

$$\phi_t^g(i) = \sum_{\tau}^{t-1} P(S_t = j\ , O_\tau^t \mid \lambda_g) \tag{41}$$

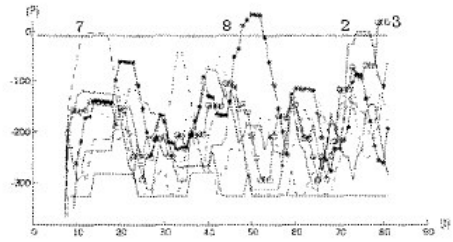
which can be implemented with less computational cost than equation 40. Since the likelihood values differ greatly across models, they take the likelihood from a training observation sequence of the model and normalize all the values with $\bar{k}_g$ of the test observations by that likelihood. Using equation 42 to develop $\bar{k}_g$,

$$\bar{k}_g = 1/\bar{P}(O_g \mid \lambda_g) \tag{42}$$

where $\bar{P}(O_g \mid \lambda_g)$ is the average observation probability given λ_g of the known observations from model *g*. On the assumption that all the models are well-trained, they assume that any observation g probability will be much higher in model *g* than for other models. After determining the gesture model with the highest likelihood for the observation t, they backtrack from the end observation $O_{\tau-1}^t$, using some threshold, over the states that maximized the likelihood and observe the statistical data to estimate the beginning state/observation of the number trajectory. They do not mention the exact method of determining the initial state, but they do mention that they calculate the maximized likelihoods from $O_{\tau-1}^t$ to the first observation in the sequence and determine a cutoff to the action based on the likelihood.

The data is gathered in the same manner as mentioned before by taking the motion difference of the hand between frames and assigning a directional code (see Figure 18). Using this method of detection, HMMs are trained for the handwritten numbers 0-3 and 6-9 (Figure 20).

a). Segmentation result for input trajectories. Each define gesture is marked by a dot at the beginning and a square at the end.



b). Likelihood evaluation by our method (for 783).

Figure 20: Example of hand written test data and Result of algorithm [42]

The results of the algorithm significantly outperform the Viterbi algorithm in Table 3.

Table 3: Deng and Tsui Results [42]

	N	I	D	S	C	Detection	Reliability
Viterbi	200	35	9	70	121	60.5%	51.5%
Our Method	200	3	7	4	189	94.5%	93.1%

Detection ratio = C/N. Reliability = C/(N+I).

Siskind and Morris use the standard HMM algorithms to detect hand – object interactions in video sequences [68]. Their experiment starts with labeling a set of 6 actions Pick-Up, Put-Down, Push, Pull, Drop, and Throw. Using the Sun Video system which consists of a single uncalibrated camera of 320 x 240 resolution at 30 frames per second, they recorded 72 videos of the aforementioned actions. First, the frames are segmented by color and motion. Then, elliptical models of the hand and object are extracted from the image sequence (Figure 21).

Figure 21: The processing stages of Siskind and Morris's tracker. (a) shows an input image. (b) shows the coloured pixels. (c) shows the output of the region grower on (b). (d) shows the moving pixels. (e) shows the output of the region grower on (d). (f) shows the combination of (c) and (e). (g) shows the ellipses that are fit to regions from (f) [68]

The elliptical models are created by taking the mean and covariance matrix of the (x,y) coordinates of the pixels in an area. Some problems encountered from this method are the creation of non-essential pixels and differences in the number of ellipses from frame to frame. To remedy the problems, they link ellipses in a ellipse chain that tracks the ellipses by analyzing the difference of their motion parameters across frames. By finding ellipse chains that are tracking the same object, the number of spurious ellipses is reduced by merging with the chains creating a constant number of total ellipse chains. For each ellipse, the following feature vector is extracted:

Absolute features

- The magnitude of the velocity vector of the centre of each ellipse
- The orientation of the velocity vector of the centre of each ellipse
- The angular velocity of each ellipse
- The first derivative of the area of each ellipse
- The first derivative of the eccentricity of each ellipse
- The first derivative of each of the above 5 features

Relative features

- The distance between the centers of every pair of ellipses
- The orientation of the vector between the centers of every pair of ellipses
- The difference between the orientations of the major axes of every pair of ellipses

- For every pair of ellipses, the difference between the orientation of the major axis of the first ellipse and the orientation of a vector from the centre of that ellipse to the centre of the second ellipse
- The first derivatives of each of the above four features

Of the entire 72 hand labeled video set, the HMM models were trained on 36 videos and tested on the other 36 videos. The models were able to recognize 35 of the 36 testing data. The misclassification that occurred was a "drop" event being classified as a "throw" event.

Kalman Filters

Kalman filters are one of the best solutions for tracking and data prediction [104,105]. Research has been done using Kalman filters in the tracking of trajectories [80]. In all real world applications, there is noise associated with sensor data (e.g., odometry in robots) or moments at which segments data is unavailable (e.g., occlusions of motion). One use of the Kalman filter is to estimate the necessary information from these moments. The Kalman filter uses a recursive algorithm that takes inital estimates of the system, creates a measure of how to change the system coefficients (Kalman gain), uses the Kalman gain along with the observed data (z) to create a new update estimate ($\hat{x}_k$) and error covariance (P_k), and finally the state transition matrix (Φ) is used to calculate the next $\hat{x}_{k+1}$ and P_{k+1} (see Figure 22)

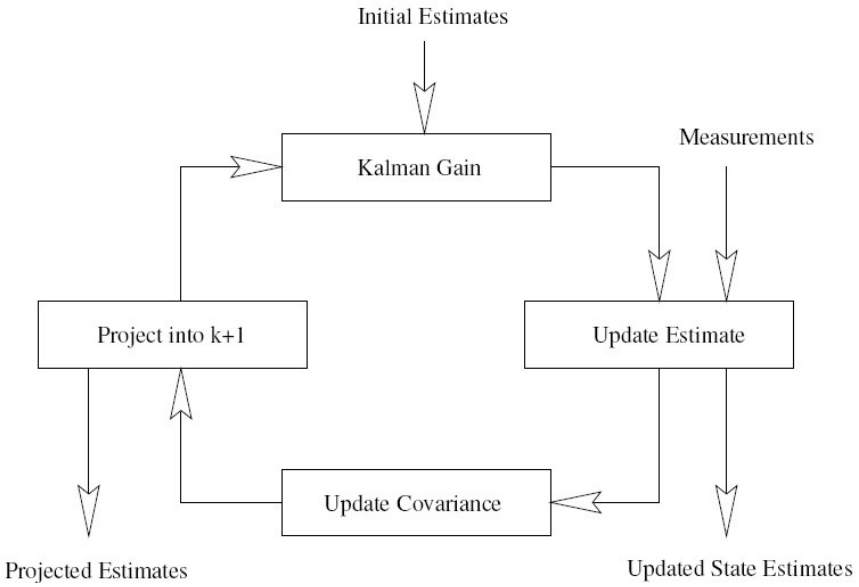


Figure 22: Kalman Filter Recursive Algorithm [104]

Let's look closely at the system that is being estimated.

State Equation:

$$\underline{x}_{k+1} = \Phi \underline{x}_k + w_k \tag{43}$$

where Φ is an n x n state transition matrix, $\underline{x}_k$ is an n x 1 state vector at time k, and w_k is and n x 1 driving, input disturbance

Measurement Equation:

$$\underline{z}_k = H \underline{x}_k + v_k \tag{44}$$

where $\underline{z}_k$ is a m x 1 measurement vector, H is a m x n connection between the state and the measurements, and v_k is m x 1 measurement noise

It is assumed that noise measurements v_k and w_k are white noise with known covariance. From observing the measurements z and using known information about the noise, H, and Φ, the state vector x and its covariance matrix (P) are estimated by using the 5 update equations:

Kalman Gain: $$K_k = P_k^{'} H^T (H P_k^{'} H^T + R)^{-1} \tag{45}$$

Update Estimate: $$\hat{x}_k = \hat{x}_k^{'} + K_k (z_k - H \hat{x}_k^{'}) \tag{46}$$

Update Covariance: $$P_k = (I - K_k H)P_k^{'} \quad (47)$$

Project into k+1: $$\hat{x}_{k+1}^{'} = \Phi \hat{x}_k \quad (48)$$

$$P_{k+1} = \Phi P_k \Phi^T + Q \quad (49)$$

The Kalman filter is a very good way to get optimal parameters or state estimation if the problem can be cast as linear state equations. Extended Kalman filters take a similar application to non-linear state equations. This is done by linearizing the state equation at small intervals of time k to model the non-linearity.

Cuevas et. al. study the Kalman filter and the extended Kalman filter to show the tracking abilities during times of occlusion [106]. In their study, they track the motions of a soccer ball and create occlusions in the data to observe the results of the filters. The state and measurement equations used to model the trajectory of the ball are:

State Equation:

$$x_{k+1} = F_{k+1,k} x_k + w_k \quad (50)$$

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \Delta x_{k+1} \\ \Delta y_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ \Delta x_k \\ \Delta y_k \end{bmatrix} + w_k \quad (51)$$

Measurement Equations:

$$y_k = H_k x_k + v_k \quad (52)$$

$$\begin{bmatrix} xm_k \\ ym_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ \Delta x_k \\ \Delta y_k \end{bmatrix} + v_k \quad (53)$$

The results from applying the Kalman filter over a trajectory are shown in Figure 23. In Figure 23, the left graph shows the trajectory of the ball and the estimated position from the Kalman Filter. The right

picture shows another ball trajectory with a section of the coordinates removed to show the prediction capabilities of the Kalman filter.

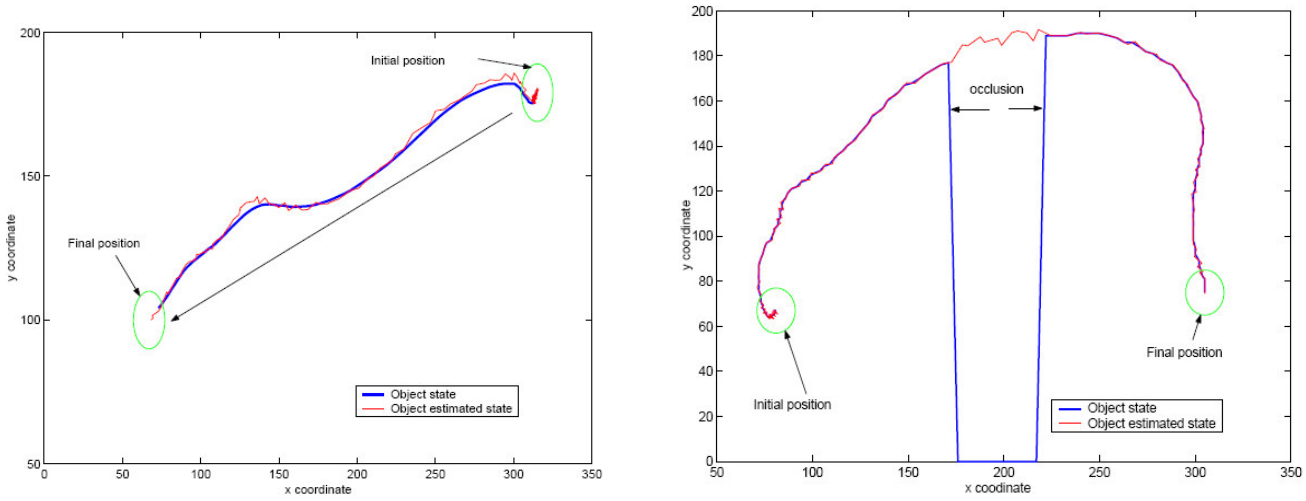


Figure 23: Position prediction with the Kalman filter and prediction of occluded path [106]

For somewhat more complex trajectories, the path of the ball cannot be modeled by a linear system. The model must be converted to a non-linear one and the Kalman filter will be replaced with the Extended Kalman Filter. The new state equation for the non-linear system is:

State Equation:

$$x_{k+1} = \mathbf{f}(k, x_k) + w_k \tag{54}$$

$$\begin{bmatrix} xk+1 \\ yk+1 \\ \Delta xk+1 \\ \Delta yk+1 \end{bmatrix} = \begin{bmatrix} \exp(-\frac{1}{4}(x_k + 1.5\Delta x_k)) \\ \exp(-\frac{1}{4}(y_k + 1.5\Delta y_k)) \\ \exp(-\frac{1}{4}\Delta x_k) \\ \exp(-\frac{1}{4}\Delta y_k) \end{bmatrix} + w_k \tag{55}$$

The results for analyzing the complex trajectory of the ball using the Kalman Filter and the Extended Kalman Filter are shown below (Figure 24).

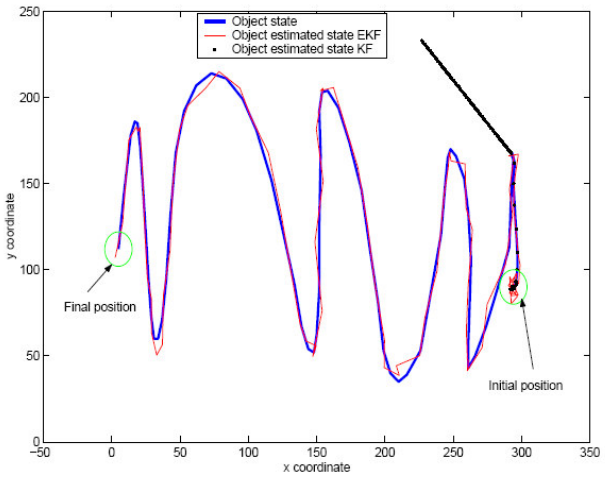


Figure 24: Results of Complex Trajectory with KF and EKF[106]

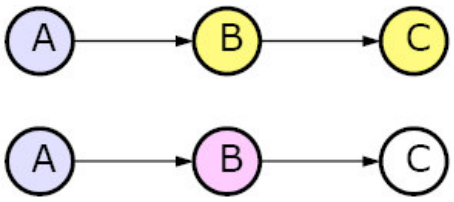
Bayesian Networks

Bayesian networks are graphical representations for probabilistic relationships among variables. The Bayesian network information is gathered from multiple sources [108–110]. Heckerman [108] describes 4 main advantages to using Bayesian networks over other data analysis tools.

1. Bayesian networks are effective against incomplete data sets. If a set of data has two variables that are strongly anti-correlated, Bayesian networks can make use of this property even if for some of the data set one of the variables is not measured.
2. Bayesian networks can show causal relationships in data. By uncovering the causal relationships in data, systems can be analyzed to determine states that are predictions to problems.
3. Bayesian networks and Bayesian statistical techniques can be used to combine prior knowledge and data.
4. Bayesian networks combined with other models can offer an efficient approach for avoiding over-fitting data.

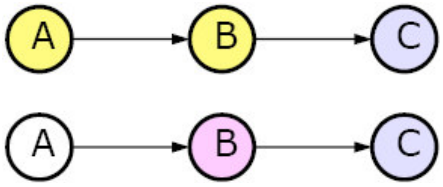
The first subject within Bayesian networks is the structure of the system. Since the joint probability distribution is to be calculated, it is necessary to have 2^N values to specify the joint probability distribution for N variables. This 2^N value can be reduced by using known information about relationships between variables. To represent this known information, some connection rules are set to govern the flow of probabilities for different variables as evidence arrives. There are 4 types of connections for the nodes (Note: nodes highlighted gray denote evidence, nodes highlighted yellow denote where information is traveling, nodes highlighted pink denote instantiation (i.e., truth value is known)):

1. Forward Serial Connection



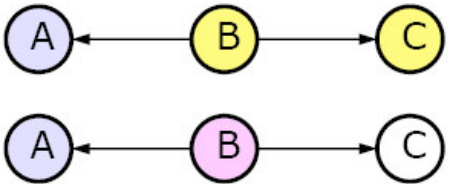
Transmits evidence from A to C (top) unless B is instantiated (bottom).

2. Backward Serial Connection



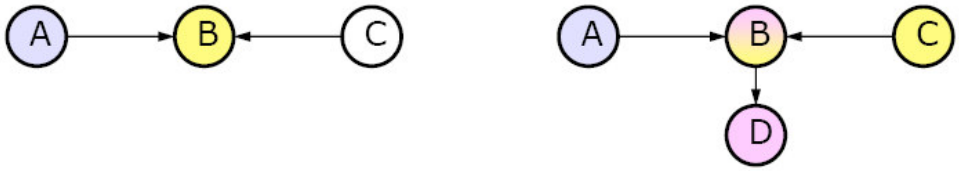
Transmits evidence from C to A (top) unless B is instantiated (bottom).

3. Diverging Connection



Transmits evidence from C to A or A to C unless B in instantiated.

4. Converging Connection



Transmits evidence from A to C or C to A only if B or a descendant of B is instantiated. After using the appropriate connections to construct the structure, joint probabilities are calculated by knowing the nodes that are D-separated and the chain rule.

D-separation [109]:

Two variables A and B are d-separated iff for every path between them, there is an intermediate variable V such that either:

- The connection is serial or diverging and V is known
- The connection is converging and neither V nor any descendant is instantiated
- Two variables are d-connected iff they are not d-separated

The Chain Rule:

The joint probability distribution is a product of all individual probability distributions that are stored in the nodes.

$$P(V_1=v_1, V_2=v_2, \ldots, V_n=v_n) = \Pi_i P(V_i=v_i \mid \text{parents}(V_i)) \quad (56)$$

where V are the variables, v are the values, and parents are all nodes that have a direct connection to the current node.

In the Bayesian network, there are 3 main parts. A set of variables having a finite set of values, a set of connections between the variables, and a specified set of joint probabilities for all nodes. A benefit to D-separation between variables is the declaration that the variables are conditionally independent given the evidence. This allows for probabilities to be reduced, making calculations efficient. Now that the joint probabilities of a single variable can be calculated, this information leads to the probability of a vector $\underline{x}$ composed of a sequence of n variables.

$$p(\underline{x} \mid \theta_s, S^h) = \prod_{i=1}^{n} p(x_i \mid pa_i, \theta_i, S^h) \tag{57}$$

where $\underline{x}$ is a set of n variables, θ_s is the vector of parameters that correspond to the probability distribution of all the variables $\left(\theta_s = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \right)$ (Note: each θ_i is a vector of parameters corresponding to the probability distribution of the i_{th} variable), S^h is the network structure with some event or hypothesis, and pa_i is the network parent nodes. If given a random sample G, what is the likelihood of it fitting parameters θ_s? The problem is expanded to include multinomial distributions for each parameter. A multinomial distribution allows for multiple parameters for each variable. Two assumptions are made to develop a solution. It is assumed there is no missing data from the random sample G and the parameter probabilities are mutually independent. These assumptions lead to the following equation which calculates the probability of a set of parameters θ_s producing sample G with structure S^h:

$$p(\theta_s \mid G, S^h) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} p(\theta_{ij} \mid G, S^h) \tag{58}$$

where n is the number of variables and q_i is the number of configurations in the structure. The equation calculates the overall probability of the data (D) being generated by structure (S^h) by taking the product of the probability distributions with parameters θ_{ij} across each parent node configuration ($1 \ldots q_i$). Furthermore, incomplete data can be approximated by finding parameters with the maximum likelihood to give variables:

$$\hat{\theta}_s = \arg\max\{p(D \mid \theta_s, S^h)\} \tag{59}$$

The general method to calculate the missing data value in the vector x is to calculate the structure with the maximum likelihood of generating the known values of the vector. Using this structure, calculate the most likely value for the missing variable. The methods to go about estimating the maximum likelihood can vary from Monte-Carlo, MAP and ML estimators, Gaussian approximations, etc. [108]. The structure can be estimated by:

$$p(S^h \mid D) = \frac{p(S^h)p(D \mid S^h)}{p(D)} \tag{60}$$

The most difficult component to calculate is the marginal likelihood of the data ($p(D \mid S^h)$) for all possible formations of the structure. The marginal likelihood of the data is the product of all the marginal likelihoods for each pair of possible variable states across all variables.

Robertson, Reid, and Brady [83] use a Bayesian network to combine direction and head pose and estimate gaze direction. The goal is to identify the gaze direction in medium scale images where the head size is roughly 20 pixels tall. They track the head by hand-selecting a region from the frame of the video and creating a normalized RGB histogram of 10 bins to define the skin tone of the person. By applying the histogram, they use that database to determine the approximate head gaze (Figure 25). The database consists of 100 samples for each of the 8 orientations of a 20x20 pixel region that contains the head. Using an approximate nearest neighbor tree, they are able to get 80% accuracy in matching the head direction.

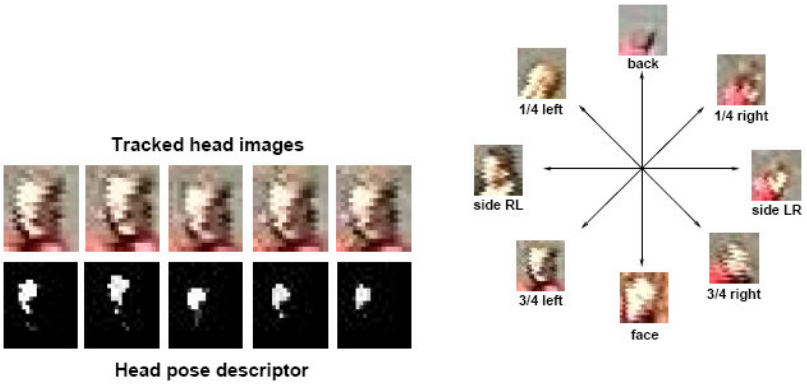


Figure 25: Skin detection and head orientation samples [83]

By detecting the motion of the head through sequences of frames, they extract the motion of the head. They calculate the joint probabilities of direction and head-pose. $P(h_{match}|h_{input})$ is the most likely head-pose match in the training data to the input head-pose. $P(d_{match}|d_{input})$ is calculated using a linear function $p(d) = 1 - \frac{d\theta}{45}$ where $d\theta = |\theta_{true} - \theta_{nearest}|$. θ_{true} is the heading from the trajectory of the tracked target. $\theta_{nearest}$ is the projection to the nearest of the 8 discrete directions. They calculate distributions for all 64 head-pose and direction combinations using the following equation for each gaze:

$$p(g \mid h,d) = \frac{p(h,d \mid g)p(g)}{p(h,d)} \tag{61}$$

where p(h,d) is uniform and p(h,d|g) is a zero-mean Gaussian centered on the current best estimate of head-pose and direction of motion. Assumptions are made corresponding to the unlikelihood of a person moving in a direction and looking in a direction greater than 90 degrees from the current moving direction. They set prior probabilities of 0.8 for $p(g_{expected})$ and 0.2 for $p(g_{unexpected})$. Using the previous time steps body pose (B) and head direction (G), the current body pose and head direction are computed for each frame:

$$P(G_t,B_t|G_{t-1},B_{t-1}) = P(G_t|B_t,B_{t-1},G_{t-1})P(B_t|B_{t-1}) \tag{62}$$

The results are shown in Figure 26.

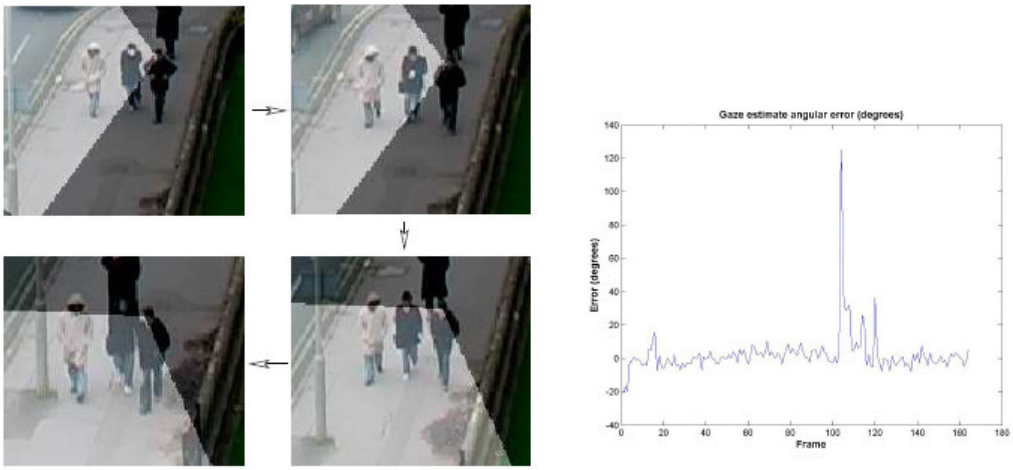


Figure 26: Gaze estimation example results [83]

The pictures on the left shows the estimated gaze and the plot on the right is the error of the estimated gaze angle from the human provided angular gaze estimations.

The different event tracking methods contain various strengths and weaknesses. The Bayesian Networks provide the most general analysis for event tracking and identification but are more complex to implement. For the experiments discussed later, the Hidden Markov Model provides a general look at the capabilities of our feature vectors to determine the feasibility of their identification of the participants and tasks while being slightly less complex to implement. The Kalman filters are very useful for tracking during occlusions, but are not really useful for tracking the hands and objects during occlusion tasks of our experiments since a majority of the motion performed will occur behind the occluder. The tasks will not give the Kalman filter enough information to track reliably behind occlusions especially due to the non-linear motions of the hands.

CHAPTER IV

SINGLE CAMERA TECHNIQUES

Object features are characteristics that describe the structure of the object. When dealing with camera sensors, these features are visual. It is very desirable that these visual characteristics allow for object locations to be detected regardless of object motion or varying backgrounds. Objects often range from tools in the task to actual body parts of the person. Objects can also be visual queues that provide information about the environment, such as shadows, to provide clues about objects in relation to each other. For multi-camera techniques, the systems are often complex. Though they offer an abundance of information, it is also more costly to acquire the system. Since the main focus of our system is use of a single camera for interdisciplinary use, the techniques employed by multi-camera systems will not be discussed. For single camera systems, the main visual characteristics extracted include color, shape, and texture. For human motion analysis, the objects fall into 2 main groups: parts of the person and interaction objects. Interaction objects have a lot of flexibility in controlled experiments because they can be chosen by the experiment designer. This flexibility allows for objects to be chosen that best interact with an experimental detection system. For tracking parts of people, one of the most prominent features of humans is the skin tone [71]. Often, the first step in detecting humans is developing a system that recognizes skin tones. When trying to differentiate between various colors, it is important to choose the proper color space to represent the desired colors. First, there will be a brief description on some mainstream color spaces.

Skin Tone Tracking

The RGB color space is an additive matrix consisting of red, green, and blue dimensions [115]. The additive space can be described as a cube where the origin is considered to be black (representing minimum contribution from all dimensions) and the furthest corner is white (representing maximum contribution from all dimensions). As the value of a single dimension grows, it represents a stronger occurrence of that primary color (i.e. red, green, or blue).

The HSV color space consists of the dimensions, hue, saturation, and value [115]. The hue space is used to represent the color and is described as a circle with red at the 0 degrees and shifting through all colors till converging at red at 360 degrees. The saturation describes the fullness of a particular color (e.g. the "redness" of red). The value describes the intensity of the color.

