INTEGRATION OF IMITATION LEARNING WITH COGNITIVE CONTROL FOR A HUMANOID ROBOT

By

Huan Tan

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

August, 2013

Nashville, Tennessee

Approved:

Kazuhiko Kawamura

Douglas Fisher

Richard Alan Peters II

Nilanjan Sarkar

D. Mitch Wilkes

Dedicated to

my beloved wife

and

my parents and my son

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor, Professor Kazuhiko Kawamura, who is not only a great professor in academic and research but also a great mentor in many areas, due to his deep insight, guidance, endless patience and encouragement provided to me throughout my PhD research at Vanderbilt University.

I would also like to thank my committee members, Professor Douglas Fisher, Professor Richard Alan Peters II, Professor Nilanjan Sarkar, and Professor D. Mitch Wilkes for their inspirations, guidance, comments, and suggestions over this dissertation.

I would like to express my sincere thanks to my beloved wife who always provides me support and encouragement to help me throughout my PhD study. I also thank to my parents for the love and support they have provided for my entire life.

I would like to thank all of the students (both past and present) at the Center for Intelligent Systems (CIS) for their help and invaluable friendship. In particular, Stephen Gordon, Juan Rojas, Erdem Erdemir, Xi Luo, Sean Thornton, Tuo Shi, Jing Fan, Yiyuan Zhao, Xue Yang, and Christ Costello. I also would like to thank all my friends at Vanderbilt University for our enjoyable friendship.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Motivation of Work

In the future, it is expected that humanoid robots will be increasingly used in working spaces, homes, public spaces, and other human environments to assist humans or to complete tasks independently. In a complex and dynamic environment, it is necessary for robots to collect useful information from the environment and try to use available resources to complete tasks. This requires robots to make some decisions and find solutions to their encountered problems quickly by utilizing their own knowledge and current environmental information. Thus, a behavior learning and generation system is necessary for robots to learn and generate behaviors in task-relevant situations. In order to realize such a system, we have to consider some issues.

First, we cannot expect that robots can generate behaviors or skills starting from scratch to solve problems totally by themselves. One solution is that humans demonstrate robots behaviors or skills to robots in advance or on site, and let robots generalize behaviors or skills from demonstrations, so that robots will generate necessary behaviors to complete tasks.

Second, working in dynamic task-relevant situations requires robots to collect environmental information and use decision-making mechanisms to find solutions. Acceptable solutions should enable robots to complete tasks without putting themselves or human collaborators in danger and damaging the environment.

Finally, learning is crucial for robots to behave more robust in dynamic working-environment. It is impossible to teach robots everything they need, but it is possible to give robots learning methods, using which they can generalize useful information from demonstrations of humans and use the generalized behaviors or skills to solve similar but slightly different tasks.

Due to the complexities of the mechanisms of humanoid robots, it is difficult to program complex behaviors for robots to use, and it also requires a large amount of time for design and programming. Thus, in order to enable humanoid robots to work independently or work with humans in task-specific situations, it is necessary to find methods for robots to learn required behaviors or skills rapidly either from humans or own exploratory trials [Tan and Kawamura, 2011].

Robotic imitation learning provides a type of tool for robots to learn behaviors or skills from humans[Atkeson and Schaal, 1997] [Billard et al., 2007]. In imitation learning, robots learn behaviors and skills from demonstrations of humans and apply these learned behaviors and skills to similar but slightly different task-specific situations. Some researchers have proposed that imitation learning could be a possible way of teaching humanoid robots simple behaviors or skills rapidly [Schaal, 1999].

With the development of imitation learning, imitation learning research has been gradually formulated into four stages: *what-to-imitate*, *how-to-imitate*, *when-to-imitate*, and *who-to-imitate* [Calinon et al., 2007] [Nehaniv and Dautenhahn, 2002]. *What-to-imitate* deals with the problem of acquisition and representation of demonstrations, *how-to-imitate* tries to find a method to incorporate the demonstration into a policy-making process(partly decision-making), *when-to-imitate* focuses on the regression model and

prediction of the data, and *who-to-imitate* determines the behavior transfer between bodies with dissimilarity [Nehaniv and Dautenhahn, 2007].

After learning and generalizing behaviors from demonstrations, robots need to apply these behaviors to tasks. Current research on imitation learning only tries to generate similar motion trajectories [Ijspeert et al., 2003] or generate similar behavior sequences [Dillmann et al., 2000] in similar task-relevant situations. These research could help robots complete simple tasks. However, when a robot is placed in a largely different task-relevant situation, a new approach is needed.