CIELAB is designed to model human perceptions of color. It strives to create a perceptually uniform representation of chroma a (A) and chroma b (B) that vary across different luminances (L). This color space is based on the CIEXYZ color space derived from a series of experiments that mapped human perceptions of color [114].

YCrCb is an encoding of the RGB format used to deal with the storage inefficiency caused by strictly encoding information in RGB format [115]. The Y holds the luma (i.e. brightness) component which contains a majority of the information and can be stored in high resolution while the chroma red (Cr) and chroma blue (Cb) are stored in lower resolution because of their lower information content.

Zarit et al. [90] did a comparison of 5 color representations (CIELAB, Fleck HS, HSV, Normalized RGB, and YCrCb) to determine the most descriptive color space for skin detection. Fleck HS is an altered version of the HSV color space that is biased towards the hues and saturations of natural skin tones. By training on multiple images, a histogram of skin values was created for each of the color spaces. They tested these histograms by applying varying thresholds from 0 to 1 and calculating the S (% of skin correct), SE (% of skin error), NSE (% of non-skin error), and C (overall % of correct pixels). By setting the threshold that allowed for specific values of S (% of skin correct),

the overall % correct was measured to determine the better color spaces. Then, a filter was added to correct pixel classifications by detecting the pixel classification of the surrounding pixels and adjusting the current pixel. The process of filtering is continued until the change in the number of skin pixels is less than 1%.

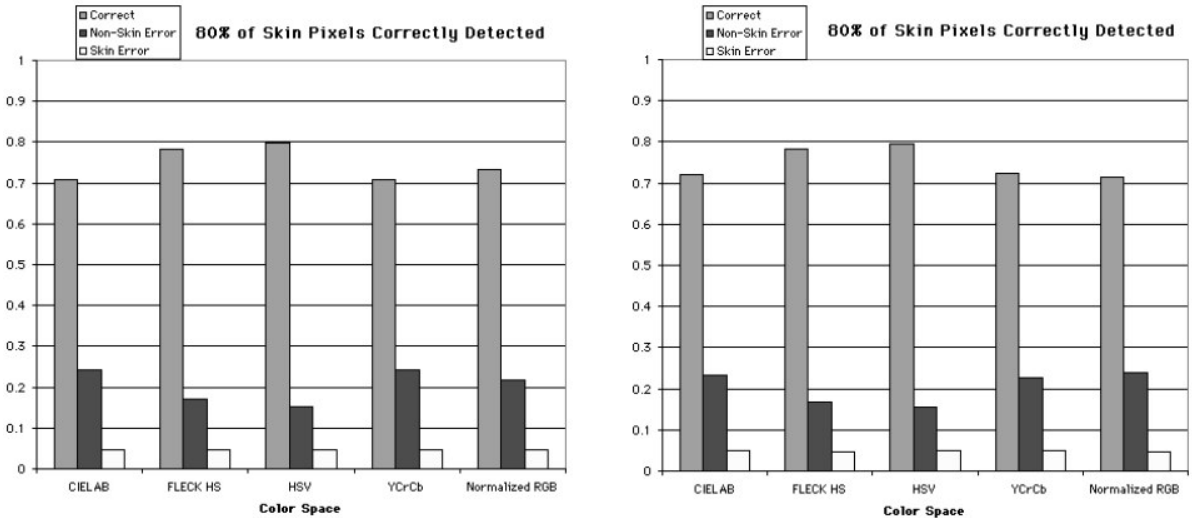


Figure 27: Results of comparison at 80%, (left) ½ original threshold, (right) ¼ original threshold [90]

In all of the comparisons evaluated, HSV and Fleck HS made up the 1st and 2nd best color spaces for modeling skin tones.

Shape Tracking

For shape tracking, the main problem is tracking the same points and correlation of points as an object moves. When dealing with a camera that returns 2D information and relating that data back to the 3D world, clues to determining and tracking these points usually are determined from their movement as the object moves [61,62]. If points can be determined on rigid objects, their movement can be described by a set of linear transformations. Of course, modeling deformable shapes is much more difficult because of different underlying models for different areas (Figure 28). For example, the corner of a person's mouth can change independently of how the head turns. Another method of shape

tracking is to overlay shapes across a region identified as being the object of interest. Examples were shown earlier (Figure 3) under modeling the body structure.

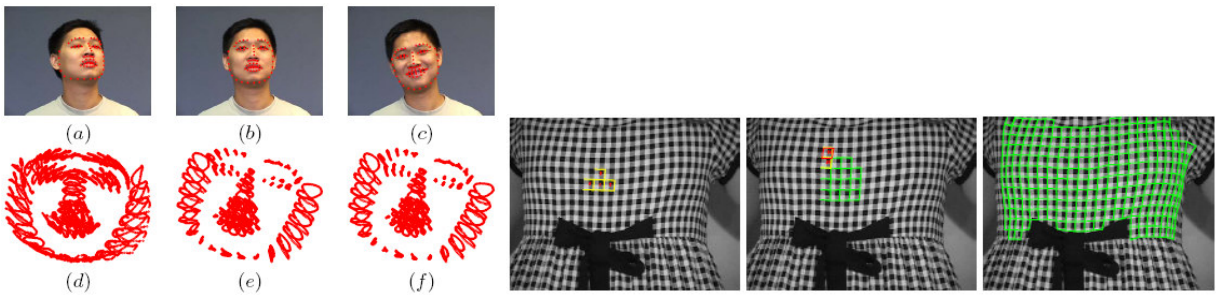


Figure 28: (left) Tracking of face movements, (right) Tracking of texture patterns [62]

Optical Flow

In the temporal boundary area for visual systems, the methods involve using optical flow and object features to detect motion queues. The temporal boundary area is the area dealing with time partitions of events. Optical flow is the regional motion throughout subsequent frames in a stream of images. The motion vectors in these regions show object boundaries by connecting similar vector regions. After detecting these motion queues, the objects can be identified and tracked. These motions can be used to give clues about many machine vision problems such as occlusions [35], distance estimations [107], key event moments, etc. Optical flow techniques rely on local space and time displacements of image values [2]. A common method for calculating optic flow is using block based motion estimation. The block based motion estimation takes a block from a frame of an image and finds the most similar block in the next frame with the closest distance spatially. A vector field is calculated from this motion estimation. One of the assumptions of optic flow for determining objects is that neighboring blocks have similar motion vectors. By using these vectors, object shapes and motion can be determined.

Optical flow fields can be used to provide vast amounts of information about the activities of a scene. Aloimonos et. al. [35] used optical flows to determine the change of occluded areas of the background by other objects and determine the distances between the object in motion and the

background. They described 3 types of classes for motion involving independently moving objects (Figure 29):

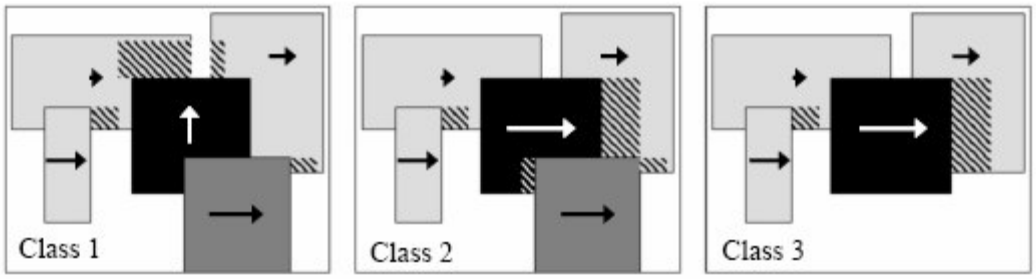


Figure 29: 3 Motion Class Examples [35]

Class 1: The independent object (represented as the black object) is moving in a different direction independent to the other background objects (represented as the gray objects). Note: the darker grey objects signify an object in front of the independent object and the striped area represents the area occluded by the moving objects. Independent objects using class 1 motions can be identified by performing motion-based clustering.

Class 2: The independent object is moving to the right along with the background objects. It moves at a faster speed but is not enough to rely on just motion based clustering. To detect this motion, they take advantage of ordinal depth conflicts. Ordinal depth is the perceived order of the depth of objects in relation to each other (i.e., an object being in front or behind another object). Using the occlusion of the black object by the dark gray object, this gives the perception of the black object being behind the gray object. By constructing structure from motion, the black object appears to be in front of the gray object because it moves faster. The two conflicts allow for the detection of the independent object.

Class 3: The last class is similar to the second class but does not have the occluding object (dark grey) to give clues about the independent object (black). To detect this set of motions, they use two calibrated stereo cameras to calculate the motion/depth ratios and group them into 3 groups. A group for the largest set of group ratios (background group), a group for pixels with greater group

ratios than the background, and a group for lesser group ratios than the background (also considered background) are extracted. The groups not of the background group are considered an object in class 3 motion.

They develop a general algorithm to detect the 3 classes:

1. Input video sequence
2. For each frame in the video
    a. Find the forward and reverse flows using the occlusions
    b. Select a set of pixels using phase correlation (see below) between two frames.
    c. Find the background motion using the set of pixels.
    d. Detect Class 1 moving objects from background
    e. Find ordinal depth relations using the flows from (a)
    f. Detect class 2 moving objects
    g. If stereo is available, detect class 3 moving objects

Phase correlation involves calculating 4 parameters from the motion in an image to detect the background. The 4 parameters are the x and y translations in Cartesian coordinates, the scale in logpolar representation, and the z rotation. These 4 parameters are most dependent on the background (assuming the background is the largest object in the video) and are used to select optimal points at which to sample the true background motion. Results can be seen in Figure 30.

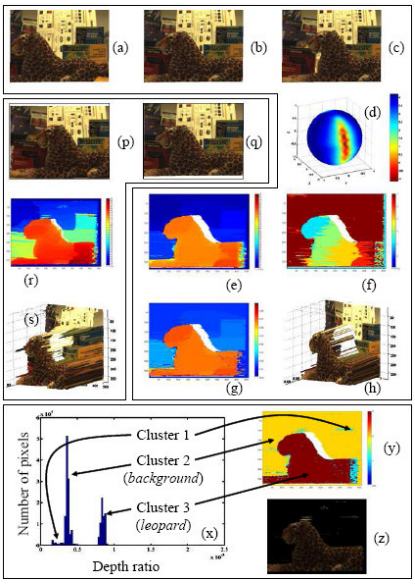
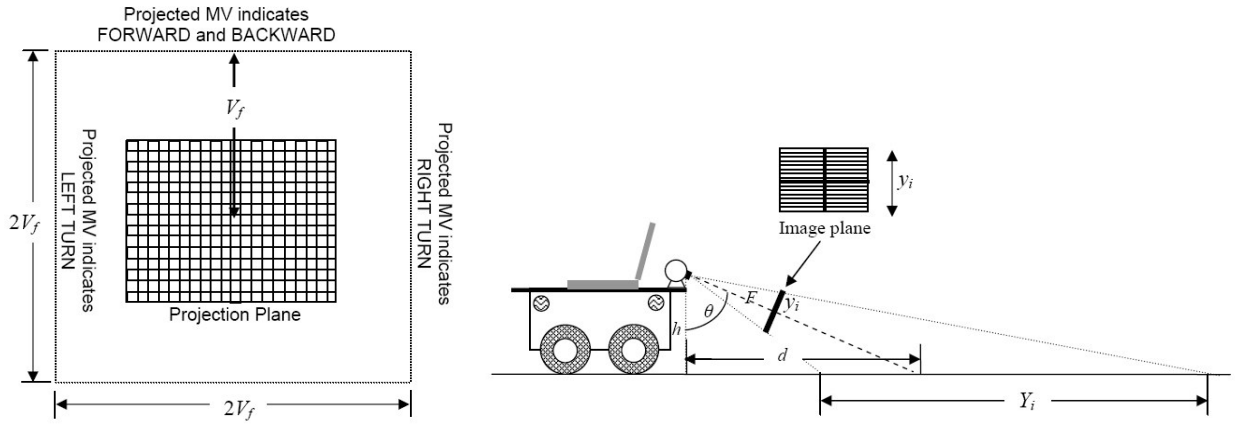


Figure 30: Aggarwal et al. results: (a)-(c) show 3 frames in a video sequence. (d) shows the computed motion valley. (e,f) show the forward and reverse flows. (g) shows the inverse depth from motion. (h) shows 3D structure from motion. (p,q) show the pair of stereo images. (r) shows the inverse depth from stereo. (s) shows the 3D structure from stereo. (x) shows the cluster groups. (y) shows the clusters in the image. (z) shows the Class 3 moving object.

Krootjohn [107] uses optical flow to determine robot location through visual odometry and detect precipice proximity. MPEG encoders have optical flow information calculated and stored in their structure. Krootjohn was able to extract that information to be used for visual odometry and precipice detection in real time. By using a single stationary calibrated web-camera angled toward the floor, he was able to calculate the translation and rotation of the robot from the flow vectors in specific regions of the viewing area. Using the vectors that project into the left and right sides of the virtual square, these values estimate the robot rotation to the left and right (see Figure 31a). The vectors that project into the upper region are used for both the forwad and backward motions (Note: The backward motions are identified by inversing vectors that have a negative y component and determined if is projects to the top of the virtual square.)



(a) (b)

Figure 31: Robot setup: (a) Image plane, (b) System configuration [107]

Using these motion vectors, the forward and reverse translations are calculated by the following equation:

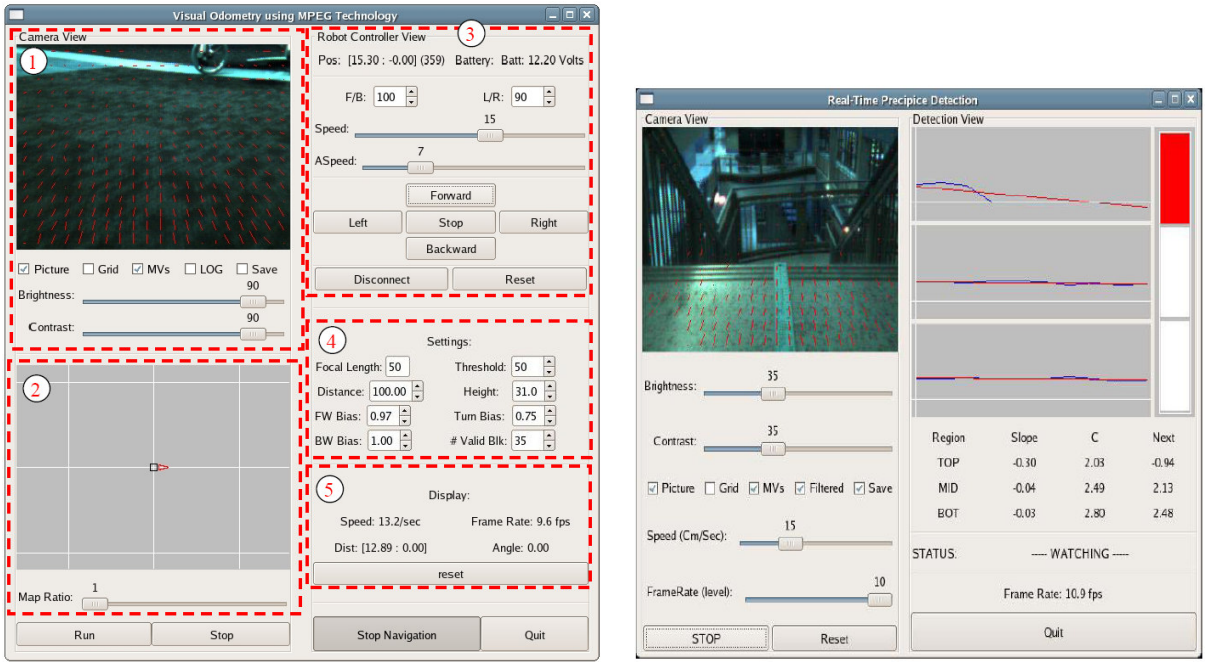
$$Y_i = h\tan(\theta + \beta_i) \tag{63}$$

where $\beta_i = \tan^{-1}\left(\frac{y_i}{F}\right)$, F is the focal length, θ is the tilt angle, h is the height of the camera, and y_i is the projected motion vector on the 2D image plane. Using the Y_i values calculated between each frame and summing each distance, the total distance is calculated. Likewise, the rotation can be calculated from the following equation:

$$\phi = \tan^{-1}\frac{(x_0 - x_c)}{F} \tag{64}$$

where x_0 is the point before rotation (x component of the motion vector), x_c is the camera translation occurring during the robot rotation, and F is the focal length. After calculating the translation and rotation of the robot, an interface showing the motion vectors and projected travel of the robot was created (Figure 32a). A similar interface is used for precipice detection where the bottom half of the image plane is broken into a 3x5 blocked area. The first row is the Watch level, the second row is the

Warning level, and the third Row is the Panic level. Each level has a binary safety value of 0 or 1 (1 being safe, 0 being not safe). If the average magnitude of the motion vectors in the Watch level are less than half that of the Panic level, it returns a 0 value. The Watch level has 3 frames for the average to rise above the threshold before the 0 value is returned. When the Watch level returns 0, the Warning level is then active and follows the same logic as the Watch level. The Panic level flags when the number of macroblocks (i.e., 5 blocks making up the section) that have an average vector above 0 is less than half and the robot stops. The interface for the precipice detection can be seen in Figure 32(b).



(a) (b)

Figure 32: Krootjohn's interface: (a) Interface for robot odometry, (b) Interface for precipice detection [107]

CHAPTER V

STATE OF THE ART HUMAN MOTION ANALYSIS SYSTEMS

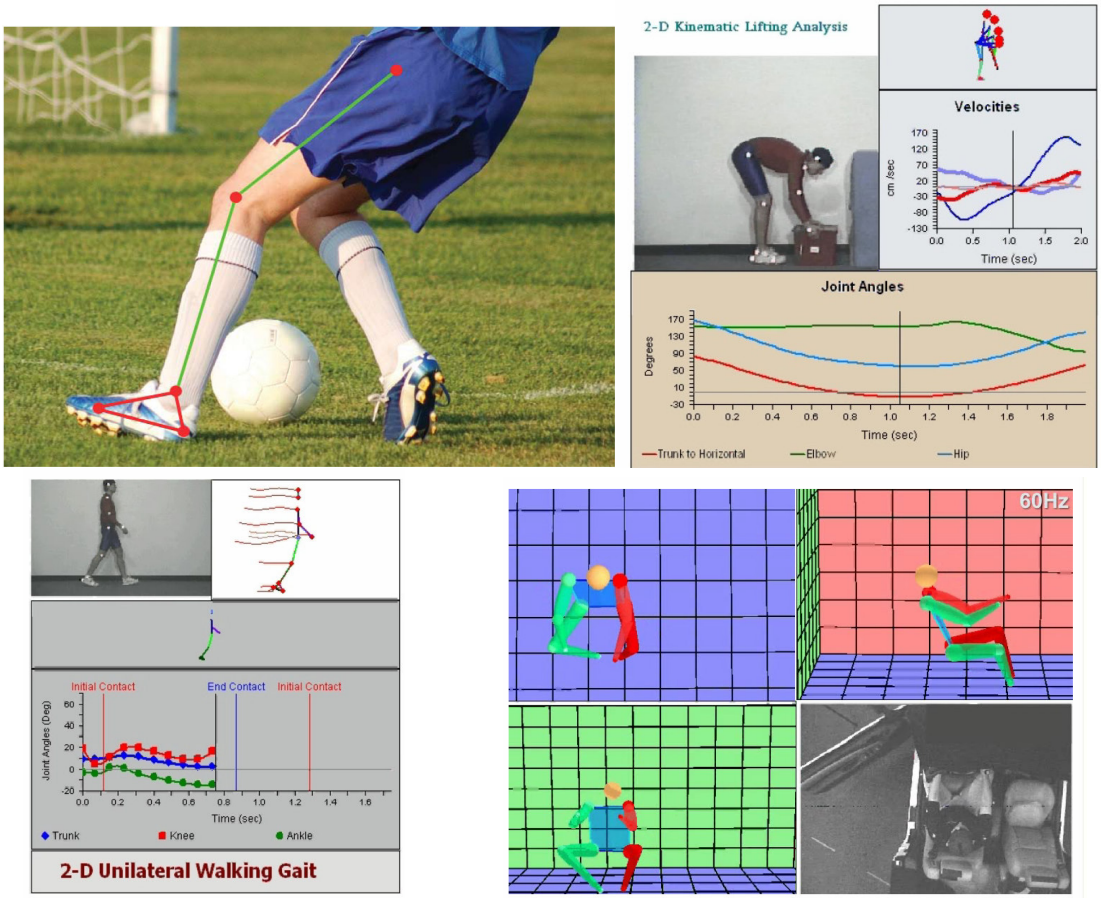
To develop an effective tool for human motion analysis, it is important to look at some of the latest systems and discuss their capabilities. This chapter focuses on covering the tools being used currently in research labs as well as products available for purchase. The systems discussed tend to focus on generalized tracking and human motion analysis. They will be described in terms of hardware and software necessities, price, and capabilities. The 5 systems that will be discussed are divided into 2 groups, research institution developed tools and commercial products. The 2 commercial products are Vicon Motus system and the Qualisys Mocap system. The 3 institution developed tools are from the University of Texas at Austin under Aggarwal, the University of Maryland under Aloimonos, and Carnegie Mellon University in the Robotics Institute.

Motus System

The first system is a tracking system known as the Motus system created by Vicon [120]. The Vicon Motus system is a software package based on Windows that connects to digital camcorders via Firewire. The system is used for many mainstream applications such as video game development, movies, and various tracking applications. The company's various systems have been used over the past 20 years of its existence by well known clients such as Sony, Microsoft, Activision, etc.

The Motus system provides a number of applications which include but are not limited to pattern tracking, gait analysis, skeleton model application, and multi-camera calibration/synchronization. The system specializes in full body tracking, by tracking points throughout a video sequence; an example can be seen in Figure 33. These points are either initialized by the user or are represented by visually unique representation usually specified colors to used for

tracking. These points naturally lead to skeletal models which are used to extract motion of the points relative to other points in the model. In most of the demos, the more complex skeletal models were formed by markings placed on the subjects. Some of the common features that are extracted from the models are point velocities and joint angles. These initial features lead to more complex model features such as center of mass and stress estimation. The system is robust in the number of environments that it may be used. The system has also been used with a number of rehabilitation clinics to allow monitoring of patients during activities especially walking. The system has the ability to perform some gait analysis such as determining the individual steps from the entire sequence of walking.



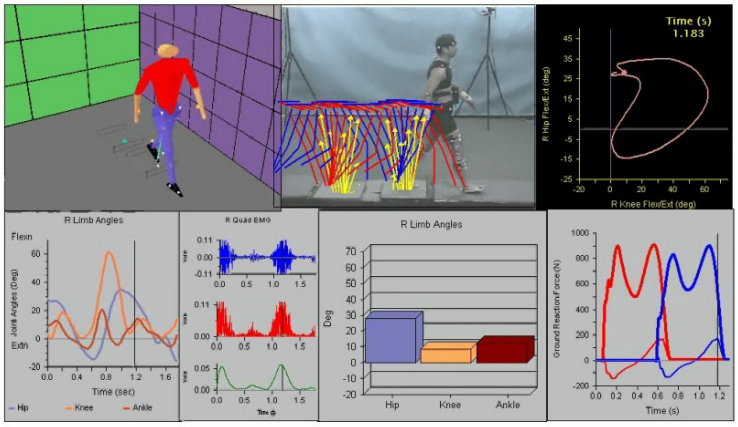


Figure 33: Sample displays of Motus capabilities

The Motus system has a lot of user learning that is necessary to use the tool, but the tool itself can scale to meet the needs of the task to be observed. The package allows for a single camera system and has the option of scaling up to 244 cameras using their hardware options. Vicon offers a number of product that can be purchased with the system such the Vicon T and MX series cameras, Giganet device (for the monitoring and synchronization of cameras as well as Ethernet connection), and remote video synchronization unit (RVSU). The basic software system was priced starting at $5,000.00 in 2008. As the system expands to a 3D setup, the price ranges from $40,000 – $50,000 for setups involving digital cameras with varying speeds and resolutions.

Mocap System

Another tracking system available on the market is the Qualisys Mocap system [121]. Like Vicon's Motus system, this system is also a Windows based system and specializes in point tracking. Their systems have been in development in Sweden since 1989. Qualisys products have been used with various hospitals and universities such as Madonna Rehabilitation Hospital, University of Salford, University of Massachusetts Amherst and Stanford Biomotion Labs. The complete Mocap system offers all the materials necessary to build a motion tracking work area. Materials include

cameras, wall mounting brackets, markers and body suits, calibration and camera lenses, tripods, force plates, and Qualisys Track Manager (QTM) software as can be seen in Figure 34.

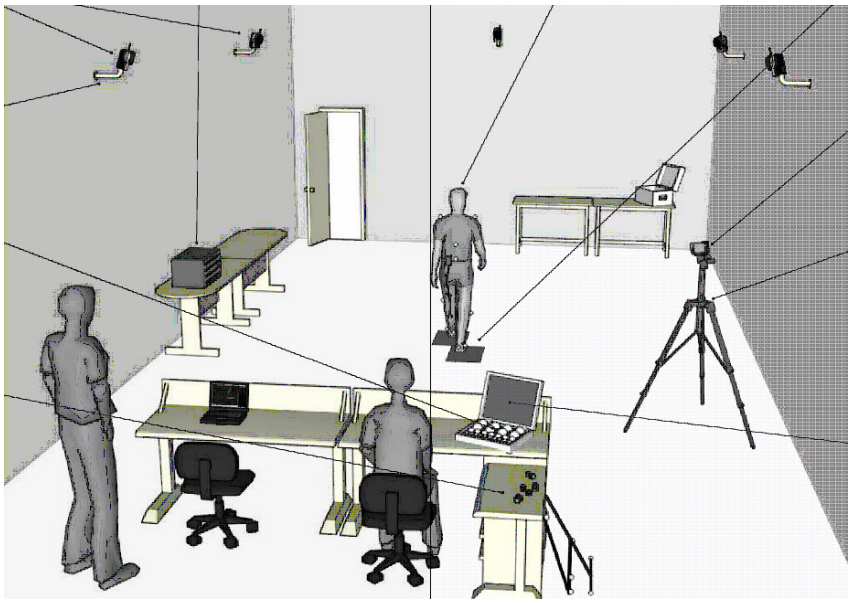


Figure 34: Overview of complete Mocap system

The Mocap system employs the camera array approach to extract 3D information. It offers software packages such as QTM 3D tracking, Visual 3D model builder, Qualisys Video Analysis (QVA), and Motion Monitor. The QTM offers 3D tracking of markers from several synchronized camera inputs as can be seen in Figure 35. It also offers point tracking and model analysis from both a 2D and 3D perspective. The Visual 3D model builder allows the user to specify different segments of the human body for analysis. The QVA monitors and manages all video feed into the system, which is useful for management of experiment recording and documentation. The Motion Monitor software uses high speed, high resolution, digital Qualisys cameras to track motion. Some of the applications for this package include eye tracking and sports performance enhancement.

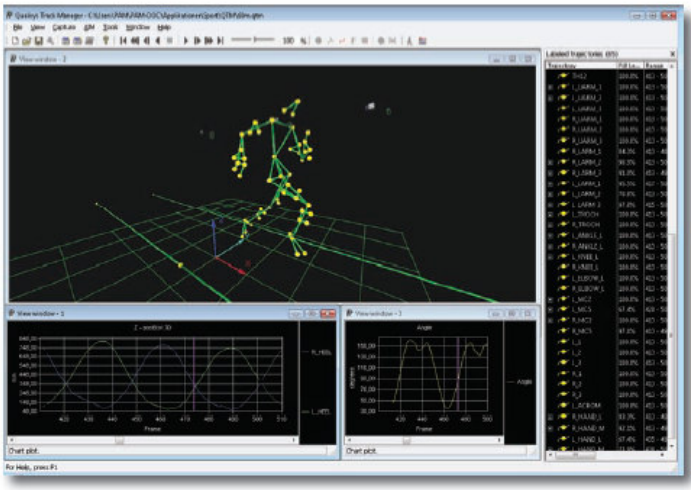


Figure 35: QTM Sample Display

The Mocap system has a lot of user learning that is necessary to use the tool, but similar to the Motus system has the option of scaling upward. The pricing of the Mocap software, upgrades, and tech support ranges from 1250 euros (~1800 dollars) for 1 year to 3000 euros (~4300 dollars) for 3 years.

University of Texas System

Whereas the previous two systems are for purchase commercially, the following systems are developed at different universities. The system developed at the University of Texas is based on analysis of human action for surveillance in public settings. The system focuses on extracting high level descriptions of the actions occurring. Due to the scope of the application, one of the goals for the system is real-time analysis. The system operates from a single camera perspective and extracts information using 2 basic detection methods. The first method is a human-blob segmentation that extracts the position of standing humans using differencing techniques to segment foreground and background as can be seen in Figure 36 [122].

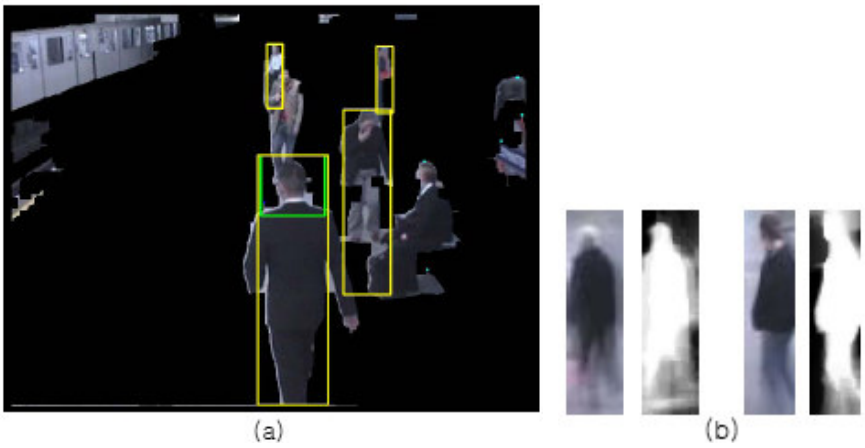


Figure 36: (a) Human tracking sample, (b) Motion foreground extraction

The second method estimates the position and orientation of the head. By assuming the upper portion of the bounding box created from the human segmentation is the head, the image is compared to a database of images resembling the head at different orientations. The information is organized into a structure containing 2 normalized arrays. The first array contains the mean pixel value of the person and the second array contains the probability of each pixel being a foreground pixel. The system tracks the person by estimating the next position in the following frame and matches the appearance profile of the person by finding the image region with the minimum difference.

This system has been applied to a number of surveillance applications such as human tracking in occlusion environments [122](Figure 37), activity recognition in surveillance videos [123] (Figure 38), and car tracking in traffic videos [124] (Figure 39).

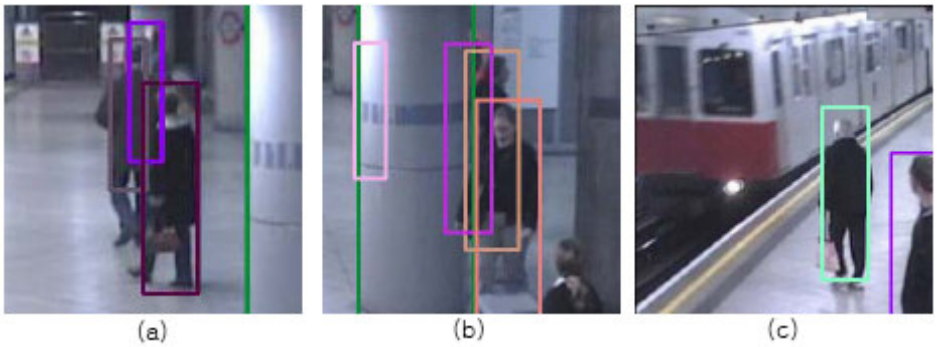


Figure 37: Subway Surveillance Tracking

Figure 38: Activity Recognition

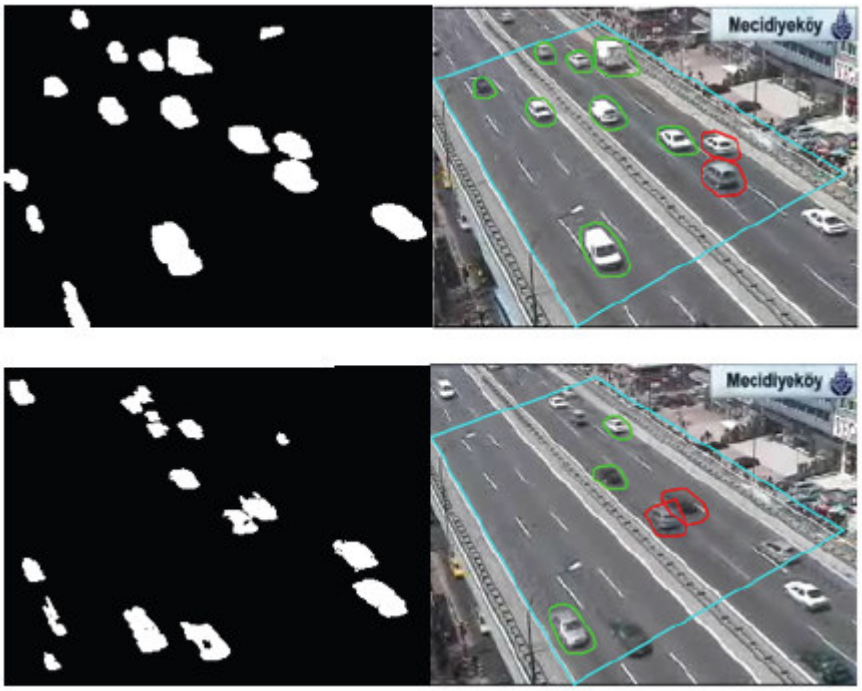


Figure 39: Vehicle Tracking

University of Maryland System

The goal of the system developed at the University of Maryland under Aloimonos is to develop formal models of human action [8]. To create a formal model for human action, a set of representations must be established called visual verbs. These visual verbs are motions performed by a single human usually by some translation of a limb. Adverbs and adjectives are used to develop the descriptions for more complex actions such as walking. Using optic flow methods combined with background subtraction, the silhouette of a person is analyzed for minima and maxima of flow values.

These minima and maxima indicate key moments in the action that are used to parse the overall human action. The poses captured at these mimima and maxima are theorized to be all that is necessary to allow for recognition of the action.

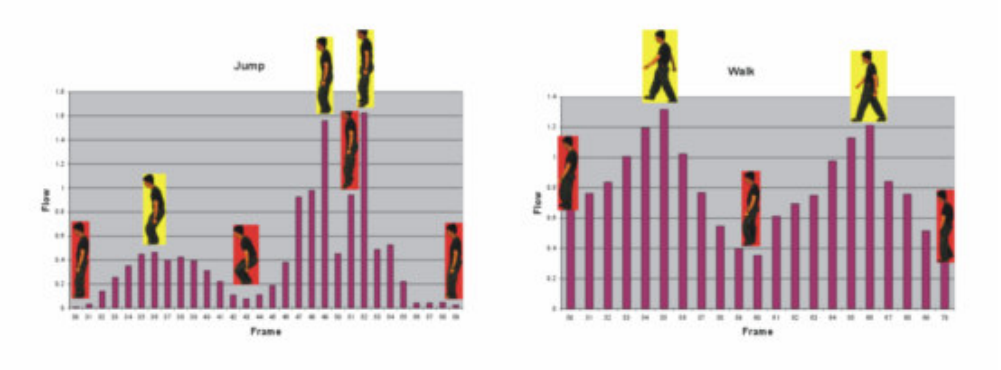


Figure 40: Example of flow minima and maxima with associated pose

To generalize the poses, the information was gathered from an 8 unit camera array viewing each action from an equally distributed full 360 view. To help generalize the data, each silhouette is averaged across multiple participants. The combined views of each pose are collected into a set of training examples for an HMM.

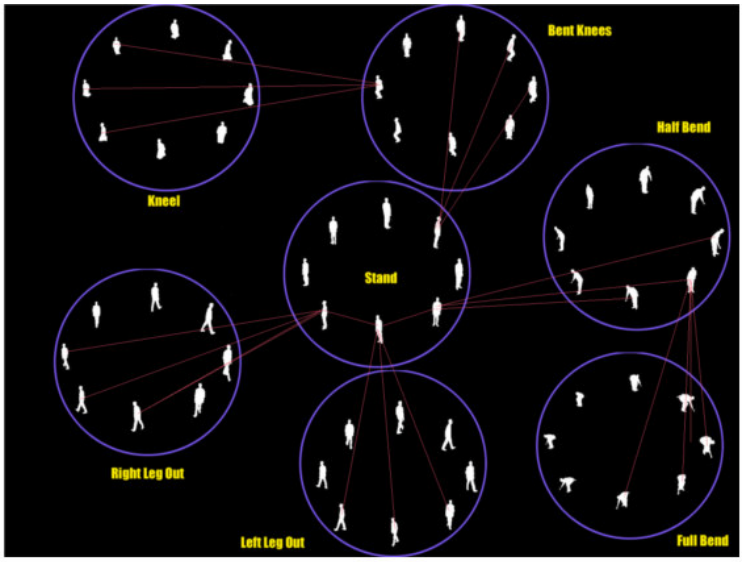


Figure 41: Pose Descriptions

The HMMs are used to describe the tasks in a sequence of poses. The sequences of poses in conjunction with the trained HMMs are used to recognize actions in a general setting.

Robotics Institute Systems at Carnegie Mellon

The final system was developed at the Robotics Institute at Carnegie Mellon University. The People Image Analysis group focuses on tracking humans or body parts of humans using visual information provided from single cameras. Some of the systems apply shape models and match them with body part characteristics to extract information. The 3D Head Motion Recovery system applies a cylindrical model to the head of a person [125]. By matching the inner corners of the eyes, edge of the nostrils, and the corner of the mouth to points on the cylinder, the 3D rotation and translation of the head is estimated.

Figure 42: Head estimation using cylindrical model

They also apply models to the hand to detect shapes and motions to be used for sign language recognition [126,127]. The model for the hand consists of shape (contour of hand), position, and motion change. A state space is created that shows "stable" nodes (moments when the hand state is fairly constant therefore having a shape) and "transition" nodes (moments when the hand is moving rapidly and no shape is estimated). By traversing the state map of the hand for each gesture that is has been trained on, detection of various words are detected and recognized.

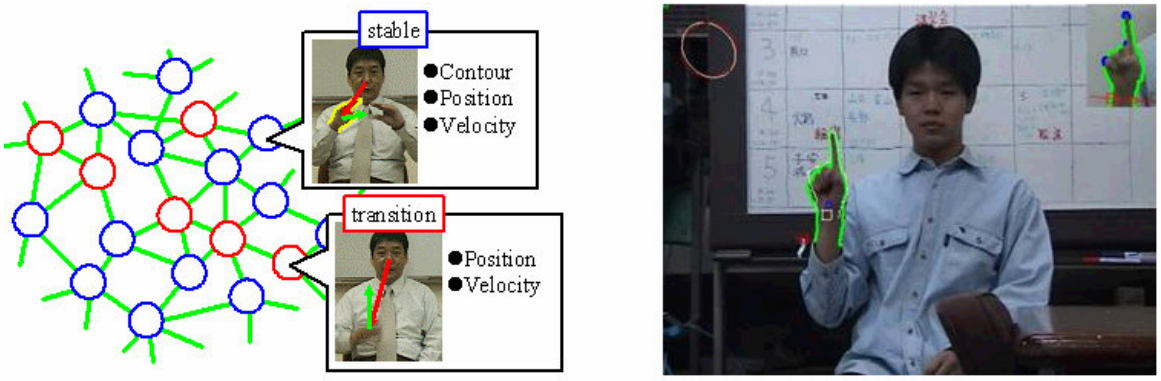


Figure 43: Hand state space and hand detection

Other systems use tracking of people for surveillance. In the case of tracking multiple humans in a scene, some of the standard tracking methods are used such as kalman filtering, particle filtering, and mean-shift tracking [128]. While these methods are accurate, they also contain some amount of weakness due to occlusions. By combining the previous methods with a probabilistic graphical model, the inaccuracies caused by lost patches are reduced.

Figure 44: People tracking with occlusions

Although each of these systems have their strengths for their chosen applications, each one also has its limitations. The two commercial systems provide excellent tracking for motion analysis under controlled conditions. They also are specialized in skeleton based modeling which limits their ability to visually recognize object contours. The commercial systems are pricey investments for a

laboratory, but if the means are available, they provide very accurate data and professional tech support for research.

The 3 university systems were developed for research toward increasing knowledge in the desired field. The surveillance systems are specialized toward tracking even with occlusions, but are limited in the action recognition available outside full body motions. Smaller motions are much more difficult for these systems to detect. The part tracking systems use more sophisticated methods for estimating states of the body parts. The models used are very accurate for estimating specific body parts in the desired range of actions. The systems are fairly limited in use for general purposes, but do very well in their specifically designed workspace.

CHAPTER VI

SINGLE CAMERA TASK OBSERVATION AND RECOGNITION SYSTEM

One of the goals of our lab is to take our original visual navigation system and adapt it to be an easy to use general tracking and recognition system for use in fields outside of engineering. This system has numerous uses and can be applied across many interdisciplinary research fields. The system should be robust in its abilities, yet cost efficient as to be available to most any lab with minimal equipment purchase. This goal was developed as we worked with the Vanderbilt University Psychology Department under an NSF grant to study human activities. Their research had shown that subjects instructed to do a number of tasks had common key moments in their activities corresponding to grasping and releasing of objects, glances toward and away from the objects/workspace. Using a student to indicate when the activities occurred, they were able to observe a high correlation with data picked up from the measurements of head and hand states. The vision system our lab has developed has been tested in the area of robot navigation coupled with working memory [51] and has shown good results in its ability to learn from the visual data provided. The navigation system was adapted to be useful in the psychology research data collection. Before explaining the goals and experiments of our new system, the visual processing method of the system and action parsing method will be covered.

System Hardware

The input device used by our lab is a Sony DCR-VX2000 camcorder. A consistent price at an online retail store for this particular camcorder could not be found, but the next generation camcorder, the Sony DCR-VX2100, could be found for as little as $1,000 from online retailers. The Panasonic DVX100, a similar type of camera, was used by the Psychology Department in gathering data and can

be purchased from online retailers at around $1,000 as well. The frames of the video from the camcorder are 480x720 pixels. This information is transmitted via 4pin-6pin FireWire cable ($10 - $20 pending length) to a laptop FireWire card ($10 – $30) aboard a laptop. The original program ran on a Linux operating system in C++. This program was modified to also extract frames from an AVI video to allow training of the system and segmentation of the frame. By allowing for the system to read from AVI videos, this gives our system the ability to do perform automated processing of a number of pre-recorded videos on a stationary desktop computer.

Our system starts by training on the objects of interest that are currently provided by the user. To represent these objects, our system uses feature vectors composed of a high dimensional HSV color space histogram along with a Laplacian texture measure. The initial visual system is also described in Tugcu's dissertation [111].

HSV Color Histogram

The frames are captured in RGB format which means the information is encoded in intensity values of red, green and blue. The image is converted to HSV format which means hue, saturation, and value. The hue value, which defines the color family, ranges from 0 to 1 where all the colors are represented as the value increases starting color (red) at 0 and returning to the same starting color (red) at the value 1 . The saturation parameter S is the degree of purity from 0 to 1 (e.g. the "redness" of the red or how vibrant the red is). The value parameter V defines the brightness of a color and it is also from 0 to 1. The color space conversion from RGB to HSV is computed as shown in the pseudocode below:

$$MAX = \max(R, G, B)$$
$$MIN = \min(R, G, B)$$
$$V = MAX$$

$$if\ (MAX = 0),\ then\ \ V = S = 0\ \ and\ \ H\ is\ undefined$$

$$S = \frac{MAX - MIN}{MAX}$$

$$if\ (R = MAX),\ then\ \ H = \left(0 + \frac{G - B}{MAX - MIN}\right) \times 60$$

$$if\ (G = MAX),\ then\ \ H = \left(2 + \frac{B - R}{MAX - MIN}\right) \times 60$$

$$if\ (B = MAX),\ then\ \ H = \left(4 + \frac{R - G}{MAX - MIN}\right) \times 60$$

$$if\ (H < 0),\qquad then\ \ H = H + 360$$

Once the HSV values are computed, the next step is to construct a probability density function (pdf) of the HSV distribution of colors. The pdf is essentially a histogram of HSV colors and computed using a color quantization method as follows: The hue space is evenly distributed into 100 bins, ranging from 0 to 1. The saturations and values are each evenly distributed into 10 bins that are also ranging from 0 to 1. This results in 10,000 different possible color representations. For a region selected by the user in the image (usually composed of a single object), an HSV color histogram is obtained by accumulating the HSV values of each pixel in the region. The region is broken into 7x7 blocks of pixels. Each block has its colors represented by the color histogram and is stored as a feature vector of that particular object.

Texture Measure

In order to understand the texture of a region, a spatial filtering technique, based on the Laplacian operator, is applied to the image. The Laplacian operator is often used for edge detection, where the regions that have rapid intensity changes are highlighted in the image. The Laplacian $L(x, y)$ of an image having pixel intensity values $I(x, y)$, is defined as:

$$L(x, y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \tag{65}$$

Since an image is composed of a set of discrete pixels, the following kernel, which approximates the 2nd derivatives in the definition of the Laplacian equation above, is applied.

$$K_{LAP}(x,y) = \begin{array}{|c|c|c|} \hline -1 & -1 & -1 \\ \hline -1 & +8 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array} / 8 \qquad (66)$$

This Laplacian measure is calculated across the 7x7 block and the mean value is stored as the final value in the visual feature vector completing our 10,001 feature vector.

After a database is created for all the desired objects, this database is compared to new image. The center of the window is then moved throughout the entire image and a feature vector is obtained for each displacement. Currently, the displacement is 4 pixels in the horizontal and vertical directions. The new vectors are compared to the training data and by using a nearest neighbor approach. The new vectors are represented by the most likely group's color representation in a new 118x178 pixel segmented image.

Distance Metric

It is expected that the patterns that are members of a specific perceptual cluster should be closely positioned in the pattern space, while those from different percepts should be positioned further apart from one another. Our system uses the Euclidean distance for the color histogram and the L1 norm to compare the texture measure. In a *K*-dimensional space, the metric distance is given by:

$$d_{ij} = \left(\sum_{k=1}^{K-1} \left| x_{ik} - x_{jk} \right|^2 \right)^{1/2} + \left| x_{iK} - x_{jK} \right| \qquad (67)$$

where *K*= 10001 in this system. As the dimension of the vectors increases, the number of training samples should also be increased considerably as much as possible in order to obtain meaningful percepts. The pure nearest neighbor search algorithm works at an order of N*d comparisons for each

new vector where N is the number of vectors in training set and d is the number of dimensions composing each vector. To shorten the amount of processing necessary for each comparison an approximate nearest neighbor tree is constructed.

Approximate Nearest Neighbor Tree Construction

The tree structure is formed as follows: Initially, at the root or first level of the tree, three points, which represent the cluster centroids, are selected randomly and then the whole data set is clustered into three subsets by assigning each feature vector to its closest representative cluster center according to the proximity distance measure. At the second level, three subsets are obtained and the same procedure is applied, which in turn results in 9 subsets or in other words nodes for the tree. This procedure continues until either all the leaf nodes belong to the same object class (a pure node) or the number of leaf nodes is below some limit, e.g., a hundred. Every feature vector in the leaf nodes has a landmark associated with it [111].

Since all the centroids for each node in the tree structure are known, searching the tree is straightforward. Given a new feature vector, the three similarity measures, which are between the new vector and the centroids of the three sub-nodes at the second level of the tree that belong to the root node, are computed. The winning sub-node is the one that is closest to the given feature vector. At the third level of the tree the same procedure is applied and a winner sub-node is selected and the fourth level of the tree has been reached. This procedure is terminated when the search has descended to a leaf node. If the leaf node is pure, that is all the feature sets belong to the same class, then the vector is labeled as the leaf node's class label. If not, that is the data set in the leaf node is mixed and below some threshold limit, then a nearest neighbor search is applied using the vectors of the leaf node, and the vector is labeled with the training vector's label that is closest [111].

Once the system is trained on all person and the objects involved with the task, the trained tree is used to processes the entire set of frames for a video of a person performing the task. Each tree is trained on the set of the same task since each task has a certain set of objects.

After each section of a new image is classified via the approximate nearest neighbor tree, the resulting pixels are allotted a classification number (i.e., label) depending on the classification. The pixels are represented for visual display via a predefined color for each classification number. Using this new classification image, each classification group is extracted and a connected component labeling algorithm determines the groups' pixels for all the objects.

Database Training

Our system uses an interface that allows the user to select regions out of sample images and provide classification. The system proceeds to break the selected region into 7x7 blocks and create feature vectors for each of the patches. This method allows for large amounts of training data to be gathered quickly. The assumption is that applications this system is used for will produce copious amounts of data. Each image has a 21004 vectors contained within at the block size of 7x7 with a hop of 4. For applications with controlled environments (i.e. visual characteristics of desired objects are chosen to be easily separable for the color space), the training of a tree can extend across multiple sessions of different subjects with the same objects needing little reinforcement learning necessary to extend the training across multiple sessions. In fact, it is safe to say that in actuality the number of trees was equal to the number of tasks created and each tree was trained on all of the participants' versions of that particular task.

Segmentation

Each classification image is broken into object images. An object image is an image containing only the pixels whose patches are classified as a particular group. Once an object image is

created, all the object classifications are set to a value of 1. To reduce noise the image is filtered with a variable averaging filter in the following steps:

1. An *n x n* filter size is chosen by the user because the size of the resulting image depends on the patch parameters appropriate filter sizes vary. The general rule is to use a size equivalent to the patch size or smaller for the application.
2. For there to be any groups stored after the group image is filtered, a group must fill at least percentage 1 (P1) of the filter.
3. If the conditions for viable groups are met, the largest value (M) from the filtered object image is stored and all values in the filtered object image that are greater than percentage 2 (P2) of M are changed into a value of 1 while the others are stored as 0 creating a filtered binary object image.
4. This new image is grouped using a connected component labeling algorithm. Statistics about each group are stored into the object descriptor class (i.e., class containing number of pixels, width, height, and (x,y) centroid of each group)

The segmentation information for each frame is stored in a text file and accessed via Matlab to perform the second stage of the analysis.

CHAPTER VII

EXPERIMENTS AND RESULTS

Much research has assumed that human-generated action segmentations represent the combined influences of basic perceptual cues such as changes in the direction of moving body parts, and more complex cognitive constraints such as an understanding both of context-consistent sequences of actions, and of the actor's goals. For example, in one recent study [112], subjects were asked to segment the movements of a two simple shapes on a computer screen. One group of subjects was told that the movements were generated by two people playing a game, and the other group was told (correctly) that the movements were randomly generated. Both groups then segmented the actions. Results indicated that the segmentations were predicted by a number of basic movement features such as direction changes and the mean proximity of the two objects. However, these basic movement features predicted segmentations most strongly when subjects believed that the movements were random. According to the researcher, this occurred because subjects in the person condition focused more on abstract conceptual goals and less on specific movement features than subjects in the random condition.

To explore the features that might predict action segmentations in a more ecological context, we completed an analysis of segmentations for a wide range of realistic actions in which a set of human models was videotaped completing a series of ten different tasks with a range of objects [113]. Instead of using basic movement features to predict segments, we defined a set of more meaningful subactions that were hand coded. These included hand-to-object contacts, object-to-object contacts, occlusions, and eye movements. We found that multiple regressions based on these subactions predicted up to 82% of the variance in the number of breakpoints entered (by eight judges) in each one-second bin.

The tasks being performed involves a person sitting at a table with a set of objects and performing some type of assembly task. The person is wearing a red glove on their right hand and a purple glove on their left hand. They are also wearing a hat with a lime-green strip down the center. The camera is situated in front of the person facing them from across the table.

The data collected is 100 videos (10 participants doing 10 tasks each). The tasks are various assembly and sorting types:

Task 1: The assembly of 3 flashlights. The flashlights are fully dismantled with the batteries, bulb and handle in 3 separate groups. The participant must construct all 3 flashlights and lay them on the table in front of the camera.

Task 2: The assembly of 3 item baskets. The basket, lid, tissue paper, and green Legos are in their own groups. There is also a stamp block. The user must construct a basket by placing a Lego, tissue paper and the lid in that order. The basket is finalized by stamping it with the stamp block. This is to be repeated for the next two baskets.

Task 3: The assembly of pipe structure. Four cylinder shaped pipes and two junction pipes are connected in a particular manner. The junction pipes have 3 openings that a cylinder pipe can fit. All the pipes must be used to construct a structure.

Task 4: The sorting and filling of containers. Six containers are stacked on top of each other and must be rearranged in a particular order with Lego blocks placed inside each one in a particular order.

Task 5: The filling of containers with Legos. Three yellow containers are to be filled with one color of Legos. The pile of Legos consist of 3 different colors and are all piled together to the right of the participant. The three containers are placed in front of the participant.

Task 6: The removal of Legos from containers and storing into another container. Three containers are located side by side. The center container is empty and the two periphery containers contain Legos to be moved to the center container.

Task 7: Occlusion Movement. Two Legos and a occluding object are on the table. The Legos are to be moved behind the occluding object, then moved to other sided of the occluding object.

Task 8: Occlusion Assembly. A T-shaped structure is created from Legos in plain sight of camera. Then, another T-shaped structure is created behind the occluding object. Finally, the occluding object is moved away.

Task 9: Assembly of 3 T-shaped structures. The different-colored T-shaped Lego structures are constructed in plain view of camera.

Task 10: Lego Stacking. Legos are to be stacked until all of the Legos are used, or the structure collapses. There are two attempts at this task.

A new feature vector is created to analyze the movement of the person in the video. The current system assumes that there are 2 actuators and 1 gaze estimator in the video. With these groups being classified by the user, the features of the video are extracted. The feature vector can vary depending on the number of frames (bin size) that it must represent. The 12 features are:

Magnitude velocity of Hand 1 - This value is calculated by taking the centroid values of hand 1 between successive frames and calculating the Euclidean distance between them. These distances are averaged across varying binsizes.

Velocity Stop of Hand 1 – This is a binary value that is decided by if the mean velocity of hand 1 is less than 1.5 pixels between successive frames of that bin (1 if true, 0 if false).

Object Contact of Hand 1 – This is a binary value that is decided by drawing a line between the centroid of an object and the centroid of hand 1. If the number of pixels that are not classified as the hand or the object in question is less than 2 for all object groups in the image, then hand 1 is considered near an object. (1 if true, 0 if false).

Hand 1/ Object Change – This is an average of the change in the number of pixels represented by objects when hand 1 is within 30 pixels of them.

Magnitude velocity of Hand 2 - This value is calculated by taking the centroid values of hand 2 between successive frames and calculating the Euclidean distance between them. These distances are averaged across varying bin sizes.

Velocity Stop of Hand 2 – This is a binary value that is decided by if the mean velocity of hand 2 is less than 1.5 pixels between successive frames of that bin (1 if true, 0 if false).

Object Contact of Hand 2 – This is a binary value that is decided by drawing a line between the centroid of an object and the centroid of hand 2. If the number of pixels that are not classified as the hand or the object in question is less than 2 for all object groups in the image, then hand 1 is considered near an object. (1 if true, 0 if false).

Hand 2/ Object Change – This is an average of the change in the number of pixels represented by objects when hand 2 is within 30 pixels of them.

Gaze Velocity – This is the mean velocity of the estimated gaze angle change across a bin. (Note: Gaze is estimated by using a stripe on the participant's hat. The angle is calculated by estimating the angle between the best fit line for the points of the stripe and the vertical line between the centroid of the stripe)

Gaze Object - This is a binary value that is decided by if the gaze angle is within 10 degrees of an object for at least half the frames of a bin (1 if true, 0 if false).

Gaze Hand – This is a discrete value with possible values of {0, 1, 2, 3}. This value is determined if the gaze estimation is within 10 degrees of :

None of the Hands – yields a value 0

Hand 1 alone – yields a value 1

Hand 2 alone – yields a value 2

Both of the Hands – yields a value 3

Gaze Stop - This is a binary value calculated by if the gaze velocity of a bin is less than the mean gaze velocity of the entire video (1 if true, 0 if false).

All of the features are dependant on bin-size. The videos are recorded with a frame rate of about 30 frames per second. A group of feature vectors are calculated for bin-sizes of 1, 3, 6, 8, 10, and 20 frames. The videos are hand segmented to determine significant moments during the task. The moments are defined as hand grasps and releases of objects throughout the task. A bin is considered a significant bin if an significant frame designated by the human rater falls within that bin. The significant moments were based on the findings of the Psychology Department which were hand grasps and hand releases of the participant. The bins are created by sequential sets of bin-size frames with no overlap (e.g. for bin-size 3, the first vector will consist of frames 1-3, the second will consist of frames 4-6, and so on).

Experiment 1: Task/Breakpoint Segmentation

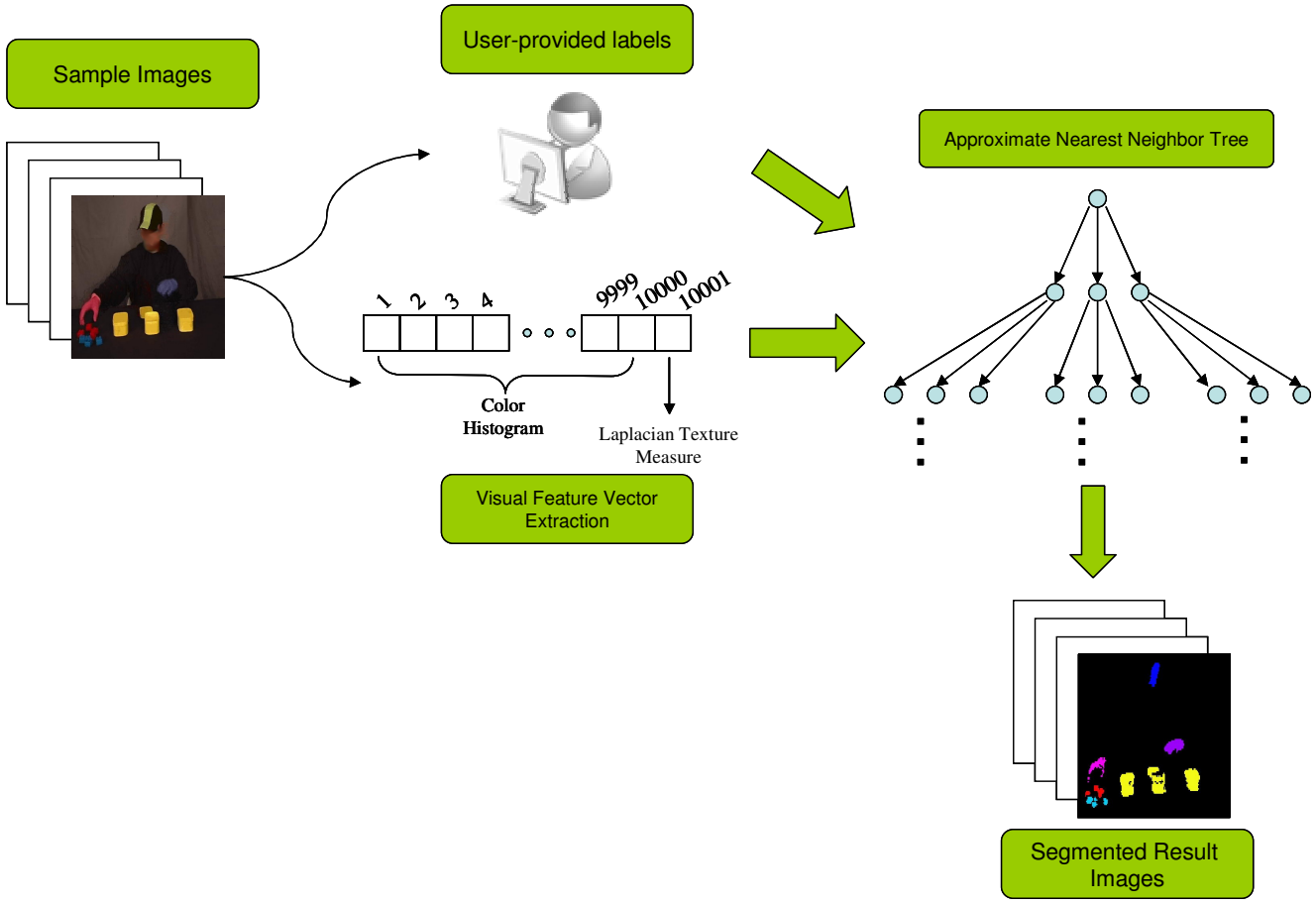


Figure 45: Supervised Video Segmentation Flow Chart

Figure 45 shows the general flow to segmenting the videos. Using the capabilities of our previous system, the user will provide labeled samples of every object in the video which involves sorting through multiple frames to provide ample sampling of each object throughout the time period. Due to the lighting differences during filming, caused by incandescent bulbs lighting the room and shadowing, as well as video format conversion information loss, the objects can have quite a varied look throughout the span of the video. Once all of the training data is collected and labeled, an approximate nearest-neighbor tree is constructed to be used to efficiently label new data. This tree is applied to each frame of the video and the resulting image is stored. All the resulting images are used to create the segmented video.

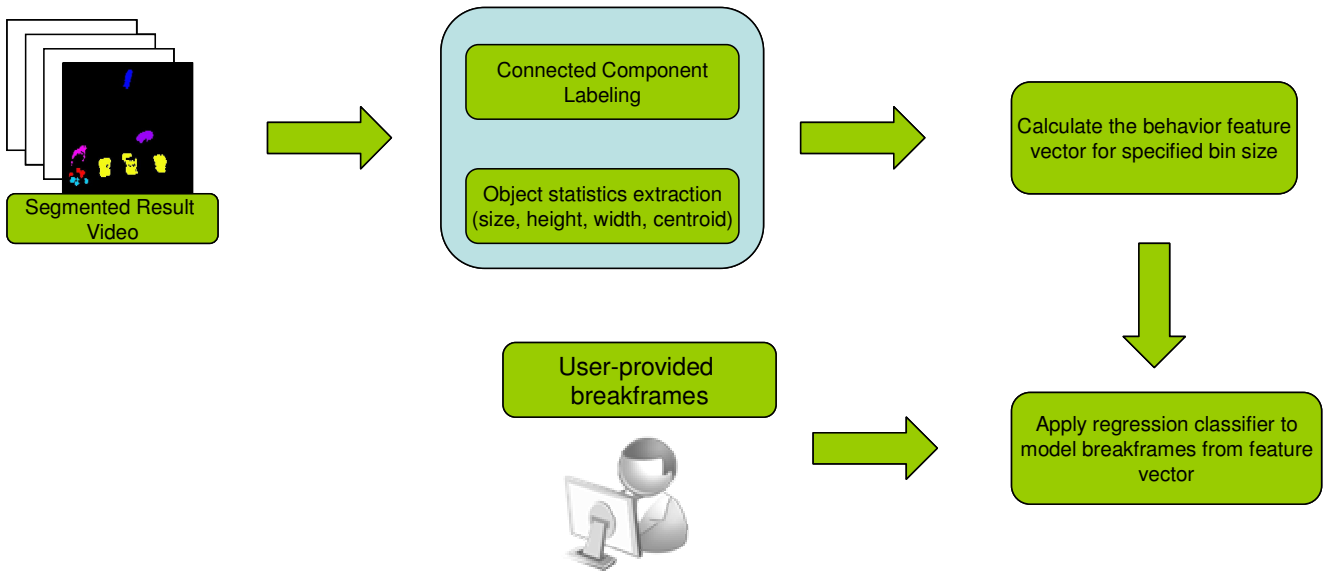


Figure 46: Supervised Behavior Extraction Flow Chart

The first important step toward task recognition and identification is to identify the crucial segments of a set of actions (a flowchart shown in Figure 46). The first goal is to provide a set of boundary points to train the breakpoint segmentation system, analyze the success of the features to predicting breakpoints, and compare the segmentations of the raters. The video frames were marked as the boundaries of key subevents occurred as defined by their importance in describing the steps

needed to complete the task. These moments are chosen to be at the finest resolution of the motions in the task. Selection of these subevents reflected the findings of the study mentioned earlier [113] where the segmentation boundaries corresponded to hand-to-object contact and object-to-object interactions with gaze confirmation (using participant gaze to disambiguate the model's current focus of attention). A frame was selected as a significant frame if there was hand to object contact; hand induced object to object contact, or releasing of an object. In many cases, key events extended over multiple frames. For example, the tasks often require combination of objects. These combinations require the contact of two objects and applying force to squeeze them together. During the moment of the hands holding the objects and applying the force, all frames depicting this event were marked for the subevent. Depending on the task, the participant, and the bin size, the number of marked bins ranged between one-third to one-half of the total bins in the video. Once the breakpoint feature vectors are identified, these vectors are used to train linear, quadratic, and Mahalnobis regression models. The explanation and results of this experiment are also available via our paper [116].

The measure used to determine performance is called d-prime (D'). This value is calculated by using the:

$$D' = f^{-1}(hit\_rate) - f^{-1}(false\_alarm) \tag{68}$$

where f(x) is the cumulative sum of a normal distribution of $\mu = 0$ and $\sigma = 1$

$$f(x) = \int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) dx \tag{69}$$

f^{-1} returns the x value that corresponds to the moment the equation f(x) meets a given value.

The hit rate is defined as the probability of the system correctly identifying the interesting bins which is calculated by taking the number of correctly identified interesting bins divided by the number of total interesting bins.

The false alarm is defined as the probability of the system misclassifying the non-interesting bins which is calculated by taking the number of misclassified non-interesting bins divided by the number of total non-interesting bins.

The norminv calculates the x value given a probability, mean, and standard deviation on the cumulative sum of a normal curve. This measure is basically a non-linear measure of the distance between hit rate and false alarm. An interesting fact about this measure is how the norminv approaches infinity as the input value approaches 1 or negative infinity approaching 0. In the case of a perfect hit rate, the d' value is infinity. In the cases for this experiment, it has been shown that the d' of infinity identifies a classifier that has an excessively high false alarm rate as well. In Table 5, there are 2 d' measures, d_1' and d_2', that are calculated. d_1' takes the average hit rate and the average false alarm rate of all then calculates the d' from that value. d_2' is calculated by taking the individual d' for all the available hit rates and corresponding false alarm rates, and taking the average of all the d' values. Since d_2' has the possibility of containing infinity values, those individual d' values are replaced with a value of 0.5 to represent maximum uncertainty. This measure is the average of the d' values for each individual test set.

The data being analyzed is in the format of 100 videos (10 participants doing 10 tasks each). Each video has its set of behavior feature vectors calculated for each of the bin-sizes for analysis. Various regression models are trained and tested on the data to determine the best methods using the d' measure. The steps to perform this experiment are as follows (in Table 4):

Table 4: Experiment 1 Roadmap

Procedure
Gather the 10 task videos for each of the 10 participants.
Identify each video with human rated significant frames.
Train the visual database of the system on each of the videos of a particular task.
Segment each video and store the object data for each frame.
Extract behavior features for all videos with each bin-size allocation.
Train the classification system with labeled behavior data.
Analyze data using regression techniques.
Refine significant moment detection to reduce the false alarm rate.

Overall, predictions of subevents based on the movement and contact variables were moderate, and strongest for 6 frame bins using a linear classifier (as can be shown by Table 5, Table 6, and Figure 47). In fact, the 6 frame bins performed the highest for each of the 3 classifiers in both the task analysis and subject analysis. To estimate the maximum success of the system in this analysis, the data was analyzed with the k-nearest neighbor method since this method converges to the MLE results. The k number of neighbors was incrementally increased by 50 to a group of 10001. Since the total amount of vectors created for the top bin-size of 6 for the entire data set was 22,862 , this max k value would sufficiently capture the maximum d'. Analysis shows d_1' increases dramatically then saturates at a value of about 1.400 with the value k around 950 nearest neighbors.

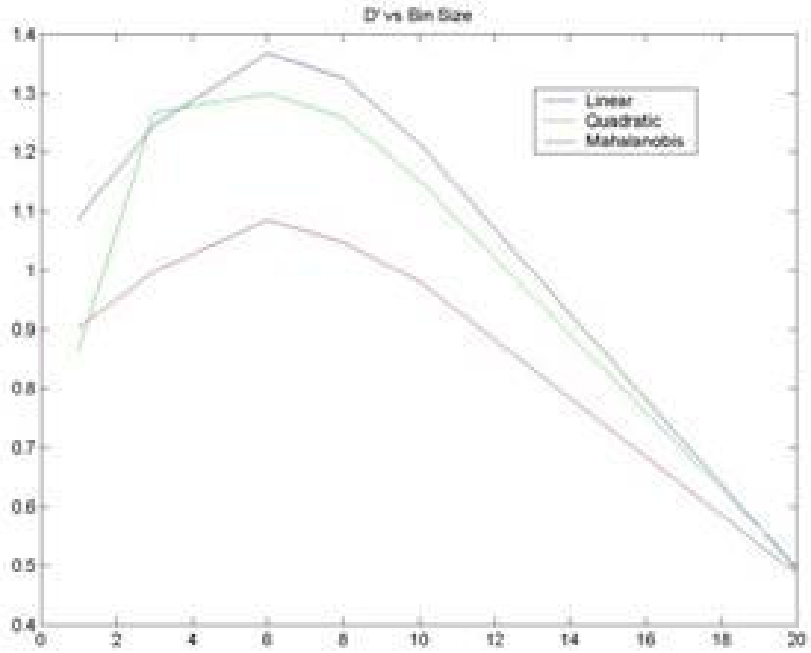


Figure 47: d' vs Bin Size

Table 5: Comparison of Bin Size and Regression Models using Subject Jack Knife

	Linear				Quadratic				Mahalanobis			
Binsize (frames)	HR (μ, σ)	FAR (μ, σ)	d_1'	d_2' (μ, σ)	HR (μ, σ)	FAR (μ, σ)	d_1'	d_2' (μ, σ)	HR (μ, σ)	FAR (μ, σ)	d_1'	d_2' (μ, σ)
1	(0.869, 0.060)	(0.513, 0.088)	1.088	(1.124, 0.294)	(0.195, 0.163)	(0.042, 0.065)	0.864	(0.886, 0.460)	(0.378, 0.119)	(0.113, 0.048)	0.903	(0.921, 0.265)
3	(0.839, 0.077)	(0.399, 0.090)	1.247	(1.309, 0.357)	(0.842, 0.078)	(0.396, 0.088)	1.267	(1.325, 0.327)	(0.478, 0.131)	(0.146, 0.062)	0.998	(1.038, 0.321)
6	(0.859, 0.079)	(0.385, 0.011)	1.368	(1.411, 0.385)	(0.819, 0.083)	(0.349, 0.096)	1.300	(1.362, 0.338)	(0.488, 0.129)	(0.132, 0.067)	1.085	(1.131, 0.332)
8	(0.843, 0.092)	(0.375, 0.126)	1.326	(1.396, 0.358)	(0.792, 0.097)	(0.328, 0.107)	1.259	(1.332, 0.375)	(0.516, 0.132)	(0.157, 0.077)	1.047	(1.109, 0.385)
10	(0.804, 0.114)	(0.360, 0.134)	1.215	(1.298, 0.385)	(0.738, 0.108)	(0.303, 0.105)	1.152	(1.210, 0.361)	(0.496, 0.131)	(0.161, 0.086)	0.982	(1.053, 0.424)
20	(0.626, 0.127)	(0.432, 0.211)	0.494	(0.472, 0.546)	(0.345, 0.128)	(0.185, 0.148)	0.500	(0.402, 0.400)	(0.395, 0.131)	(0.226, 0.155)	0.487	(0.435, 0.418)

Table 6: Comparison of Bin Size and Regression Models using Task Jack Knife

	Linear				Quadratic				Mahalanobis			
Binsize (frames)	HR (μ, σ)	FAR (μ, σ)	d_1'	d_2' (μ, σ)	HR (μ, σ)	FAR (μ, σ)	d_1'	d_2' (μ, σ)	HR (μ, σ)	FAR (μ, σ)	d_1'	d_2' (μ, σ)
1	(0.869, 0.067)	(0.516, 0.094)	1.081	(1.128, 0.295)	(0.355, 0.290)	(0.136, 0.154)	0.727	(0.890, 0.461)	(0.391, 0.123)	(0.122, 0.051)	0.887	(0.906, 0.242)
3	(0.836, 0.085)	(0.399, 0.099)	1.232	(1.305, 0.368)	(0.841, 0.079)	(0.398, 0.094)	1.257	(1.317, 0.323)	(0.475, 0.125)	(0.147, 0.064)	0.988	(1.028, 0.335)
6	(0.857, 0.084)	(0.387, 0.119)	1.356	(1.410, 0.374)	(0.820, 0.085)	(0.353, 0.104)	1.292	(1.356, 0.341)	(0.487, 0.123)	(0.136, 0.071)	1.063	(1.122, 0.356)
8	(0.842, 0.095)	(0.378, 0.134)	1.313	(1.391, 0.343)	(0.789, 0.097)	(0.330, 0.112)	1.244	(1.314, 0.359)	(0.508, 0.129)	(0.161, 0.083)	1.008	(1.071, 0.391)
10	(0.803, 0.119)	(0.364, 0.142)	1.199	(1.275, 0.395)	(0.738, 0.112)	(0.307, 0.112)	1.143	(1.206, 0.358)	(0.491, 0.128)	(0.163, 0.088)	0.960	(1.026, 0.435)
20	(0.629, 0.134)	(0.436, 0.227)	0.491	(0.480, 0.524)	(0.337, 0.113)	(0.192, 0.149)	0.450	(0.395, 0.440)	(0.379, 0.117)	(0.227, 0.150)	0.443	(0.405, 0.448)

To assess the degree to which our 12 predictor variables can be represented by a smaller number of more basic factors, we performed a principle components analysis. We wanted to determine if there was a possibility to achieve performance closer to the MLE performance using a subset of the features presented. First, the entire database is thinned out. The thinned data are the points that have the smaller distances from its nearest neighbor. The thinned data vectors are about half in number compared to the full data set. Fisher's linear discriminant analysis is applied to the thinned data as well as a principal component analysis (PCA). By calculating the eigenvalues and eigenvectors of that cross correlation method, the top 3 eigenvalues that caused the most variance in the data were identified. The eigenvectors corresponding to these eigenvalues were applied multiplied to the data and plotted. Four distinct groups could be seen, each with interesting points tightly clustered and non-interesting points trailing outward as seen by Figure 48.

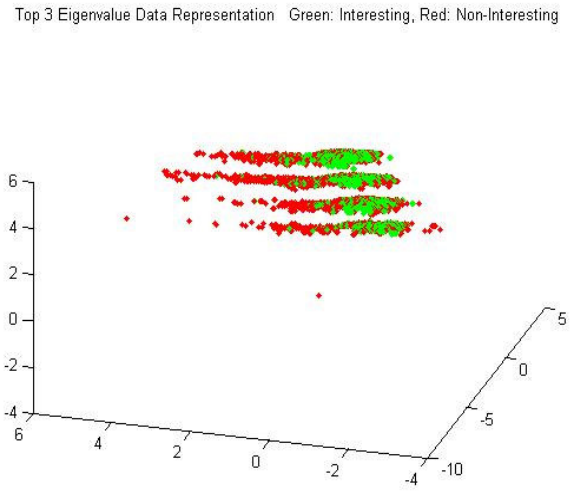


Figure 48: Top 3 Eigenvalue Data Representation for PCA

It was decided that further analysis of the feature combinations were to be examined. Exhaustive analyses of all combinations of features (up to 6 total features) were examined in predicting the key moments and the top 5 feature sets were calculated. The purpose of doing this analysis was to determine if a subset of the features used would provide as good or better results from the use of all features. Table 10 and 11 has the same format as Table 2 and 3 except showing the

statistics of each technique's 1st – 5th best feature combination results for the subject jack knife and task jack knife. Tables 7-12 show the top 5 feature combinations for each of the regression methods for the subject and task analysis.

The most prominent features for the linear set are 1, 2, 5, and 8. These 4 features are found in all instances of the top 5 feature sets. Features 1 and 5 are the velocities of the two hands, feature 2 correspond to the binary feature for stopped hand 1 motion, and feature 8 corresponds to the amount of pixel change around hand 2. In the linear regression case, these features embody the information needed of the hand. Notice that the best feature sets also involve gaze information (feature 10) or rather the gaze toward an object.

Table 7: Top 5 Feature Sets for Linear Analysis using Subject Jack Knife

	Top 5 Feature Sets for Linear Analysis					
1st	10	8	5	4	2	1
2nd	8	7	5	4	2	1
3rd	9	8	7	5	2	1
4th	9	8	5	2	1	-
5th	8	5	2	1	-	-

Table 8: Top 5 Feature Sets for Linear Analysis using Task Jack Knife

	Top 5 Feature Sets for Linear Analysis					
1st	10	8	5	4	2	1
2nd	8	6	5	4	3	1
3rd	10	8	6	5	4	1
4th	5	2	1			
5th	8	6	5	4	1	-

The quadratic results are very similar except using even fewer features. Features 1 and 5 are necessary in every instance for the top 5 sets and feature 12 adds the gaze information needed for the top set of features in the subject jack knife analysis.

Table 9: Top 5 Feature Sets for Quadratic Analysis using Subject Jack Knife

	Top 5 Feature Sets for Quadratic Analysis					
1st	12	7	5	1	-	-
2nd	12	11	7	5	1	-
3rd	5	1	-	-	-	-
4th	7	5	1	-	-	-
5th	5	3	2	1	-	-

Table 10: Top 5 Feature Sets for Quadratic Analysis using Task Jack Knife

	Top 5 Feature Sets for Quadratic Analysis					
1st	5	1	-	-	-	-
2nd	5	3	2	1	-	-
3rd	5	2	1	-	-	-
4th	12	11	7	5	1	-
5th	11	5	1	-	-	-

The Mahalanobis results show a high tendency toward the binary or discrete values. This measure focuses on the hand stop features (2 and 6) but also uses the gaze angular velocity (feature 9) and gaze toward objects (feature 10) to perform at its highest capacity.

Table 11: Top 5 Feature Sets for Mahalnobis Analysis using Subject Jack Knife

	Top 5 Feature Sets for Mahalanobis Analysis					
1st	10	9	8	6	3	2
2nd	10	9	8	7	6	2
3rd	10	9	8	6	2	-
4th	9	7	6	4	2	-
5th	9	6	4	2	-	-

Table 12: Top 5 Feature Sets for Mahalanobis Analysis using Task Jack Knife

	Top 5 Feature Sets for Mahalanobis Analysis					
1st	2	3	4	6	9	-
2nd	2	4	6	9	-	-
3rd	2	3	6	8	-	-
4th	2	3	6	4	-	-
5th	2	6	8	-	-	-

The velocity features correlate with grasping since a majority of the grasps require a pause in the hand motion. This same reasoning also explains the correlation between the binary hands stopped features as well. Since grasps occur when the velocity of the hand is low and low gaze motions indicate focusing on an action, these findings support the findings of the Psychology Department analysis of a high correlation between significant moments in the task with hand grasps and gaze. The top 5 Feature Sets for each of the cases show that all the regression cases have comparable performance measures to the MLE d' performance of 1.400 as seem in Tables 13 and 14 . The quadratic provides a slightly better that the estimator for the data in this experiment.

Table 13: Top 5 Feature Set Results for Subject Jack Knife

	Linear				Quadratic				Mahalanobis			
Top 5 Results	HR (μ, σ)	FAR (μ, σ)	d_1'	d_2' (μ, σ)	HR (μ, σ)	FAR (μ, σ)	d_1'	d_2' (μ, σ)	HR (μ, σ)	FAR (μ, σ)	d_1'	d_2' (μ, σ)
1st	(0.891, 0.079)	(0.435, 0.134)	1.397	(1.461, 0.391)	(0.908, 0.069)	(0.458, 0.134)	1.431	(1.486, 0.424)	(0.875, 0.082)	(0.407, 0.128)	1.388	(1.439, 0.386)
2nd	(0.889, 0.076)	(0.432, 0.132)	1.391	(1.460, 0.371)	(0.903, 0.070)	(0.450, 0.125)	1.427	(1.458, 0.426)	(0.875, 0.083)	(0.406, 0.128)	1.388	(1.439, 0.386)
3rd	(0.886, 0.080)	(0.426, 0.136)	1.391	(1.467, 0.397)	(0.917, 0.066)	(0.482, 0.132)	1.426	(1.466, 0.437)	(0.875, 0.082)	(0.406, 0.128)	1.388	(1.438, 0.385)
4th	(0.886, 0.080)	(0.426, 0.136)	1.390	(1.465, 0.395)	(0.913, 0.065)	(0.474, 0.132)	1.426	(1.482, 0.428)	(0.875, 0.083)	(0.406, 0.128)	1.388	(1.438, 0.386)
5th	(0.886, 0.079)	(0.427, 0.136)	1.390	(1.465, 0.395)	(0.904, 0.074)	(0.453, 0.130)	1.426	(1.457, 0.450)	(0.875, 0.083)	(0.406, 0.128)	1.388	(1.438, 0.386)

Table 14: Top 5 Feature Set Results for Task Jack Knife

	Linear				Quadratic				Mahalanobis			
Top 5 Results	HR (μ, σ)	FAR (μ, σ)	d_1'	d_2' (μ, σ)	HR (μ, σ)	FAR (μ, σ)	d_1'	d_2' (μ, σ)	HR (μ, σ)	FAR (μ, σ)	d_1'	d_2' (μ, σ)
1st	(0.888, 0.082)	(0.435, 0.139)	1.382	(1.455, 0.386)	(0.914, 0.071)	(0.482, 0.137)	1.412	(1.453, 0.439)	(0.875, 0.083)	(0.406, 0.128)	1.388	(1.439, 0.386)
2nd	(0.858, 0.090)	(0.380, 0.128)	1.378	(1.446, 0.384)	(0.901, 0.080)	(0.453, 0.135)	1.408	(1.450, 0.450)	(0.875, 0.083)	(0.406, 0.128)	1.388	(1.439, 0.385)
3rd	(0.859, 0.092)	(0.382, 0.128)	1.377	(1.450, 0.384)	(0.900, 0.077)	(0.450, 0.134)	1.407	(1.463, 0.423)	(0.874, 0.083)	(0.405, 0.129)	1.387	(1.439, 0.386)
4th	(0.882, 0.093)	(0.426, 0.151)	1.373	(1.452, 0.411)	(0.900, 0.076)	(0.450, 0.130)	1.407	(1.454, 0.423)	(0.874, 0.083)	(0.405, 0.129)	1.387	(1.439, 0.386)
5th	(0.858, 0.091)	(0.383, 0.129)	1.372	(1.442, 0.381)	(0.909, 0.071)	(0.472, 0.133)	1.406	(1.447, 0.422)	(0.874, 0.083)	(0.405, 0.129)	1.387	(1.439, 0.386)

So far we have shown that the system provides good results in the d' measures, but the main shortcoming of getting even better scores is the high false alarm rates. It was decided to look at where the false alarms were occurring to determine where the false alarms are occurring. Another interesting point from our data is in the distribution of the false alarms and the misses across the number of bin offsets. Of the entire database of 22862 vectors, 38.5% (4921 vectors) were labeled as false alarm in the 6 frame bin-size linear regression case as seen in Figure 49. Of the false alarms, 61% were within one bin of a correct classification which suggests that by adding a tolerance of one bin to the bin classifications, the false alarms could be reduced to around 14% and two bins would reduce the false alarms to around 7%. These additional tolerances would be logical since the human rated classifications carry some ambiguity in the segmentation boundaries. Similarly, the hit rate could be

raised by applying a similar tolerance of one or two bins in the case of the misses (~14%, 1412 vectors) as seen in Figure 50. A tolerance of one bin would reduce the miss rate to around 4% which increases the hit rate to about 96%.

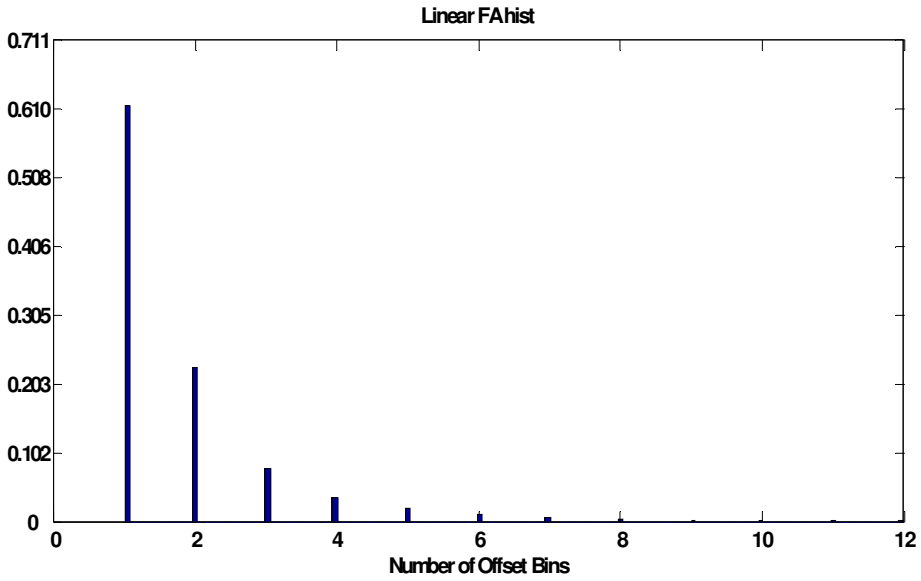


Figure 49: Percentage of False Alarms per Number of Offset Bins

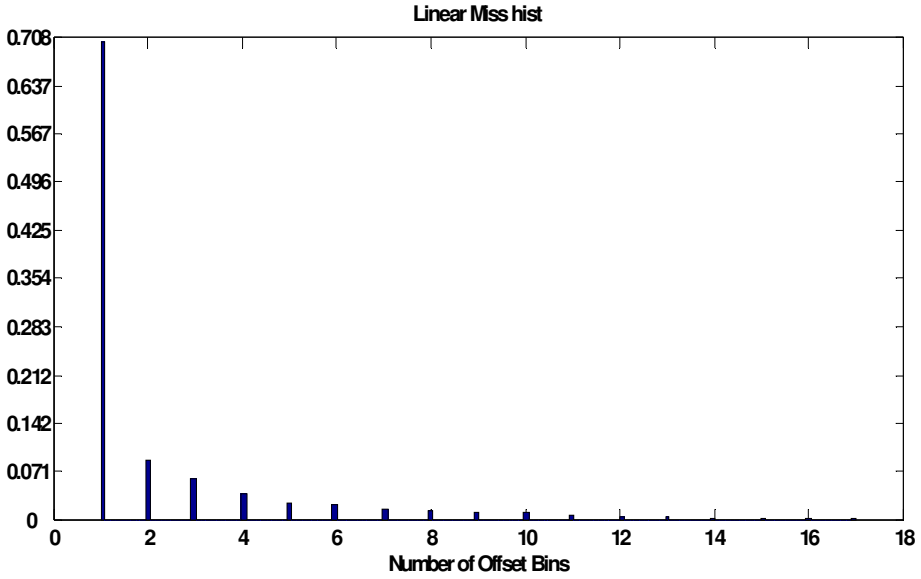


Figure 50: Percentage of Misses per Number of Offset Bins

One of the shortcomings of this experiment is the lack of data available from various raters for use of training the system. The intentional vision research from the Psychology Department provided

breakpoint segment analysis from 2 sources, the original rater (psychology graduate student) and various interrater participants (undergraduate volunteers). The original rater was required to segment each of the videos while the interraters were required to segment one video from each of the performing participants. The two ratings were compared to each other to show that 82% of the variance was accounted for using one second bins. The manner of the ratings that the original rater uses is fairly different from the ratings created to train the system (system rater – the author), but it is necessary that the ratings that train the system have a high correlation with ratings from the original and interrater ratings. Once the system is trained on the system rater breakpoints, the breakpoints developed by the system will be the system estimates. In Newtson's work [117], he established a method of comparing two sets of data to determine the correlation between them. The method basically calculated the probability of overlap when comparing a coarse and a fine set of data as the following:

$$P(overlap) = P(coarse) \times P(fine) \times Bins \tag{70}$$

The method itself will not work directly between the interrater ratings, original ratings, and system training ratings, but the overall concept still applies. The correlation between the sets of data with different amounts of classified samples can be seen as the amount of separation between the probability of the number of overlaps (breakpoints in same position) and the probability of randomly choosing overlap positions. One fact to note is the system breakpoints must be converted to the format of the original breakpoints as shown in Figure 51. The original and interrater breakpoints are given as a set of times that correspond to moments in the video sequence. The system rater breakpoints are a sequence of ones (breakpoints) and zeros (not breakpoints) that span the video sequence. The video is broken into 6 frame bins (as determined by the optimal settings in Tables 2 and 3). The system rater breakpoints were designed to extend across a range of bins for as long as the action occurred. The original breakpoints only specified a single time allocation for an action. Since the original and

interrater ratings are given in seconds, the times are converted to frames. Then, the frames are represented with a 1 or 0 in the appropriate 6 frame bin.

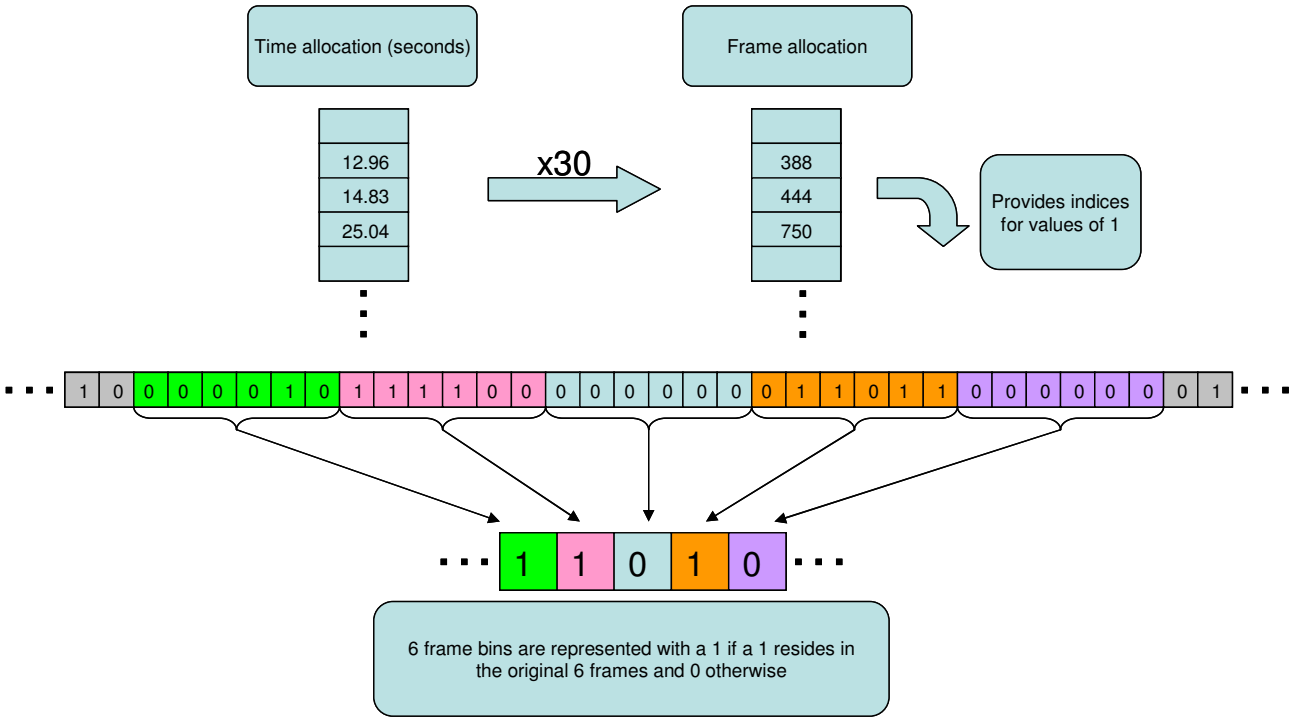


Figure 51: Converting time breakpoints to 6 frame bin format

Since the system rater breakpoints are ranges of ones meant to capture all representations of the action, these ranges must be converted to an appropriate representation resembling the format of the original and interrater ratings. This single original breakpoint can be assumed to occur in the proximity of the middle of the corresponding sequence of breakpoints for the same action that the system rater breakpoints specify. To convert the system breakpoints to the format of the original breakpoints, all consecutive sequence of breakpoints are converted to a single breakpoint located in the center of the span of time. This conversion is done for both the system rater and the system estimate in the following analysis.

Once all the data is in the same format, the method of comparison to determine correlation is as follows. One rater's vector is set as the base vector and another rater's vector is set as the test vector. The base vector is treated as the ground truth for the comparison. Since the objective for the

experiment is to develop a system that can replicate human breakpoints chosen by the psychology students and volunteers, the priority of base truth goes as follows: original rater > interraters> system rater> system estimate. The probability of randomly choosing a bin that contains a breakpoint in the base vector (probability of base) is simply the # of breakpoint bins divided by the total number of bins. When determining the accuracy of overlap, each breakpoint bin in the test vector is compared to the base vector at the same position. If there exists a breakpoint in the base vector within a certain variability (variability ranges from 0 – 2, as seen in Figure 52), then the breakpoint is considered to overlap in both the test and base vectors. These overlaps are counted and divided by the total number of possible breakpoints in the test vector to attain the accuracy of overlap. If the data has some correlation, then the accuracy of overlap should be more than the probability of breakpoint. As bin variability increases, the probability of breakpoint also increases. If the data sets are truly correlated, then the accuracy of overlap should have a greater increase if not equal to the probability of breakpoint increase.

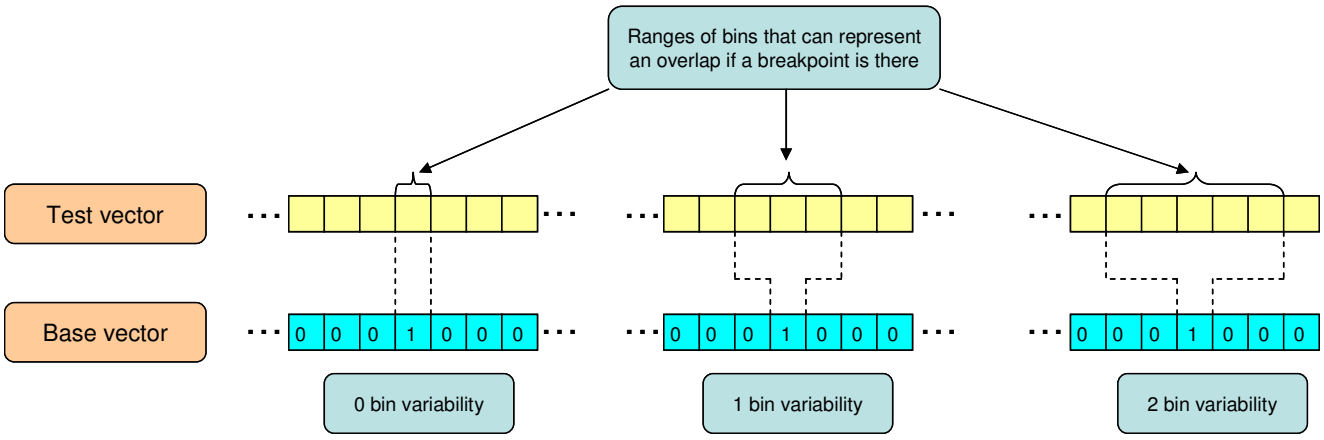


Figure 52: Bin variability

As can be seen in Tables 15 -17, the probabilities show that there is correlation between system and the original breakpoints. The correlation is not apparent for any of the comparisons except the interraters and original with 0 bin variability. With 1 bin variability, the difference between the probability of base and accuracy of overlap grows. The accuracy grows even more when allowed the

full 1 second bin (30 frames) with 2 bin variability. This confirms that the system rater and the system estimate, which are meant to represent the features that the intentional vision group found, are related to the human breakpoint selections.

Table 15: Correlation with 0 bin variability with bin-size of 6

Test vs Base	Probability of Base	Accuracy of Overlap
Interrater vs Original	0.271	0.562
System Rater vs Original	0.288	0.266
System Estimate vs Original	0.288	0.269
System Rater vs Interrater	0.251	0.256
System Estimate vs Interrater	0.250	0.266
System Estimate vs System Rater	0.157	0.169

Table 16: Correlation with 1 bin variability with bin-size of 6

Test vs Base	Probability of Base	Accuracy of Overlap
Interrater vs Original	0.596	0.932
System Rater vs Original	0.646	0.788
System Estimate vs Original	0.646	0.792
System Rater vs Interrater	0.582	0.751
System Estimate vs Interrater	0.582	0.737
System Estimate vs System Rater	0.463	0.753

Table 17: Correlation with 2 bin variability with bin-size of 6

Test vs Base	Probability of Base	Accuracy of Overlap
Interrater vs Original	0.742	0.955
System Rater vs Original	0.797	0.895
System Estimate vs Original	0.797	0.898
System Rater vs Interrater	0.741	0.861
System Estimate vs Interrater	0.741	0.854
System Estimate vs System Rater	0.670	0.937

To the increase of overlap as bin variability increases, this analysis is to determine if there is a consistent lag or lead to the breakpoints determined by the system rater when compared to the breakpoints of the interraters and the original raters. In Table 118, the mean and standard deviation of the offsets are posted for the comparisons between the interraters, original rater and the system rater. The same dynamic of using the test vectors and base vectors are used with the hierarchy remaining the same as before. The nearest breakpoint in the base vector is found for each breakpoint in the test vector. The number of bins either leading (represented as negative) or lagging (represented as positive) are collected for each breakpoint. If there is an equal number of bins leading and lagging a particular breakpoint, then the breakpoint is assumed to be leading and a counter is incremented to count the number of occurrences (# of equal lead/lag). The mean and standard deviation are calculated from the list of bin offsets. As can be seen from Table 15, there is no consistent lead or lag that can be applied to provide a significant increase in correlation.

Table 18: Overall Lead/Lag Breakpoint Analysis

Test vs Base	Offset (μ, σ)	# of Equal Lead/Lag	# of Non-overlap b.pts.
Interrater vs Original	(-0.146, 2.493)	50	369
System Rater vs Original	(-2.098, 18.111)	342	1724
System Rater vs Interrater	(-0.319, 2.898)	28	192

Experiment 2: Participant Identification

One interesting question is whether the action behaviors can be used to determine participant identities. The next step after establishing reliable segmentations of actions is to determine the actual movements and identify the subject performing the task. By studying these motions, it will be interesting to determine if there is enough data to specifically identify one subject from another. Another interesting question is linked to studies done in psychology and neuroscience fields concerning consistency detection in humans. Zacks et al.[118-119] also determined that when people watch events and naturally break them into discrete sections. Recognition and identification occur more quickly and accurately with the development of better predictions toward the next state of actions. Observing patterns in the sequences of actions learns predictions. This leads to the idea that once actions are learned, a person applies deviation detection to determine inconsistencies in a new example of the performed task. Therefore, these inconsistencies should be available to detect outliers among the participant population. The following steps will be used to accomplish these goals.

Table 19: Experiment 2 Parameter Optimization Roadmap

Procedure for Parameter Optimization for Participants
Extract information about movements between segmentation bounds to create action feature vector.
Cluster action feature vectors with k-means.
Form symbols for each action using the clusters.
Train HMM for each participant by using all the tasks of a specific participant.
Run against data to identify participant.
Iterate number of groups (k) and number of states for the HMM to determine optimal settings.

We iterate through a k from 2-10 and a number of HMM states of 1-10. Once an optimal parameter set is found the same parameter set is used in the cross validation analysis.

Table 20: Experiment 2 Testing Roadmap

Procedure for Testing
▪ Train HMMs using jack knife and using random cross-validation methods o For the cross-validation, 7 random videos will be used to train per person in 2 ways ▪ First, the same 7 random task will be used for the training of each person's HMM ▪ Second, a new random set of 7 from a person's video set will be chosen for each person's HMM
Test HMMs against data not used in training.

The entire testing procedure will be repeated 20 times to provide an accurate assessment of the system's performance.

Table 21: Subject Cross Validation Accuracies

	Accuracy
1st Validation Method	0.1100
2nd Validation Method	0.1217

We hypothesize that the feature vectors which are designed for generic action parsing will not have very much information pertaining to a particular person. As can be seen from Table 21, the cross validation methods show that the behavior features used to segment actions do not contain enough information to dissociate amongst participants with the available data. These results are reasonable since tasks performed by the participants are fairly linear, allowing for little variation in the ways they

are to be accomplished. Since the steps performed amongst each participant for each task are virtually the same, it is logical that the system will not be able to detect specific participants without more data or a different set of features. Though participant identification across all 10 subjects is shown to not to be applicable for these features, the features definitely contain some information about participants that can be used to determine subsets of the participants that can be identified reliably. The first step to determining whether subsets of participants exist is to reduce the number of participants compared amongst each other. The data is broken up into all possible combinations of 2 participants up to 5 participants. The data from the combination of participants are compared to the HMM models of each participant to determine accuracy.

Table 22: Participant Combination Analysis using Cross Correlation Analysis 1

Number of Participants	Overall Accuracy	Top 4 Accuracies	Top 4 Participant Combinations	Bottom 4 Accuracies	Bottom 4 Participant Combinations
2	0.5267	0.6917	14,11	0.4250	20,13
		0.6750	19,11	0.4167	17,15
		0.6417	17,14	0.3833	17,12
		0.6417	16,11	0.3250	19,18
3	0.3750	0.5778	15,14,11	0.2833	17,15,12
		0.5500	14,12,11	0.2722	20,19,15
		0.5111	17,16,11	0.2667	20,17,15
		0.5111	18,16,11	0.2500	19,18,11
4	0.2803	0.3917	15,14,13,11	0.1958	20,19,17,12
		0.3917	14,13,12,11	0.1958	20,19,18,15
		0.3875	19,15,14,11	0.1750	17,15,12,11
		0.3750	18,15,14,11	0.1667	19,17,13,12
5	0.2283	0.3200	18,17,14,12,11	0.1600	20,18,15,13,11
		0.3100	16,15,14,12,11	0.1600	20,18,16,14,13
		0.3067	18,15,14,12,11	0.1567	20,18,15,13,12
		0.3000	20,16,13,12,11	0.1533	20,18,17,15,13

Table 23: Participant Combination Analysis using Cross Correlation Analysis 2

Number of Participants	Overall Accuracy	Top 4 Accuracies	Top 4 Participant Combinations	Bottom 4 Accuracies	Bottom 4 Participant Combinations
2	0.5865	0.8583	17,16	0.4750	12,11
		0.8000	19,17	0.4583	20,11
		0.7917	17,14	0.4083	16,11
		0.6833	17.13	0.3750	15,11
3	0.3956	0.5667	17,14,12	0.2722	14,12,11
		0.5444	20,19,11	0.2444	20,12,11
		0.5389	20,17,14	0.2444	18,16,11
		0.5000	17,16,12	0.2444	15,12,11
4	0.2882	0.3750	20,17,15,13	0.2042	20,19,12,11
		0.3750	19,17,14,12	0.2000	18,16,12,11
		0.3708	19,17,16,13	0.1958	19,16,12,11
		0.3708	20,17,15,13	0.1958	20,15,12,11
5	0.2067	0.2700	20,19,17,14,13	0.1467	20,19,16,12,11
		0.2600	19,14,13,12,11	0.1433	20,17,16,12,11
		0.2567	19,18,17,15,14	0.1367	20,17,16,14,11
		0.2567	20,19,17,14,13	0.1267	20,18,17,12,11

For each of the smaller combinations of participants (Table 22-23), the models are able to give a slightly better than chance overall analysis of data. The bottom 4 combinations are displayed to determine any patterns seen among participants that are outliers. Participants 11, 12 and 16 seem to be fairly consistent in participating with the lowest accuracies of each set. These results reinforce there is very little information among the behavior feature vector to determine the identity of the participant performing the task for this set of data.

Experiment 3: Task Identification

Likewise, it will be interesting to determine if there is enough data to specifically identify one task from another. The behaviors identified provide information for task segmentation, but do they also contain information about the identity of the task itself. This experiment mirrors the Subject Identification in the steps that are followed:

Table 24: Experiment 3 Parameter Optimization Roadmap

Procedure for Parameter Optimization for Tasks
Extract information about movements between segmentation bounds to create action feature vector.
Cluster action feature vectors with k-means.
Form symbols for each action using the clusters.
Train HMM for a task by using all the videos of a particular task.
Run against data to identify task.
Iterate number of groups (k) and number of states for the HMM to determine optimal settings.

We iterate through a k from 2-10 and a number of HMM states of 1-10. Once an optimal parameter set is found the same parameter set is used in the cross validation analysis.

Table 25: Experiment 3 Testing Roadmap

Procedure for Testing
▪ Train HMMs using jack knife and using random cross-validation methods o For the cross-validation, 7 random videos will be used to train per task in 2 ways ▪ First, the same 7 random subjects' task performance will be used for the training of each task HMM ▪ Second, a new random set of 7 participants will be chosen for each task HMM
Test HMMs against data not used in training.

The entire testing procedure will be repeated 20 times to provide an accurate assessment of the system's performance.

Table 26: Task Cross Validation Accuracies

	Accuracy
1st Validation Method	0.3333
2nd Validation Method	0.2650

We hypothesize that the feature vectors which are designed for generic action parsing will have some information pertaining to a particular task, but not enough to be highly deterministic. As can be seen from Table 26, the cross validation methods show that the behavior features used to segment actions contain some information to dissociate amongst tasks with the available data. The system performs better than chance, but still is not highly accurate. This information implies that there is more information about the tasks held within the behavior features than there is pertaining to specific

participants. The tasks are analyzed in combinations to see if there is significant information about subsets of the tasks using the behavior features. The data is broken up into all possible combinations of 2 tasks up to 5 tasks. The data from the combination of tasks are compared to the HMM models of each task to determine accuracy.

Table 27: Task Combination Analysis using Cross Correlation Analysis 1

Number of Tasks	Overall Accuracy	Top 4 Accuracies	Top 4 Task Combinations	Bottom 4 Accuracies	Bottom 4 Task Combinations
2	0.6430	0.9083	7,1	0.4333	6,3
		0.8833	10,5	0.4333	5,1
		0.8750	10,1	0.4167	9,7
		0.8667	10,2	0.4000	4,3
3	0.4872	0.7111	10,5,3	0.3000	8,6,4
		0.7111	10,4,2	0.2889	10,9,7
		0.7111	7,4,2	0.2833	8,6,3
		0.6944	10,3,2	0.2500	5,2,1
4	0.3875	0.5542	10,7,4,2	0.2375	9,8,6,4
		0.5250	10,7,3,2	0.2333	9,8,7,6
		0.5208	10,7,5,3	0.2208	8,6,4,3
		0.5208	10,5,4,3	0.2125	9,8,6,3
5	0.3232	0.4400	10,7,5,3,2	0.2200	9,8,7,4,3
		0.4333	10,9,5,4,3	0.2100	10,9,8,4,3
		0.4333	10,8,3,2,1	0.2033	9,8,6,7,4
		0.4267	7,6,3,2,1	0.1933	8,7,6,4,3

Table 28: Task Combination Analysis using Cross Correlation Analysis 2

Number of Tasks	Overall Accuracy	Top 4 Accuracies	Top 4 Task Combinations	Bottom 4 Accuracies	Bottom 4 Task Combinations
2	0.6331	0.9083	10,5	0.4583	5,2
		0.8667	10,1	0.4250	6,3
		0.8667	8,2	0.4250	5,1
		0.8500	7,5	0.4083	8,6
3	0.4708	0.6833	7,3,2	0.3000	6,4,3
		0.6722	10,6,2	0.2889	8,4,3
		0.6722	10,4,2	0.2833	8,6,4
		0.6667	10,5,3	0.2833	6,4,3
4	0.3751	0.5250	10,6,3,2	0.2292	9,6,5,4
		0.5167	10,7,4,2	0.2292	9,6,4,3
		0.5125	10,5,3,2	0.2125	9,8,6,3
		0.5083	10,3,2,1	0.1917	9,8,6,4
5	0.3089	0.4367	10,9,4,3,2	0.2133	10,9,8,6,4
		0.4300	10,7,3,2,1	0.2133	8,6,5,2,1
		0.4067	10,6,5,3,1	0.2000	9,8,6,4,3
		0.4067	10,8,5,2,1	0.1967	8,6,4,3,1

As can be seen from Table 27-28, the overall accuracies for the task combination analysis in each case are above chance. The tasks the system seems to have the most difficulty dissociating from are tasks 9, 8, 7, and 3. Those tasks are in a majority of the task combinations for the bottom 4 of each combination analysis. Tasks 7 and 8 are both occlusion tasks that have significant portions of the activities hidden behind a blinder so confusion can be expected in the case of these two tasks. Task 3 is an assembly task of a set of pipes. Participants tended to neither assemble the pipes in any specific order nor keep the structure in any particularly consistent orientation. The model of this activity probably ranged across various other participants. The data shows that there is some information in task identification among these behavior features, but the information in its present form is not enough to be highly accurate.

Experiment 4: Autonomous Action Segmentation

The next experiment was to develop the system to autonomously train and segment action videos. The goal of this experiment was to create a system that will segment the percepts in a video desired by the user and identify significant moments to segment the actions. This feature will allow

use of the system in diverse disciplines with little to no training necessary. The steps for Experiment 4 can be seen in Table 26.

Table 29: Experiment 4 Roadmap

Autonomous Segmentation Procedure
Develop system to autonomously detect percepts that correspond with user desired objects.
Segment videos and record object information for all frames.
Extract behavior features for all videos at optimal bin-size.
Train the classification system with labeled behavior data.
Analyze data using regression techniques.
Compare results of autonomous moment segmentation to supervised segmentation.

Most of the procedure for experiment 4 mirrors the first experiment once the videos are segmented. The main difference is the method in which the videos are segmented which are the first 2 tasks from Table 29. In Experiment 1, the system was provided the percepts of the video via user identification. In Experiment 4, the system must collect its own visual data and segment the data into natural groups. The system must take these natural groups and determine what these objects are (i.e. identification of the actor body parts, background, pertinent objects, etc.). Once these steps are completed, the process falls into nearly the same process of Experiment 1 in terms of behavior analysis. Another observation to be noted throughout the discussion is the amount of time each process takes. Since the times were not measured explicitly, the estimated times are an approximation

of the maximum time it took for any of the databases. First, let's go through the methodology for providing the autonomously segmented images as shown in Figure 53.

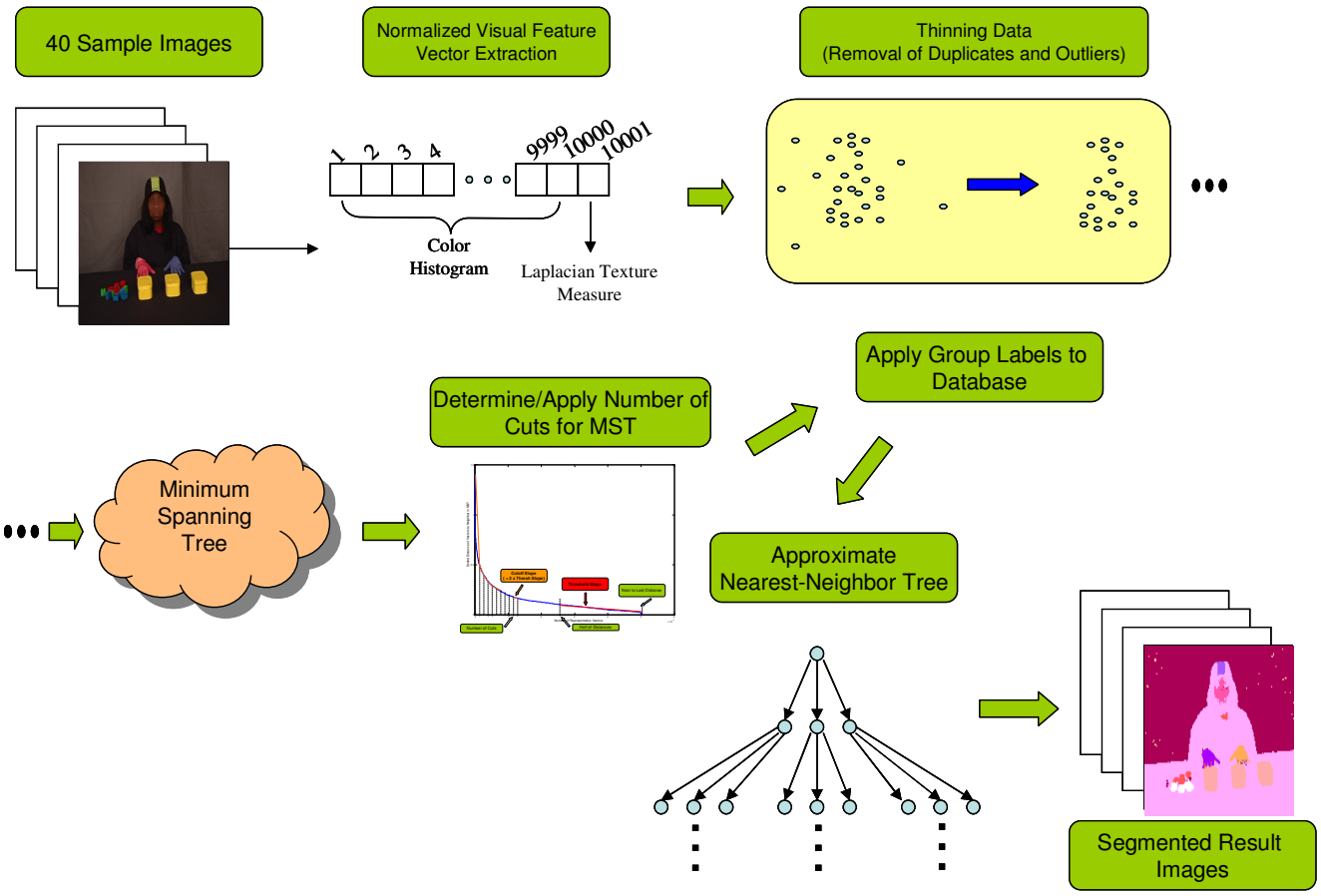
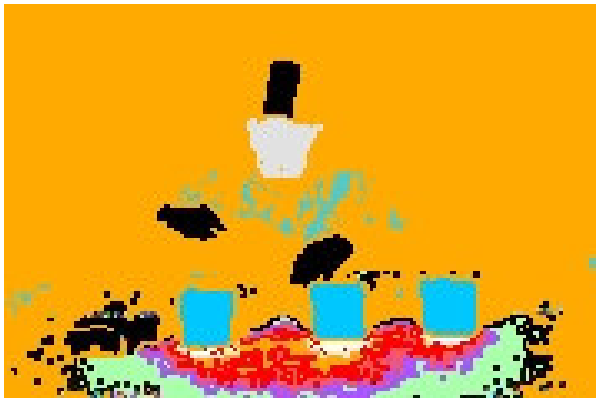


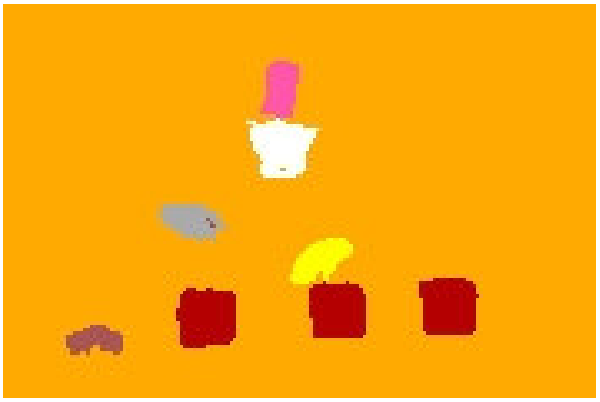
Figure 53: Unsupervised Video Segmentation Flow Chart

Originally, the method was tested with Dr. Wang's algorithm, but the results did not show as accurate of segmentation as desired (as seen in Figure 54b). By normalizing the vectors to unit norm which projected them to the unit hypersphere of their space, the resulting segmentation was shown to improve significantly (as seen in Figure 54c). This is the first major change to the unsupervised segmentation process.

(a)



(b)



(c)

Figure 54: (a) Original Image, (b) Segmentation without normalization (c) Segmentation with normalization

The reason that the distance measure used strictly with the non-normalized vector did not work as well was because as the dimensionality grew, the vectors were forced toward the origin. Since the vectors are primarily color histograms, they follow the L1 line/plane/hyperplane (depending on the dimension of the space). As can be seen in Figure 55, if the distance from origin to the center of the L1 norm in the case of 2 dimensions would be $\frac{\sqrt{2}}{2}$ but this can be shown to extend to $\frac{\sqrt{N}}{N}$ in an N dimensional case. The plot in Figure 56 shows how quickly the distance between the L1 norm plane and the origin drops as the dimension space continues to grow. By allowing our high dimensional feature vectors to remain on this L1 plane, a large amount of our vectors move near the origin. Since we use the L2 norm to measure between vectors, the effect of this move towards the origin is a loss of discriminating

ability of the L2 distance measure. To counteract this it was decided to normalize the vector which projects it to the L2 unit hypersphere. This allows the L2 norm to continue to be a functional distance measure for any dimensional space.

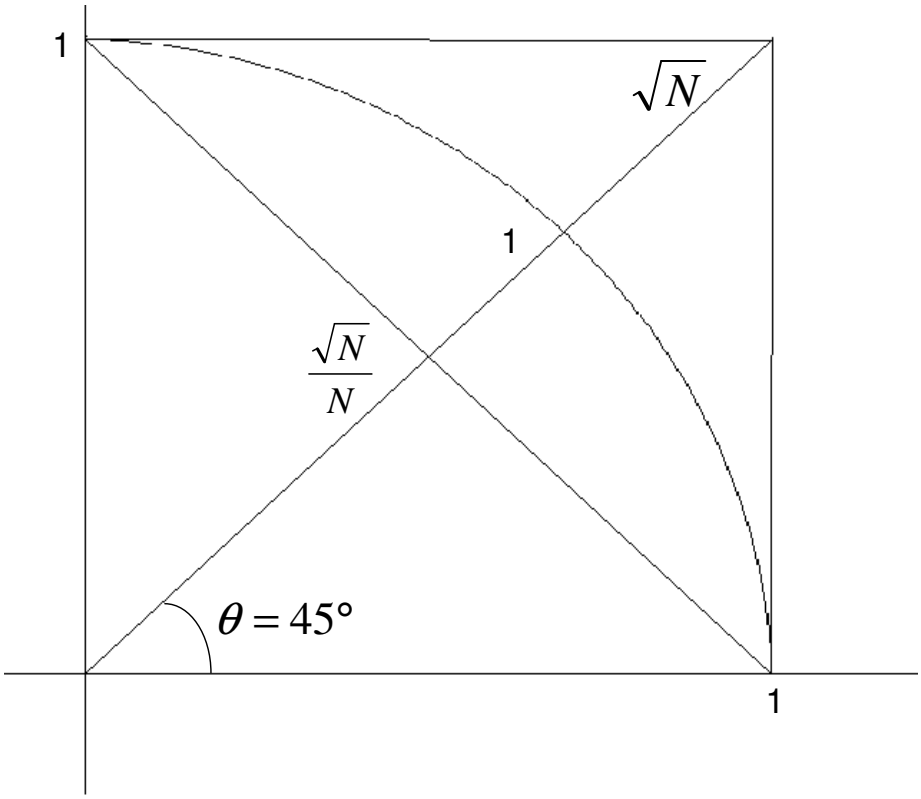


Figure 55: Two Dimension Projection Example

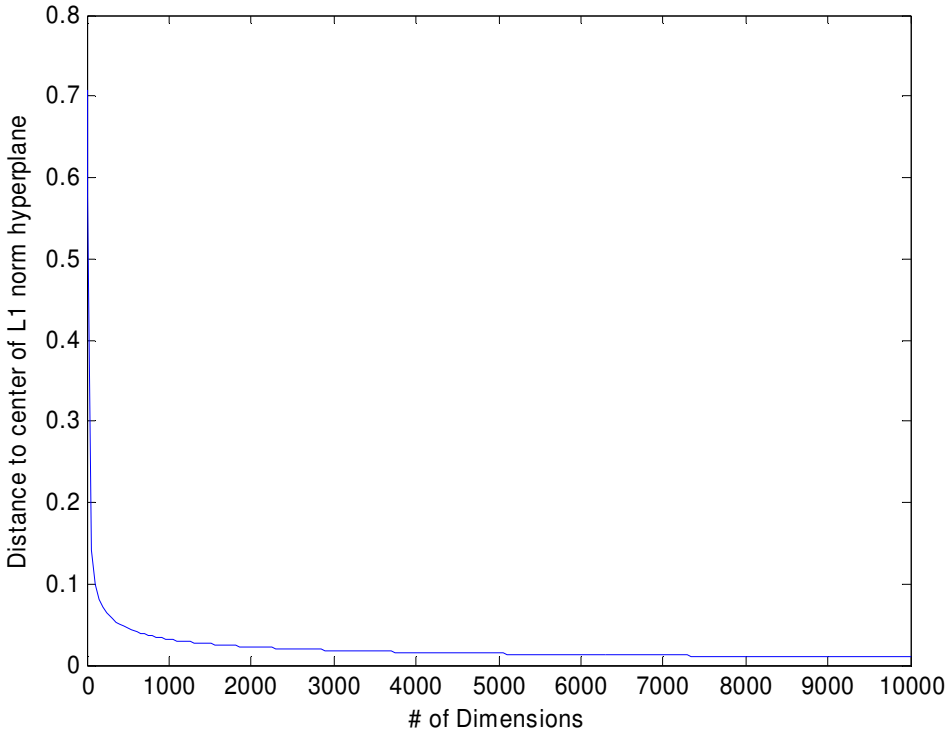


Figure 56: Distance to Center of L1 Norm Hyperplane as a Function of Dimensionality

Figure 53 shows the overall process for an unsupervised method of segmenting the video. The method is similar to the method of Experiment 1 except a few extra steps are necessary to determine the natural groups of the video. Since the user is removed, the training must involve collecting training images from the video. For our process, 40 images are collected from the data stream. All objects are generally visible at the beginning of the videos since the tasks are primarily assembly tasks. To ensure significant sampling of all the objects, 20 of the sample images are collected from the first fourth of the video and the remaining 20 are collected from the rest of the video. From the 40 images, a total of 840,160 vectors (21,004 from each image) are extracted. Due to the large number of vectors, the database is thinned. The data base is thinned by finding all duplicate vectors (vectors within 0.0000001 distance of another vector) and reducing them to a single representation. The vectors are also thinned by using the mean distance between the vectors of the first 2 images as a threshold and removing any vectors whose nearest neighbor is further away than that threshold. Table 30 shows the

mean resultant database size is composed of 48,283 vectors. This interestingly coincides with a report from Kalayeh and Landgrebe that the number of training vectors necessary to train a system using linear classifiers is on the order of 5 times the dimensionality of the feature space [130]. The amount of time for thinning each database was usually no more than 1 day.

Table 30: Overall Thinning Reduction Percentages

Mean Resultant Database Size	Mean % of Database	Mean Duplicate Size	Mean Duplicate %	Mean Outlier Size	Mean Outlier %
48283	5.75 %	702290	83.59 %	89583	10.66 %

The thinned data is used to create a Minimum Spanning Tree (MST). To aid the explanation of the unsupervised segmentation of images, it is useful to go through the process that Wang developed for her dissertation work [123]. On the assumption that a group of feature vectors can be differentiated from another group by using an L2 distance norm, a minimum spanning tree was created to determine naturally forming groups.

The autonomous segmentation methodology for Wang started the same as for the supervised segmentation for Tugcu; a database of vectors is collected to be used for the creation of clusters. The difference between them is that the supervised database was collected and labeled by the user while the unsupervised method would extract the unlabeled visual feature vectors from 40 sample images from the environment. With this multitude of unlabeled visual feature vectors, Wang would use the vectors to create a minimum spanning tree to determine the natural groups. In Wang's dissertation [129], she explains the minimum spanning tree as the following:

"The minimum spanning tree method is a graph analysis of arbitrary point sets of data. In a graph, two points can be connected by either a direct edge or a sequence of edges called a path. A loop in a graph is a closed path. A connected graph has one or more paths between any pair of points. A tree is a connected graph without closed loops. A spanning tree is a tree that contains every point in the data set. If a value is assigned to each edge in the tree, the tree is called a weighted tree. For

example, the weights for each edge can be the distance between the two points. The weight of a tree is the total sum of edge weights in the tree. The minimum spanning tree (MST) is the spanning tree that has the minimal total weight among all possible spanning trees for the data set. The minimum spanning tree has the following property that can be used for clustering if the weight associated with each edge denotes the distance between the two points. That is, the weight associated with every edge in the minimum spanning tree will be the shortest distance between two sub-trees that are connected by that edge. Therefore, removal of the longest edge will theoretically result in a two-cluster grouping. Removal of the next longest edge will result in a three-cluster grouping, and so on. These correspond to choosing breaks where maximum weights occur in the sorted edges. When the tree is built, after sorting the edge in decreasing order, the edges can be cut to form clusters."

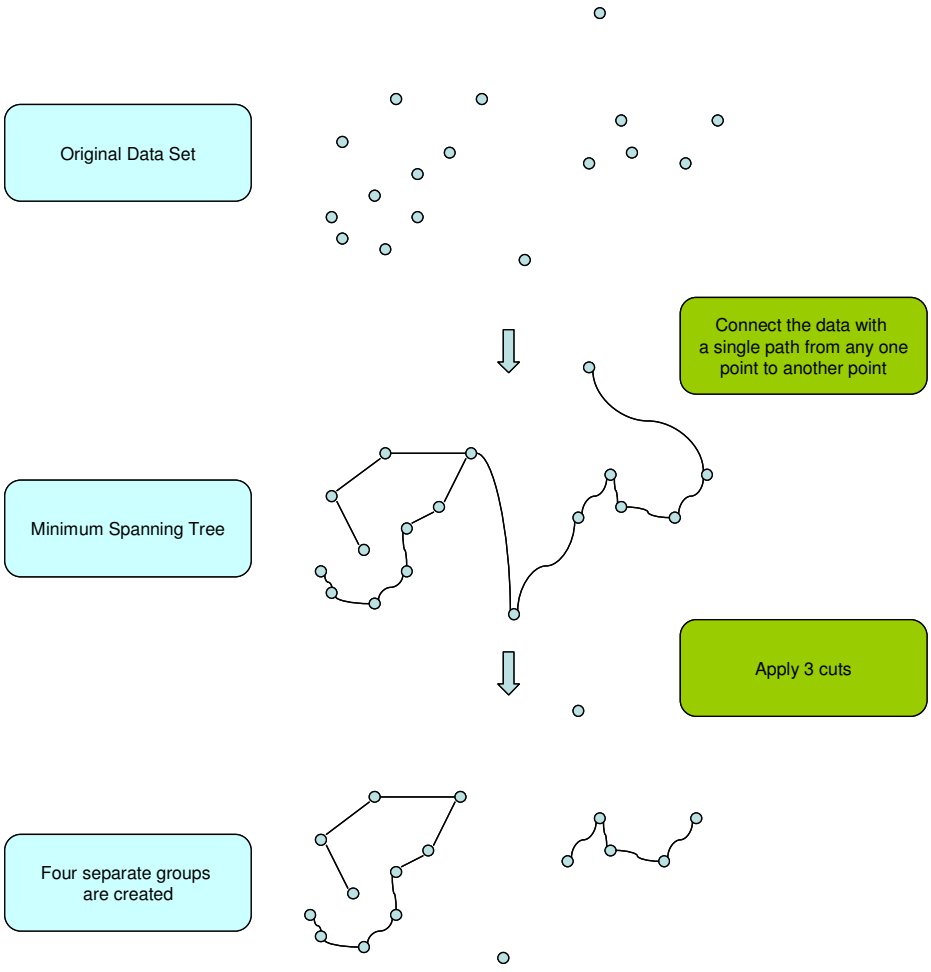


Figure 57: Minimum Spanning Tree Example

With these edges, Wang would designate the number of cuts to be applied to the edges. The cuts would be applied to the longest remaining edges. Depending on the number of cuts applied, a number of natural groups will fall out from the data. By looking at Figure 57, an example of the MST process is shown. A sample set of points are turned into a minimum spanning tree by connecting them to their neighbors but maintaining the rule of only one path existing between any 2 points. Finally, a number of cuts (in the example, 3 cuts) are applied to the tree which involves removing that number of the longest connections between points. As can be seen from the Figure 57, 4 natural groups fall out of the set of points. Once the natural groups were formed, Wang would observe the segmentations to determine if groups were over-segmented and recombine them. Over-segmenting occurs when too many cuts are applied to the tree and groups are broken into sub-groups that should remain as one group. Wang's method for analysis has been modified to create a fully automated approach for evaluating the natural segmentations of an image. The creation of the MST usually took no more than 4 hours for these databases.

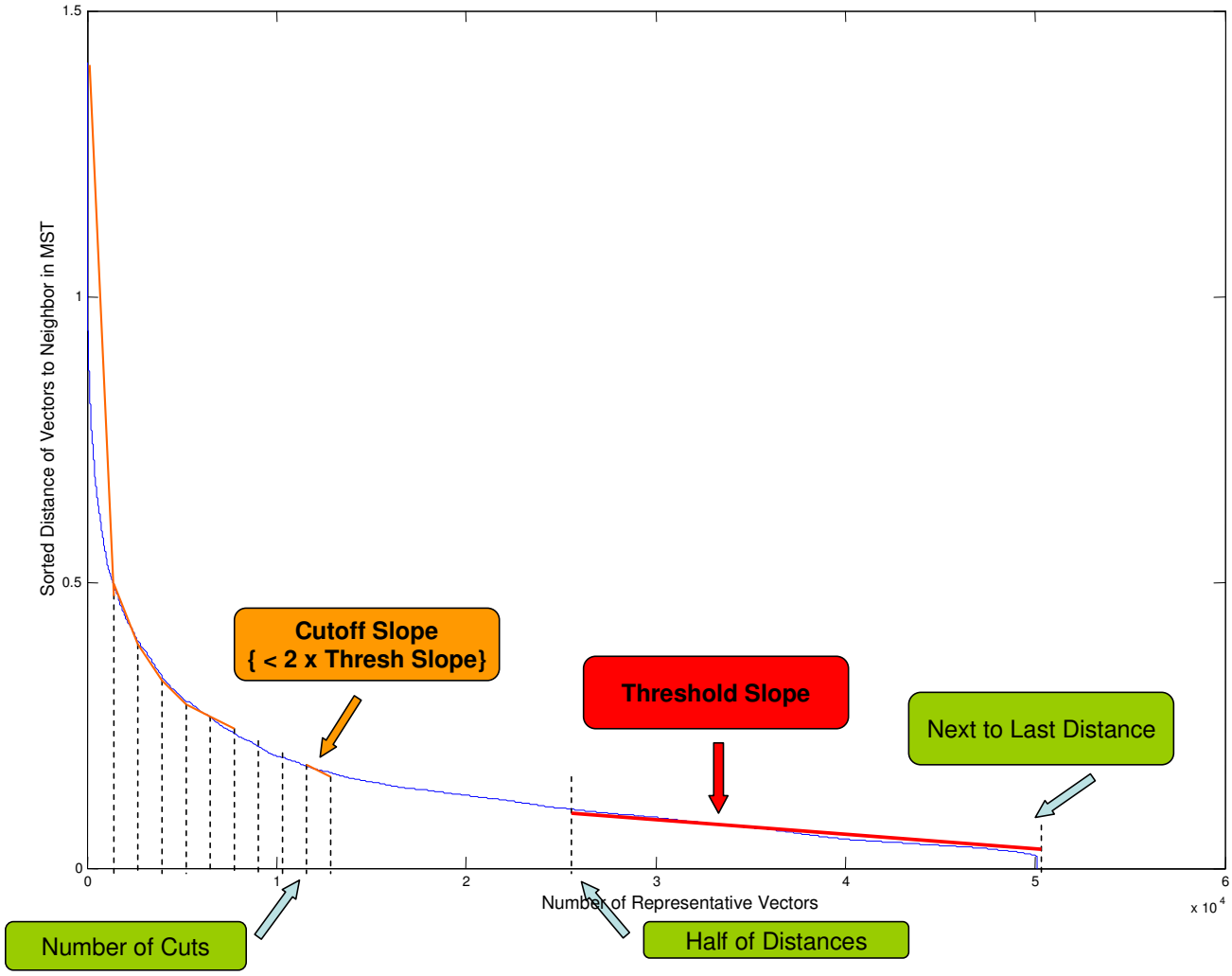


Figure 58: Number of Cuts Algorithm

The next step is to determine the correct number of cuts in the minimum spanning tree to provide optimal segmentation. The plot in Figure 58 shows the distances of the minimum spanning tree from the largest to smallest. The behavior of the plot seems to remain somewhat consistent for all the databases (one database for each video). There is a sharp drop in distances and the plot eventually settles out toward a nearly linear decrease. By experimenting with cuts across various databases, it was found that the best number of cuts for the MST resided near the beginning of when the tail of the plot becomes approximately linear. The plots are always close to linear before half of the data is reached. The way the number of cuts is determined is to calculate the threshold slope which consists

of using the last half of the distance values and calculating a slope. The threshold slope is used to monitor the slopes calculated from the beginning of the distances plot at a sampling size of 200 samples. The cutoff slope is determined by backtracking the threshold slope until the cutoff/threshold slope ratio is less than 2. The number where this condition occurs is determined to the number of cuts to be applied to the MST.

After a number of cuts are applied, a number of natural groups are determined. For a group to be designated significant, it must contain at least 100 vectors in its grouping. If a similar concept were to be applied to the example MST in Figure 56 with a minimum of 4 vectors, then the number of representative groups would fall to just 2 valid groups. Each of these groups is given a number and a label corresponding to this group number. In the supervised case, the labels were things such as Hand1 or Stripe. In the unsupervised case, the labels are designated as Object1 or Object5. Since a label is available, an approximate nearest neighbor tree can be created from the data. Once again, the approximate nearest neighbor tree is used to create the segmented video.

One addition to note about the creation of the approximate nearest neighbor tree is the addition of an accuracy check. As discussed previously, the tree is formed by randomly selecting 3 vectors and grouping the rest of the data according to those vectors in the supervised case. For the autonomous case, the number of node vectors per group was increased to 5 and the number acceptable vectors in a leaf node were under 1000 vectors. Since the 5 node vectors are chosen at random for each case, this leaves the possibility of creating a tree that does not properly represent the data. Previously, the segmentation created from the trees would be observed by the user to determine if the tree was accurate. To automate that check, an image from the video is segmented using an exact nearest neighbor process. A tree would be created and that tree would segment the same image. If the tree's segmented image agrees with 98% of the nearest neighbor image, the tree is saved. Otherwise, the program recreates the tree. To limit the amount of time the program can run, a maximum of 20 trees are created if none of those trees meet the 98% accuracy threshold. Of the 20 trees, the most accurate

one is saved. This process usually took no more than 1.5 days, but also contained a lot of variance if an optimal tree is found early in the 20 step iteration (time sometimes as low as 2-4 hours).

Using this method for applying cuts to the minimum spanning tree yielded an acceptable segmentation for 84 of the 100 videos that were tested. The requirements for an acceptable segmentation were correct separation of the crucial percepts (two hands and the strip on the hat). Object identification varied with the amount of example vectors available to the system. Due to limitations such as the size of objects and amount of time in view of the camera, certain objects would either not have enough representation in the video to be modeled by the spanning tree with its current settings. In some of the cases, the number of cuts applied would reduce a group to below the 100 vector threshold discussed above. In the case of a significant percept (i.e., hands or stripe) being affected, the number of cuts had to be manually adjusted to provide optimal segmentation. In Table 31, the adjustments of the 16 unacceptable segmentations are shown.

Table 31: Adjusted Cuts Statistics

Participant #_Task #	Algorithm Cuts	Adjusted Cuts	Database Size	% Difference
11_1	8527	4500	39483	10.20
11-7	8857	1000	30411	25.84
13-4	6488	30000	89121	26.38
13-7	7600	4600	25557	11.74
14-1	10179	No change*	34907	0.00
14-4	8284	16000	74786	10.32
14-7	9273	5500*	33029	11.42
16-1	11001	8000*	42620	7.04
16-3	9330	8500	34134	2.43
16-6	6361	30000	90460	26.13
16-7	8006	6500*	31191	4.83
19-4	6406	22400	77722	20.58
19-7	8257	4400*	27273	14.14
19-8	8447	4000	30023	14.81
20-7	6840	No change*	27854	0.00
20-8	7747	No change*	31733	0.00

* Insufficient database to represent crucial percepts

Of the 16 videos that the algorithm failed to produce desired segmentation, 7 did not contain the information necessary to produce the desired segmentation with any number of cuts. The remaining 9 needed about an average of 16% adjustment with their respective database size. It also should be noted that half of the failed videos were occlusion tasks (task 7 and task 8) which were specifically designed to have a significant portion of the action performed outside the direct view of the camera. Also, tasks 1 and 4 contain fairly large objects that occlude the view of the hands. Task 1 had three flashlight handles standing up next to each other. Depending on the participant's position, a majority of their left hand (Hand2) was partially blocked from view. The same observation could be noted

about the stack of containers available in task 4. When noting possible difficulties of those 4 tasks, the 14 out of 16 videos belonging to one of those 4 groups becomes a bit more understandable.

The approximate nearest neighbor trees are used to segment the entire video. The tree with a branching of 5 nodes, a maximum leaf node capacity of 1000 vectors, and a maximum level of 60 was able to process a frame of video in less than 1 minute, usually closer to 30 to 45 seconds. Due to the length of the videos (which were recorded at 30 frames per second) being no longer than 2 minutes (3600 frames), the time for processing took no more than 2.5 days. A vast majority of the videos were closer to 1 min and 30 seconds or less in duration (resulting in a processing period of less than 1 day).

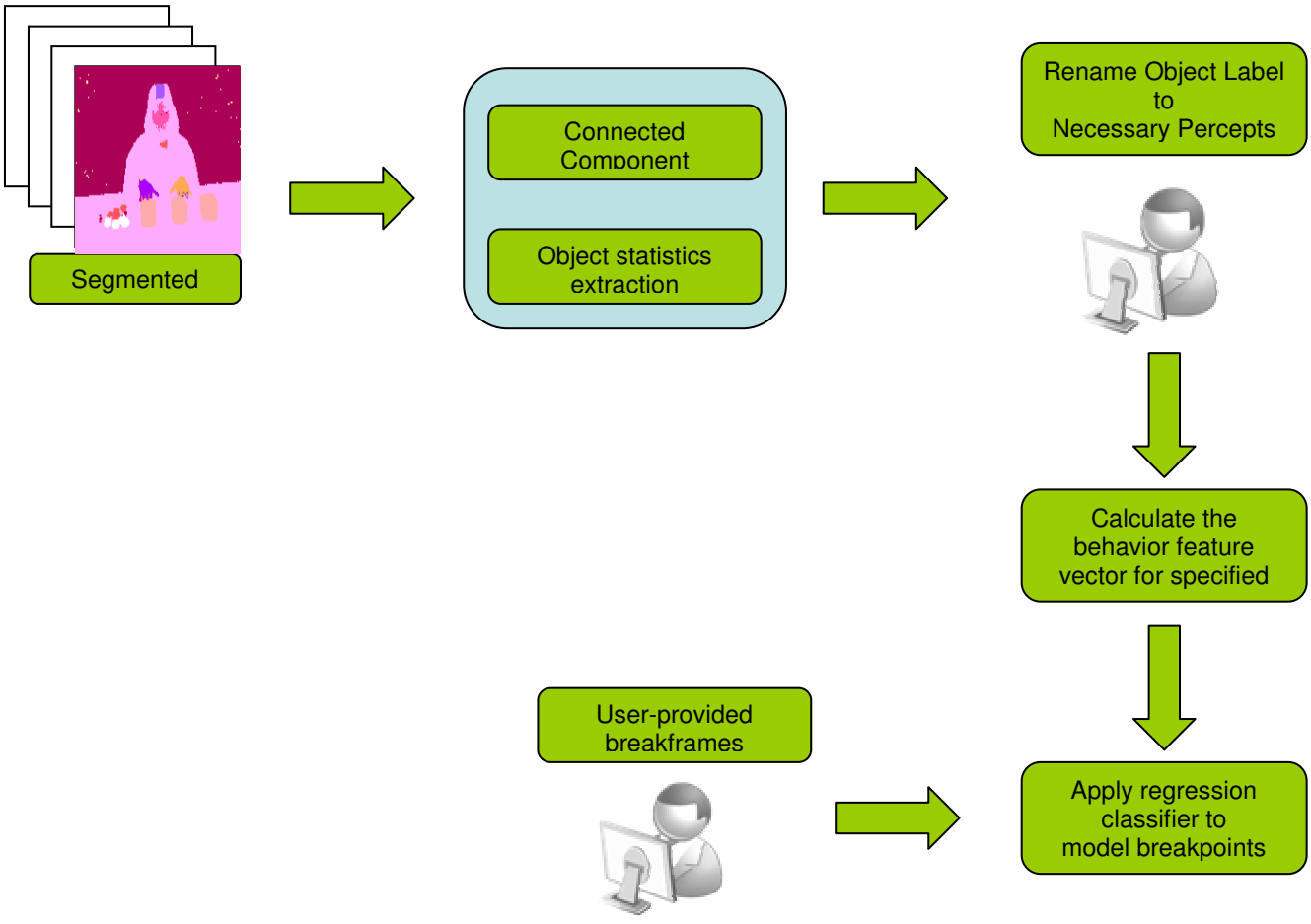
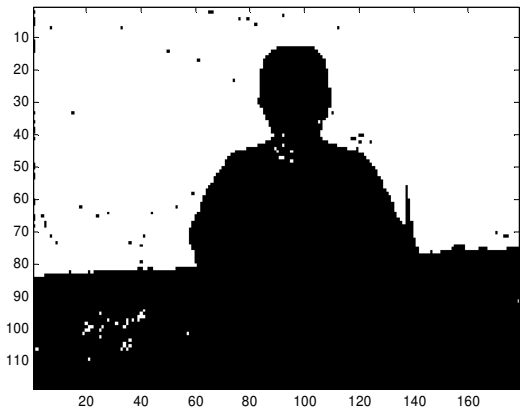


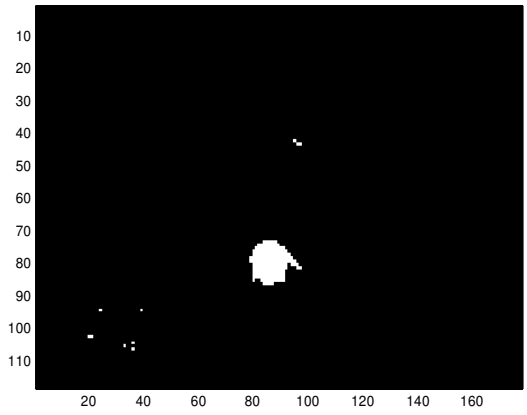
Figure 59: Unsupervised Behavior Extraction Flow Chart

Once the segmented video is created, the process of behavior extraction is nearly the same as in Experiment 1 as seen in Figure 59. The one major difference is the renaming segment. For the data of the segmented video to integrate with the previous process, the label names must correspond to the label names used in calculating the behavior feature vector (i.e. Object1 -> Hand1, Object2 -> Background, …). The only labels that must be identified are Hand1, Hand2, Stripe and Background. Background allows the program to ignore the percept. The rest of the objects such as the Legos or containers are only identified as pertinent objects so the "Object#" name is applicable. The program goes through and determines that the large objects that take up a majority of the frame throughout the video are considered Background. The Stripe is determined to be the highest object throughout the video that is not a Background object. The 2 objects that move the most are determined to be the hands (the hand predominately on the left of the image being Hand1 and the other being Hand2).

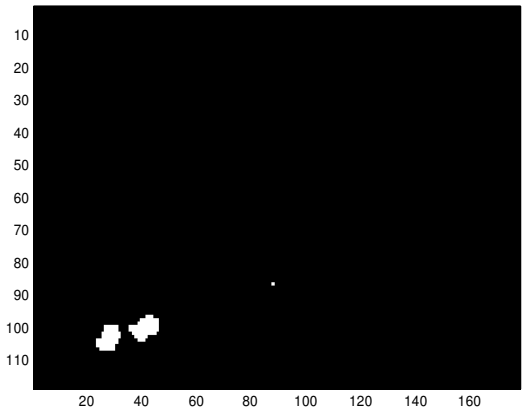
Due to fluctuations in the frame representation caused by video encoding when converted to Cinepak, the parameter of motion that was expected to determine the hands was not sufficient. Since the automatic object labeling algorithm was becoming very specific for this particular study, the objects are determined by the user. This is done by looking at individual percept representations and providing the correct name (example in Figure 60). The label renaming of percepts took about 1-2 minutes per video. Once the labels are renamed, the exact same process for behavior vector extraction is used. The behavior feature vector extraction for a single video took about 8 minutes.


10
20
30
40
50
60
70
80
90
100
110
20
40
60
80
100
120
140
160

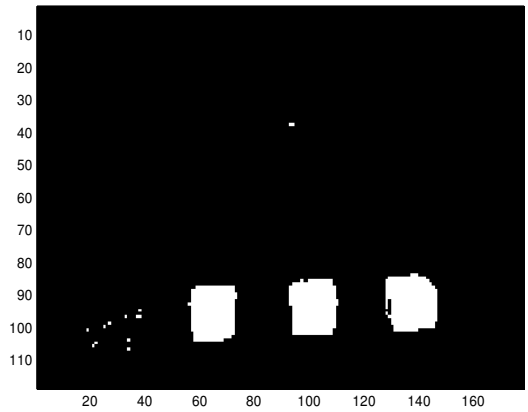
(a)


10
20
30
40
50
60
70
80
90
100
110
20
40
60
80
100
120
140
160

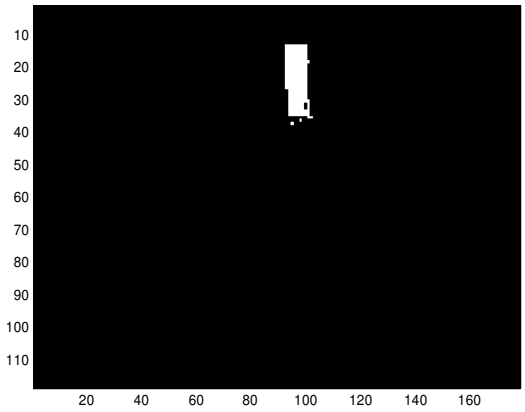
(b)


10
20
30
40
50
60
70
80
90
100
110
20
40
60
80
100
120
140
160

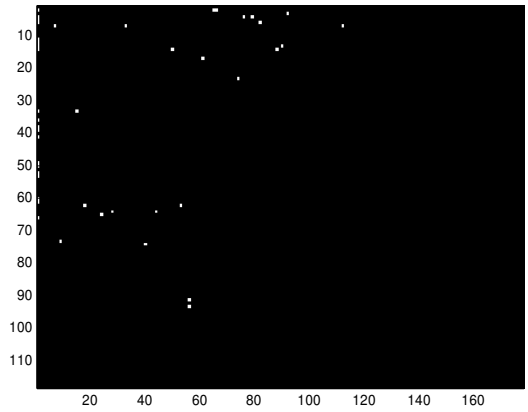
(c)


10
20
30
40
50
60
70
80
90
100
110
20
40
60
80
100
120
140
160

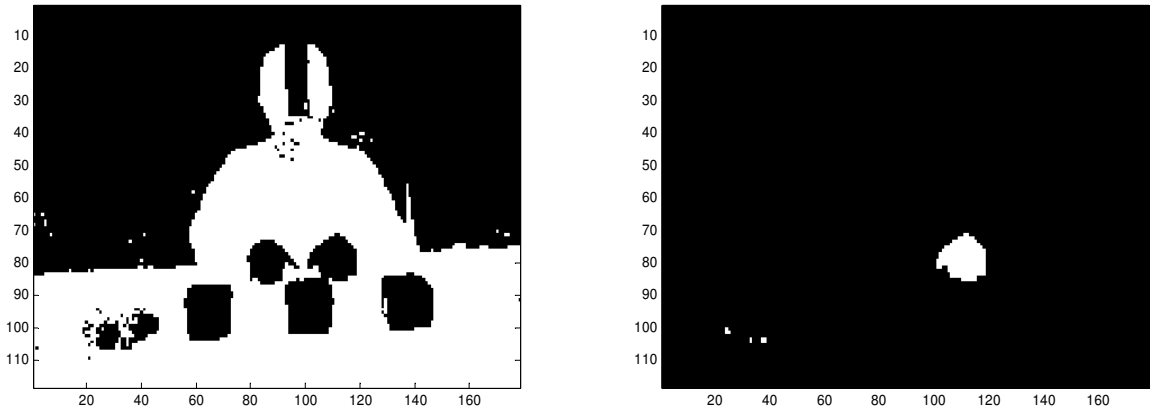
(d)


10
20
30
40
50
60
70
80
90
100
110
20
40
60
80
100
120
140
160

(e)


10
20
30
40
50
60
70
80
90
100
110
20
40
60
80
100
120
140
160

(f)



(g) (h)

Figure 60: Example Label Representations in a Frame; (a) Object 1 – Background, (b) Object 2 – Hand1, (c) Object 3 – Legos, (d) Object 4 – Containers, (e) Object 5 – Stripe, (f) Object 6 – Noise, (g) Object 7- Background, (h) Object 8 – Hand2

The same analyses done in the supervised version are done to the autonomous version to ensure that the results are basically equivalent. Once the behavior feature vectors are extracted, linear, quadratic, and mahalanobis regression techniques are used to analyze the vectors that are created from a bin-size of 6. Table 32 compares both the autonomous results with the supervised results at bin-size of 6. The average hit rate and false alarm rate are slightly higher for the autonomous but the results are nearly identical. Table 33 reinforces the observation from Table 32 showing a very slight increase in both the hit rate and false alarm rate for the autonomous results.

Table 32: Comparison of Supervised and Autonomous at Bin-Size of 6 using Subject Jack Knife

	Linear				Quadratic				Mahalanobis			
	HR (μ, σ)	FAR (μ, σ)	d_1'	d_2' (μ, σ)	HR (μ, σ)	FAR (μ, σ)	d_1'	d_2' (μ, σ)	HR (μ, σ)	FAR (μ, σ)	d_1'	d_2' (μ, σ)
Supervised	(0.859, 0.079)	(0.385, 0.011)	1.368	(1.411, 0.385)	(0.819, 0.083)	(0.349, 0.096)	1.300	(1.362, 0.338)	(0.488, 0.129)	(0.132, 0.067)	1.085	(1.131, 0.332)
Autonomous	(0.868, 0.086)	(0.402, 0.132)	1.132	(1.393, 0.427)	(0.843, 0.088)	(0.377, 0.130)	1.134	(1.334, 0.399)	(0.489, 0.133)	(0.132, 0.091)	1.400	(1.107, 0.411)

Table 33: Comparison of Supervised and Autonomous at Bin-Size of 6 using Task Jack Knife

	Linear				Quadratic				Mahalanobis			
	HR (μ, σ)	FAR (μ, σ)	d_1'	d_2' (μ, σ)	HR (μ, σ)	FAR (μ, σ)	d_1'	d_2' (μ, σ)	HR (μ, σ)	FAR (μ, σ)	d_1'	d_2' (μ, σ)
Supervised	(0.857, 0.084)	(0.387, 0.119)	1.356	(1.410, 0.374)	(0.820, 0.085)	(0.353, 0.104)	1.292	(1.356, 0.341)	(0.487, 0.123)	(0.136, 0.071)	1.063	(1.122, 0.356)
Autonomous	(0.868, 0.088)	(0.403, 0.135)	1.131	(1.389, 0.441)	(0.844, 0.090)	(0.378, 0.130)	1.135	(1.340, 0.400)	(0.502, 0.140)	(0.140, 0.098)	1.355	(1.118, 0.408)

After ensuring that the regression analysis results are similar, we would guess that the correlation between the autonomous system and the other raters would not show much of a difference as well. Using the test and base vector analysis, the autonomous system was placed at the lowest priority in terms of truth (i.e. original rater > interrater > system rater > supervised system estimate > autonomous system estimate). As can be seen in Table 34 – 36, a similar amount of correlation as seen in the previous supervised analysis is also shown here.

Table 34: Correlation with 0 bin variability with bin-size of 6

Test vs Base	Probability of Base	Accuracy of Overlap
Auto_System Est. vs Original	0.293	0.267
Auto_System Est. vs Interrater	0.251	0.247
Auto_System Est. vs System Rater	0.159	0.172
Auto_System Est. vs System Est.	0.152	0.158

Table 35: Correlation with 1 bin variability with bin-size of 6

Test vs Base	Probability of Base	Accuracy of Overlap
Auto_System Est. vs Original	0.653	0.791
Auto_System Est. vs Interrater	0.582	0.733
Auto_System Est. vs System Rater	0.470	0.766
Auto_System Est. vs System Est.	0.449	0.701

Table 36: Correlation with 2 bin variability with bin-size of 6

Test vs Base	Probability of Base	Accuracy of Overlap
Auto_System Est. vs Original	0.802	0.893
Auto_System Est. vs Interrater	0.741	0.846
Auto_System Est. vs System Rater	0.695	0.935
Auto_System Est. vs System Est.	0.681	0.902

Experiment 5: Natural Scene Testing

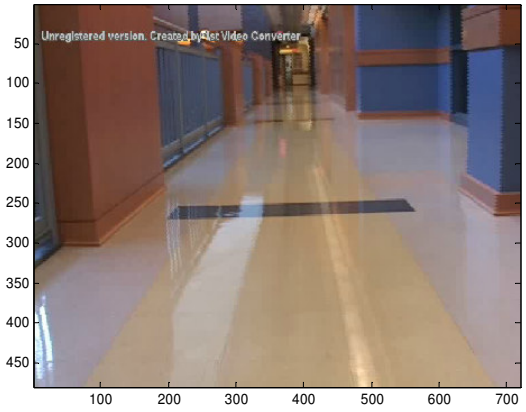
To determine how robust the system is to moderately controlled environments and natural scenes, the autonomous system was applied to 3 naturally occurring scenes. A sample video was created for 2 indoor scenes (3rd floor hallway overlooking the atrium in Featheringill Hall and the 3rd floor hallway connecting to Jacobs Hall) and 1 outdoor scene (the path between Featheringill Hall (FGH) and the Free Electron Laser (FEL) center). Figure 61 shows the 3 scenes and sample segmentations and Table 37 shows the statistics for determining the cuts necessary to provide the sample segmentations.

One main point to note is the system tends to under-segment the scene for the natural scenes. This is probably due to the higher amount of group overlap among the vectors. The controlled environments are designed to have objects that look drastically different from one another to provide easier segmentation. Natural scenes have reflections and less color differentiation between objects in the environment. The vectors formed from these occurrences may form a bridge between different objects. For example, the white floor and the wood panel are fairly different visually but the vectors formed from the wood panel reflection on the white floor would connect the two groups. Hence, the algorithm will allow the two objects to be united as one large group. The only way around this problem currently is to force the number of cuts applied to the MST to be enough that the groups are segmented, but that requires human intervention at this time.

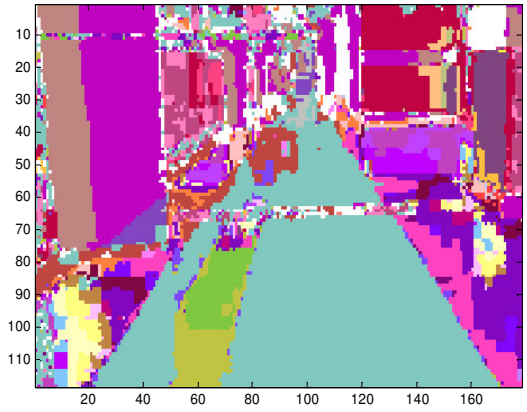
Another point to note is the difference in database size of the controlled environments and the natural scenes. The controlled environments database size averaged at about 50,000 vectors where the natural scene environments average at around 400,000 vectors. The difference in the size of the training databases forces the system to take much longer (about 10-15 times) longer to progress through the autonomous segmentation steps.

Table 37: Natural Scene Statistics

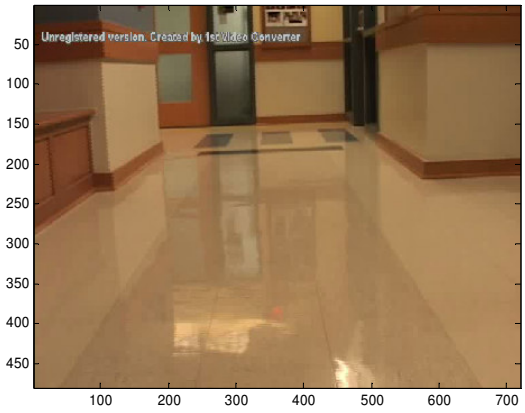
Scene	Algorithm Cuts	Adjusted Cuts	Database Size	% Difference
Indoor Atrium	44973	250000	382757	53.57
Indoor Jacob Hall	71197	none	222898	n/a
Outdoor FGH	22430	175000	548939	27.79



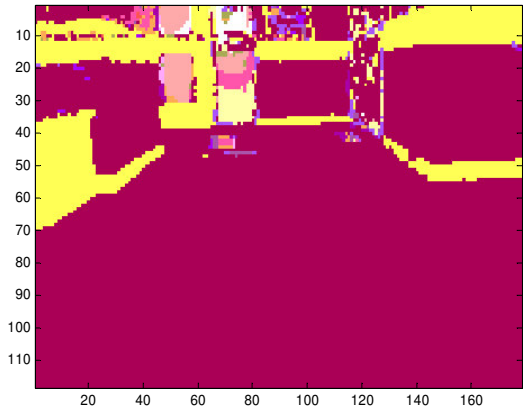
(a)



(b)

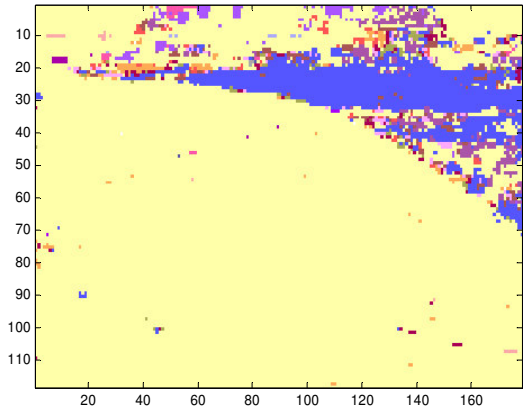


(c)



(d)

(e)



(f)

Figure 61: Natural Scene Segmentation Examples; (a) Indoor Atrium, (b) Indoor Atrium Segmentation, (c) Indoor Jacob Hall, (d) Indoor Jacob Hall Segmentation, (e) Outdoor FGH, (f) Outdoor FGH Segmentation

CHAPTER VIII

CONCLUSION AND FUTURE WORK

Conclusion

The goal of this research was to create a vision system that uses high dimensional visual feature vectors and evaluate its performance in conjunction with supporting the intentional vision studies done by the Vanderbilt Psychology Department. The vision system was to be tested using supervised methods and extended to be autonomous for flexibility in interdisciplinary use. The intentional vision research found that there were a set of sub-actions that were highly correlated with the determination of significant moments parsing an activity. These sub-actions included hand-to-object contacts, object-to-object contacts, occlusions, and eye movements. Using these sub-action descriptions, a set of behavior features were established to either singularly or conjunctively account for those sub-actions. Subsequently, these behavior features were tested to determine their ability to predict the system rater's breakpoints (significant moments) and tested to determine their correlation with the original rater and interraters. The features were also analyzed to see if they could be used to also predict overall tasks and/or participants. Furthermore, the autonomous system was tested on some mildly controlled and natural environments outside the data given by the intentional vision research.

It was determined that use of a high dimensional visual system had some interesting properties that are interesting for further study. The projection of high dimensional sparse feature vectors seems to allow for a more intuitive segmentation of the scenery. Both the supervised and the autonomous systems were shown to consistently provide segmentations that are expected in controlled environments. One of the shortcomings of the systems is the amount of time necessary to train them. In the supervised case, the man hours necessary to train and segment 100 videos was on the order of

months as well as fairly in depth knowledge of the system to correct any training errors that occurred. Conversely, the autonomous system requires very little man hours to set the system up, but requires more time to train and segment the videos. It does have the added bonus of being able to work continuously.

Both systems were able to extract behavior feature vectors and reliably identify them according to the training set. A lot of the error is due to the fuzzy boundaries between sub-actions and the necessity of providing a definite segmentation to a indefinite moment. The data was shown to be correlated with the intentional vision research findings and supports the idea of using the objective analysis of the system to define the subjective human ratings and observations. The behavior features were found to have some information pertaining to the identification of a task when applied to a Hidden Markov Model, but almost no information to identify a participant from this set of examples.

The autonomous system is robust enough to segment natural and mildly controlled scenes pretty well. The time issue becomes even more of a factor here when dealing with databases that do not reduce as much as the controlled environment databases do. For all steps in the process except the segmentation of the video using the approximate nearest neighbor tree, the time increases anywhere from 10-15 times the original reported times. The cuts algorithm did tend to under segment the scenery for the natural scenes. The overall information in the database has the capability of performing desirable segmentations at any environment tested thus far.

Future Work

Improved Cuts Algorithm

One of the weaknesses of the system was the simple algorithm used to determine the number of cuts. Though it had a 84% success rate in the controlled environments, it only worked for 1 of the 3 natural scenes. By applying an improved algorithm that takes into account stability of clusters as cuts

are applied, maybe this secondary measure can work in conjunction with the slope algorithm to determine a more appropriate number of cuts to apply to the MST.

Integration to Parallel Computing

As stated before, one of the biggest shortcomings of the system is it being slow. By using multi-core processors or multi-thread graphics card (i.e. GeForce 8800 GTS), the speed of the system can be improved drastically. Most of the processes can be ported over to a parallel computing platform to drastically reduce computation time.

Improved Action Feature Vectors

The behavior feature vector used to find segmentation boundaries between actions as labeled by the user. The same behavior feature vectors do not appear to be as strong in actually describing the actions themselves. A new action feature vector should be used to describe the activity between the segments. Once this action vector is completed, it could be used to provide more accurate task identification. The task identification could be used to predict the next sub-actions to occur. By using this prediction and the subsequent action, a prediction error signal can be formed. This prediction error signal can be used to signal a new event or specify the need to change to a alternative motion model.

BIBLIOGRAPHY

[1] Aggarwal, J.K., Cai, Q**., Human motion analysis: A review**. *Computer Vision and Image Understanding*, 73(3):428–440, 1999.

[2] Aggarwal, J.K., Nandhakumar, N., **On the Computation of Motion from Sequences of Images: A Review**, *Proceedings of IEEE(76)*, 1988, pp. 917-935.

[3] Madabhushi, A., Aggarwal, J.K., **Using Head Movement to Recognize Activity**, *International Conference on Pattern Recognition*,Vol IV, 698-701, 2000

[4] Bobick, A.F., Davis, J.W., **The Recognition of Human Movement Using Temporal Templates**, *Pattern Analysis and Machine Intelligence(23)*, No. 3, March 2001, pp. 257-267.

[5] Kojima, A., Izumi, M., Tamura, T., Fukunaga, K., **Generating Natural Language Description of Human Behavior from Video Images**, *International Conference on Pattern Recognition* , Vol IV, 728-731, 2000.

[6] Peursum, P., Bui, H.H., Venkatesh, S., West, G.A.W., **Human action segmentation via controlled use of missing data in HMMs**, *International Conference on Pattern Recognition*, Vol IV, 440-445, 2004.

[7] Wallhqff, F., Zobl, M., Rigoll, G., **Action Segmentation and Recognition in Meeting Room Scenarios**, *ICIP04*(IV: 2223-2226).

[8] Ogale, A.S., Karapurkar, A., Aloimonos, Y., **View-Invariant Modeling and Recognition of Human Actions Using Grammars**, *Workshop on Dynamical Vision*, 2006, (115-126).

[9] Barbic, J., Safonova, A., Pan, J.-Y., Faloutsos, C., Hodgins, J. K., Pollard, N. S., **Segmenting Motion Capture Data into Distinct Behaviors**,. *Proceedings of Graphics Interface*, May 2004, .

[10] Rao, C., Yilmaz, A., Shah, M., **View-Invariant Representation and Recognition of Actions**, *IJCV(50)*, No. 2, November 2002, pp. 203-226.

[11] Ali, A., Aggarwal, J.K., **Segmentation and Recognition of Continuous Human Activity**, *EventVideo01,* 2001.

[12] Kang, S.B., Ikeuchi, K., **Determination of Motion Breakpoints in a Task Sequence from Human Hand Motion**, *CRA94*(551-556).

[13] Wang, L., Hu, W.M., Tan, T.N., **Recent developments in human motion analysis**, *PR(36)*, No. 3, March 2003, pp. 585-601.

[14] Rui, Y. , Anandan, P., **Segmenting Visual Actions based on Spatio-Temporal Motion Patterns**, *CVPR00*(I: 111-118).

[15] Jiar, Y., Wheeler, M.D., Ikeuchi, K., **Hand Action Perception and Robot Programming**, *CMU-CS-TR*-96-116, March 1996.

[16] Kuniyoshi, Y., Inoue, H., **Qualitative Recognition of Ongoing Human Action Sequences**, *IJCAI93*(1600-1609).

[17] Kuniyoshi, Y., Inaba, M., Inoue, H., **Learning by Watching: Extracting Reusable Task Knowledge from Visual Observation of Human Performance**, *RA(10)*, 1994, pp. 799-822.

[18] Yu, C., Ballard, D., **Learning to Recognize Human Action Sequences**, IEEE International Conference on Development and Learning (ICDL'02), Cambridge, MA, June 12 - 15, 2002, pp. 28-34

[19] Valera, M., Velastin, S.A., **Intelligent distributed surveillance systems: a review**, *VISP(152)*, No. 2, April 2005, pp. 192-204.

[20] Guerra-Filho, G., Fermüller, C., Aloimonos, Y. **Discovering a Language for Human Activity**. In Proc. of the AAAI 2005 Fall Symposium on Anticipatory Cognitive Embodied Systems (FS05), Washington, D.C., pages 70-77, 2005.

[21] Buxton, H., **Learning and Understanding Dynamic Scene Activity: A Review**. *IVC(21)*, No. 1, January 2003, pp. 125-136.

[22] Robertson, N., Reid, I.D., **A general method for human activity recognition in video**, *CVIU(103)*, No. 2-3, November-December 2006, pp. 232-248.

[23] Kojima, A., Tamura, T., Fukunaga, K., **Natural Language Description of Human Activities from Video Images Based on Concept Hierarchy of Actions**, *IJCV(50)*, No. 2, November 2002, pp. 171-184.

[24] Mori, T., Segawa, Y., Shimosaka, M., Sato, T., **Hierarchical recognition of daily human actions based on Continuous Hidden Markov Models**, *AFGR04*(779-784).

[25] Park, J., Park, S., Aggarwal, J.K., **Model-Based Human Motion Tracking and Behavior Recognition Using Hierarchical Finite State Automata**, *Lecture Notes in Computer Science, Springer Berlin* , Vol. 3046, 311-320, April 2004.

[26] Sullivan, J., Carlsson, S., **Recognizing and Tracking Human Action**, *ECCV02*(I: 629 ff.).

[27] Howe, N.R., Leventon, M.E., Freeman, W.T., **Bayesian Reconstruction of 3D Human Motion from Single-Camera Video**, *Advances in Neural Information Processing Systems 12*, 1999, edited by S. A. Solla, T. K. Leen, and K-R. Muller,2000.

[28] Tao, Y., Hu, H., **Colour-based human motion tracking for home-based rehabilitation**. *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, The Hague, The Netherlands, 773-781, 2004.

[29] Dempster, A.P., Laird, N.M., Rubin, D.B., **Maximum likelihood from Incomplete Data via the EM Algorithm.** *Journal of the Royal Statistical Society B(Methodological),* Vol. 39, No. 1, pp. 1-38, 1977.

[30] Rabiner, L., **A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.** *Proc. of the IEEE,* Vol. 77, No. 2, pp.257-285, 1989.

[31] Hunter, J., **Human Motion Segmentation and Object Recognition using Fuzzy Rules**, *Proceedings of 14th Annual IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN 2005)*, Nashville, TN, August 13-15, 2005,pp 210-216, 2005.

[32] Park, J., Park, S., Aggarwal, J.K., **Human Motion Tracking by Combining View-Based and Model-Based Methods for Monocular Video Sequences**. *ICCSA (3) 2003*: 650-659.

[33] Aggarwal, J.K., **Problems, ongoing research and future directions in motion research**, *MVA(14)*, No. 4, September 2003, pp. 199-201.

[34] Sato, K., Aggarwal, J.K., **Temporal spatio-velocity transform and its application to tracking and interaction**,*CVIU(96)*, No. 2, November 2004, pp. 100-128.

[35] Ogale, A., Fermuller, C., Aloimonos, Y., **Motion Segmentation Using Occlusions**, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 27, 6, June 2005, pp. 988-992

[36] Munoz-Salinas, R., Aguirre, E., Garcia-Silvente, M., **People detection and tracking using stereo vision and color**, *IVC(25)*, No. 6, 1 June 2007, pp. 995-1007.

[37] Denman, S., Chandran, V., Sridharan, S., **An adaptive optical flow technique for person tracking systems**, *PRL(28)*, No. 10, 15 July 2007, pp. 1232-1239.

[38] Yu, Y., Harwood, D., Yoon, K., Davis, L.S., **Human appearance modeling for matching across video sequences**, *MVA(18)*, No. 3-4, August 2007, pp. 139-149.

[39] Lanz, O.[Oswald], **Approximate Bayesian Multibody Tracking**, *PAMI(28)*, No. 9, September 2006, pp. 1436-1449.

[40] Lee, M.W. , Nevatia, R., **Body Part Detection for Human Pose Estimation and Tracking**, *Motion07*(23-23).

[41] Lu, S.J., Zhang, J., Feng, D.D., **Detecting unattended packages through human activity recognition and object association**, *PR(40)*, No. 8, August 2007, pp. 2173-2184.

[42] Deng, J.W., Tsui, H.T., **An HMM-based Approach for Gesture Segmentation and Recognition**,
ICPR00(Vol III: 679-682).

[43] Kim, D.[Daehwan], Kim, D.[Daijin], **An Intelligent Smart Home Control Using Body Gestures,** *2006 International Conference on Hybrid Information Technology (ICHIT'06)*, 447 – 452.

[44] Beauchemin, S.S., Barron, J.L., **The Computation of Optical-Flow**, Surveys(27), No. 3, September 1995, pp. 433-467.

[45] Moons, T., Pauwels, E.J., Van Gool, L.J., Oosterlinck, A., **Towards a General Framework for Feature Extraction**, *CVPR92*(865-868).

[46] Murase, H., Nayar, S.K., **Learning Object Models from Appearance**,*AAAI-93*(836-843) *Model Acquisition*.

[47] Novak, C.L., Shafer, S.A., **Anatomy of a Color Histogram**, *CVPR92*(599-605).

[48] Shah, M., **Understanding human behavior from motion imagery**, MVA(14), No. 4, September 2003, pp. 210-214.

[49] Sharma, V., Davis, J.W., **Simultaneous Detection and Segmentation of Pedestrians using Top-down and Bottom-up Processing**, *VS07*(1-8).

[50] Sharma, V., Davis, J.W., **Extraction of Person Silhouettes from Surveillance Imagery using MRFs**, *WACV07*(33-33).

[51] Tugcu, M., Wang, X., Hunter, J.E., Phillips, J., Noelle, D., and Wilkes, D. M., **A computational Neuroscience model of working memory with application to robot perceptual learning**, *Third IASTED International Conference on Computational Intelligence* (CI), Banff, Alberta, Canada, July 2-4 2007.

[52] Ogawara, K., Iba, S., Tanuki, T., Kimura, H., Ikeuchi, K., **Acquiring hand-action models by attention point analysis**, *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*, May, 2001, pp. 465-470.

[53] Ogawara, K., Iba, S., Tanuki, T., Kimura, H., Ikeuchi, K., **Recognition of human task by attention point analysis**, *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, October, 2000, pp. 2121-2126.

[54] Ke, Y., Sukthankar, R., and Hebert, M., **Event Detection in Crowded Videos**, *IEEE International Conference on Computer Vision*, October, 2007.

[55] Cabero, M.J., De la Torre Frade, F., Arizaga I., Sanchez, A., **Indoor People Tracking based on Dynamic Weighted Multidimensional Scaling**, *IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, October, 2007.

[56] Stein, A., Hoiem, D., Hebert, M., **Learning to Find Object Boundaries Using Motion Cues**, *IEEE International Conference on Computer Vision (ICCV)*, October, 2007.

[57] Stein, A., Hebert, M., **Combining Local Appearance and Motion Cues for Occlusion Boundary Detection**, *British Machine Vision Conference (BMVC)*, September, 2007.

[58] Unnikrishnan R., Pantofaru, C., Hebert, M., **Toward Objective Evaluation of Image Segmentation Algorithms**, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 29, No. 6, June, 2007, pp. 929-944.

[59] Koppal, S.J., Narasimhan, S.G., **Clustering Appearance for Scene Analysis**, *IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 2, June, 2006, pp. 1323 - 1330.

[60] De la Torre Frade, F., Kanade, T., **Discriminative Cluster Analysis**, *International Conference on Machine Learning*, ACM Press, New York, NY, USA, Vol. 148, June, 2006, pp. 241 - 248.

[61] Lin, W., Liu, Y., **Tracking Dynamic Near-regular Textures under Occlusion and Rapid Movements**, *9th European Conference on Computer Vision*, May, 2006.

[62] Xiao, J.,Georgescu, B.,Zhou, X., Comaniciu, D., Kanade, T., **Simultaneous Registration and Modeling of Deformable Shapes**, *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 2, 2006, pp. 2429 - 2436.

[63] Sivic, J.,Russell, B., Efros, A.A.,Zisserman, A., Freeman, B., **Discovering Objects and Their Location in Images**, *International Conference on Computer Vision (ICCV 2005)*, October, 2005.

[64] Unnikrishnan, R., Pantofaru, C., Hebert, M., **A Measure for Objective Evaluation of Image Segmentation Algorithms**, *Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '05), Workshop on Empirical Evaluation Methods in Computer Vision*, Vol. 3, June, 2005, pp. 34 - 41.

[65] Yacoob, Y., Black, M.J., **Parameterized modeling and recognition of activities**, *Sixth International Conference on Computer Vision,* Jan 1998, pp. 120-127

[66] Davis, J., Bobick, A., Richards, W., **Categorical representation and recognition of oscillatory motion patterns**. *IEEE Conference on ComputerVision and Pattern Recognition*, pp. 628–635, 2000.

[67] Seitz, S., Dyer, C., **View-Invariant Analysis of Cyclic Motion**, *International Journal of Computer Vision*, Vol. 25, No. 3, 1997, pp. 231-251.

[68] Siskind, J.M., Morris, Q., **A maximum likelihood approach to visual event classification**. In *ECCV-96*, pp. 347–360, 1996.

[69] Zacks, J., Tversky, B., **Event structure in perception and cognition**. *Psychological Bulletin*, 127(1):3–21, 2001.

[70] Eickeler, S., Knsmala, A., Rigoll, G., **Hidden Markov Model Based Continuous Online Gesture Recognition**, in *Int. Conference onPattern Recognition (ICPR),* Brisbane. Aug. 1998, pp.1206- 1208.

[71] Soriano, M.,Huovinen, S.,Martinkauppi, B., Laaksonen, M., **Skin detection in video under changing illumination conditions**, in *Proc. 15th InternationalConference on Pattem Recognition,* 2000, pp. 839-842.

[72] Bobick, A.F., Ivanov, Y. A., **Action recognition using probabilistic parsing**. In *IEEE Conference on Computer Visionand Pattern Recognition*, 1998.

[73] Bobick, A.F., Davis, J.W., **An appearance-based representation of action**. In *IEEE International Conference onPattern Recognition*, 1996.

[74] Hongeng, S., Bremond F., Nevatia, R., **Representation and optimal recognition of human activities**. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume1, pages I–818–I–824, 2000.

[75] S. Park, S., Aggarwal, J.K., **Recognition of two-person interactions using a hierarchical Bayesian network**. In *Proceedings of the IEEE International Workshop on Visual Surveillance*, November 2003.

[76] Hu, M., **Visual Pattern Recognition by Moment Invariants**, *IRE Trans. Information Theory*, vol. 8, no. 2, pp. 179-187, 1962.

[77] Yamato, J., Ohya, J., Ishii, K., **Recognizing Human Action in Time Sequential Images Using Hidden Markov Models**, *Proc. Computer Vision and Pattern Recognition*, pp. 379-385, 1992.

[78] Ayers, D.,Shah, M., **Recognizing human action in a static room**, *In Proceedings ComputerVision and Pattern Recognition,* pages 42-46, 1998

[79] Barron, J.L., Fleet, D.J., Beauchemin, S.S., **Performance of Optical Flow Techniques**, *International Journal of Computer Vision,* 12:1, pp. 43-77, 1994

[80] Black, J.,Makris, D.,Ellis, T.J., **Validation of Blind Region Learning and Tracking,** *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation*, Beijing, China, October, 2005

[81] Boiman, O., Irani, M., **Detecting Irregularities in Images and in Video**, *IEEE International Conference on Computer Vision (ICCV)*, Beijing, October 2005

[82] Brand, M., Kettnaker, V., **Discovery and Segmentation of Activities in Video**, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 8, August 2000

[83] Robertson, N.M., Reid, I.D., Brady, J.M., **What are you looking at? Gaze estimation in medium-scale images***, Proc. Human Activity Recognition and Modelling* , British Machine Vision Conference (BMVC), Oxford, UK, September 2005

[84] Robertson, N.M., Reid, I.D., **Estimating Gaze Direction from Low-Resolution Faces in Video,** *Proc. 9th European Conference on Computer Vision (ECCV)*, Graz, Austria, May 2006

[85] Zhong, H.,Shi, J., Visontai, M., **Detecting Unusual Activity in Video**, *Computer Vision and Pattern Recognition*, Washington D.C., USA, June 2004

[86] Mori, T., Tsujioka, K., Shimosaka, M., Sato, T., **Humanlike Action Recognition System Using Features Extracted by Human**. In *Proc. of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1214–1220, 2002.

[87] Park, J., Park, S., Aggarwal, J.K., **Model-based human motion capture from monocular video sequences**. *International Symposium on Computer and Information Sciences*, 2869, 2003.

[88] Comaniciu, D., Ramesh, V., Meer, P., **Real-time tracking of non-rigid objects using mean shift**. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume 2, pages 142–149, Hilton Head Island, South Carolina, 2000.

[89] Wachter, S., Nagel, H.H., **Tracking of persons in monocular image sequences**.In *Nonrigid and Articulated Motion Workshop*, 1997.

[90] Zarit, B.D., Super, B.J., Quek, F.K.H., **Comparison of Five Colour Models in Skin Pixel Classification**. *International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems,* 58-63, 1999.

[91] Park, S., Aggarwal, J.K., **Segmentation and tracking of interacting human body parts under occlusion and shadowing**. In *IEEE Workshop on Motion and Video Computing*, pages 105–111, Orlando, FL, 2002.

[92] Kim, J.D., Kim, S.D., Kim, J.K., **Fast convergent method for optical flow estimation in noisy image sequences**, *Electron. Lett*. 25 (1) (1989) 74–75.

[93] Lee, H.K., Kim, J.H., **An HMM-Based Threshold Model Approach for Gesture Recognition**, *IEEE Tran. On PAMI,* Vol.21, No.10, Oct. 1999

[94] Jurafsky, D., Martin, J.H., **Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition (2ed.)**, Prentice Hall, 2008.

[95] Jackson, P., **Introduction to Markov Models and HMMs**, *Hidden Markov Model Tutorial Sessions,* Center for Vision Speech and Signal Processing, University of Surrey, 2004

[96] Jackson, P., **Likelihood calculation and Viterbi decoding**, *Hidden Markov Model Tutorial Sessions,* Center for Vision Speech and Signal Processing, University of Surrey, 2004

[97] Jackson, P., **Maximum likelihood re-estimation**, *Hidden Markov Model Tutorial Sessions,* Center for Vision Speech and Signal Processing, University of Surrey, 2004

[98] Jackson, P., **Output probability distribution functions**, *Hidden Markov Model Tutorial Sessions,* Center for Vision Speech and Signal Processing, University of Surrey, 2004

[99] Jackson, P., **Extensions and applications**, *Hidden Markov Model Tutorial Sessions,* Center for Vision Speech and Signal Processing, University of Surrey, 2004

[100] Eisner, J., **An interactive spreadsheet for teaching the forward-backward algorithm**, In *Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching NLP and CL,* pp. 10-18, 2002

[101] Ogawara, K., Takamatsu, J., Iba, S., Tanuki, T., Kimura, H., Ikeuchi, K., **Acquiring hand-action models in task and behavior levels by a learning robot through observing human demonstrations**, *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, September, 2000.

[102] Douglas, B., **State Machines and Statecharts**, *Embedded Systems Conference*, 1999.

[103] Koutsoukos, X., **Finite State Machines**, *Foundations of Hybrid and Embedded Systems Lecture*, 2005

[104] Thacker, N.A., Lacey, A.J., **Tutorial: The Likelihood Interpretation of the Kalman Filter,** *TINA Memos: Advanced Applied Statistics*, 002,1996

[105] Kalman, R.E., **A new approach to linear filtering and prediction problems**,*Transactions of the ASME, Ser. D., Journal of Basic Engineering,* 82, 34-45,1960.

[106] Cuevas, E., Zaldivar, D., Rojas, R., **Kalman filter for vision tracking**, Technical Report B-05-12, Freie Universität Berlin, Fachbereich Mathematik und Informatik, 2005

[107] Krootjohn, S., **Video image processing using MPEG technology for a mobile robot**, Ph.D. Dissertation, Vanderbilt University, August 2007.

[108] Heckerman, D., **A Tutorial on Learning with Bayesian Networks**. In *Learning in Graphical Models,* M. Jordan, ed.. MIT Press, Cambridge, MA, 1999. Also appears as Technical Report MSR-TR-95-06, Microsoft Research, March, 1995.

[109] ____, **Bayesian Networks**, *Techniques in Artificial Intelligence*, MIT Open Courseware, 2007

[110] ____, **Learning With Hidden Variables**, *Techniques in Artificial Intelligence*, MIT Open Courseware, 2007

[111] Tugcu, M., **A Computational Neuroscience Model with Application to Robot Perceptual Learning**, Ph.D. Dissertation, Vanderbilt University, August 2007

[112] Zacks, J. M., **Using movement and intentions to understand simple events**. *Cognitive Science*, 28, 979-1008, 2004.

[113] Levin, D. T., Hunter, J. E., Wilkes D. M., Heaton, C., and Saylor, M. M., **Specifying the looking and reaching actions that predict breakpoint judgments**, *Manuscript in Preparation,* 2008.

[114] CIE (1932). *Commission internationale de l'Eclairage proceedings, 1931*. Cambridge University Press, Cambridge

[115] Gonzalez, R. and Woods, R. E. (2002) *Digital Image Processing,* 2nd ed. Prentice Hall Press

[116] Hunter, J. E., Wilkes, D. M., Levin, D. T., Heaton, C., and Saylor, M. M., **Autonomous Segmentation of Human Action for Behaviour Analysis**, *International Conference on Development and Learning*, Monterey, CA, August 9-12, 2008

[117] Newtson, D. (1973). **Attribution and the Unit of Perception of Ongoing Behavior**, *Journal of Personality and Social Psychology*, 28, 28-38.

[118] Zacks, J. M., Speer, N. K., Swallow, K. M., Braver, T. S., & Reynolds, J. R. (2007). **Event perception: A mind/brain perspective**. *Psychological Bulletin*, 133, 273-293.

[119] Reynolds, J. R., Zacks, J. M., & Braver, T. S. (2007). **A computational model of event segmentation from perceptual prediction**. *Cognitive Science*, 31, 613-643.

[120] Vicon Information Website, http://www.vicon.com/, http://www.vicon.lt/

[121] Qualysis Information Website, http://www.qualisys.com/

[122] Ryoo, M. S. and Aggarwal, J. K., **Observe and Explain: A New Approach for Multiple Hypotheses Tracking of Humans and Objects**, *IEEE Confernce on Computer Vision and Pattern Recognition* in Anchorage, AK, June 2008.

[123] Ryoo, M. S. and Aggarwal, J. K., **Recognition of High-level Group Activities Based on Activities of Individual Members**, *Proceedings of IEEE Workshop on Motion and Video Computing (WMVC)* in Copper Mountian, Co, 2008.

[124] Jun, G., Aggarwal, J.K., and Gokmen, M., **Tracking and Segmentation of Highway Vehicles in Cluttered and Crowded Scenes**, *Proceedings of IEEE Workshop on Application of Computer Vision (WACV)* in Copper Mountian, Co, 2008.

[125]Cohn, J., Kanade, T., Moriyama, T., Ambadar, Z., Xiao, J., Gao, J., and Imamura, H., tech. report CMU-RI-TR-02-06, Robotics Institute, Carnegie Mellon University, November, 2001

[126]Imai, A., Shimada, N., and Shirai, Y., **Hand Posture Estimation in Complex Backgrounds by Considering**, *Proc. of Asian Conf. on Computer Vision (ACCV) 2007*, November, 2007, pp. 596 - 607.

[127]Imai, A., Shimada, N., and Shirai, Y., **3-D Hand Posture Recognition by Training Contour Variation**, *Proc. of 6th Int. Conf. on Automatic Face and Gesture Recognition*, 2004, pp. 895 - 900.

[128] Multi-People Tracking, The Robotic Institute, Carnegie Mellon website, http://www.ri.cmu.edu/research_project_detail.html?type=description&project_id=622&menu_id=261

[129] Wang, X., **A Vision-Based Perceptual Learning System for Autonomous Mobile Robot**, Ph.D. Dissertation, Vanderbilt University, August 2007.

[130] H. M. Kalayeh and D. A. Landgrebe, **Predicting the Required Number of Training Samples**, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, No. 6, pp. 664-666, November 1983.