This dissertation investigate how to generalize common task-relevant features or constraints of demonstrated behaviors, how to store learned behaviors in memories, and how to generate additional behaviors or behavior sequences to complete tasks. The focus of this dissertation is not on the development of a novel algorithm of generating similar motion trajectories. The innovative work of imitation learning in this dissertation is to find a method for robots to generalize observed behaviors in demonstrations and to use learned behaviors in different task-relevant situations.

The goal of cognitive robotics is to generate human-like intelligence for robots which combines perception, action, learning, decision-making, and communication [Kawamura and Browne, 2009]. “Cognitive control” is a construct from contemporary cognitive neuroscience that refers to processes that allow information processing and behavior to vary adaptively from moment to moment depending on current goals, rather than remaining rigid and inflexible [Ragland et al., 2007]. Cognitive control processes include a broad class of mental operations including goal or context representation and maintenance, and strategic processes such as attention allocation and stimulus-response

mapping [Ragland et al., 2007]. Cognitive control provides a method for robots to interact with humans and robots, and executes suitable behaviors in respond to the emergencies and uncertainties in the environment.

This dissertation additionally investigates how to integrate cognitive control with our robotic imitation learning framework, how to implement the integrated cognitive architecture, and how to make decisions of switching tasks, learning new behaviors and executing suitable behaviors or behavior sequences through cognitive control processes.

In all, using imitation learning, robots learn behaviors and skills from humans and generate similar behaviors or behavior sequences in task-relevant situations; using cognitive control methods, robots decide how to use imitation learning framework to complete tasks using learned behaviors or switch tasks according their judgments in dynamic environment.

Contribution of Work

The main contribution of this work will be: (1) a framework to generalize demonstrated behaviors and generate behavior sequences using behavior graph; and (2) an integrated system which combines imitation learning and cognitive control for a humanoid robot to switch strategies to complete tasks. In this dissertation, the generalization of demonstrated behaviors, the storage of generalized behaviors and the description of the relationship among these learned behaviors, and the generation of behaviors or new behavior sequences in new task-relevant situations will be entailed. These methods will be integrated in cognitive control processes for robots to deal with

dynamic situations. Finally, the developed system will be implemented on a cognitive architecture [Kawamura et al., 2008].

Organization of Dissertation

The rest of this dissertation is organized into four chapters: Chapter II provides an overview of fundamental concepts of robotic imitation learning and cognitive control. Robotic imitation learning is divided into behavior acquisition, behavior segmentation, dimension reduction methods, behavior representation methods and behavior generation. Current related research on these topics will be discussed. Cognitive control is to switching strategies to achieve an internal goal, and is related to attention, perception, sensory-motor coordination, memory, decision-making, learning, internal rehearsal, etc. Chapter III explains the integration of robotic imitation learning and cognitive control, as well as how system components are designed. Chapter III also describes how to implement the integrated system using a cognitive architecture. Chapter IV describes three experiments used to test and validate the developed system. Both simulation and experimental results using the ISAC humanoid robot are given in this chapter, as well as the discussion of results. Chapter V quantitatively evaluates the results of the three experiments. Chapter VI concludes this dissertation and proposes future directions of research.

CHAPTER II

OVERVIEW OF COGNITIVE ROBOTICS, COGNITIVE CONTROL AND IMITATION LEARNING

Overview of Cognitive Robotics

The goal of cognitive robotics is to realize human-like intelligence for robots by combining perception, action, learning, decision-making, and communication [Kawamura and Browne, 2009]. Design of a cognitive robot is a systematical integration work [Cassimatis et al., 2004] which includes four important principles: developmental organization, social interaction, embodiment and physical coupling, and multimodal integration [Brooks et al., 1998]. Asada and his colleague proposed that cognitive robots should interactively collect and analyze information to achieve some level of human cognition[Asada et al., 2001].

In the nineties, people realized that it was difficult to create truly artificial intelligence systems because according to Turing's test, machines such as robots should behave intelligently like a real human, but this is impossible due to the fact that the mechanism of human cognition is still not well understood. For example, researchers still do not know how the knowledge is stored in the brain, how evolution processes are happening in the brain, and how billions of neurons work together to plan and complete a complex task. Therefore some researchers think that if the ultimate goal is to make a robot agent interact in intelligent way with people, then such an agent should have as the basis of its experience [Brooks, 1986].

More and more researchers accepted that robotic cognition should be related to the situated environment [Brooks, 1991]. Of course, robotic cognition does not need to duplicate human cognition, but researchers often could obtain inspiration from the research of human cognition.

Since it is impossible to emulate exactly how the brain works on humanoid robots some researchers tried to define the requirement of intelligence for robots. For example, McCathy proposed requirements for human-like robots [McCarthy, 1996]. Sloman analyzed McCathy's statement, combined it with current robotics research and listed several long-term goals for cognitive robots, including: required mechanisms, deeply embodied agents, informationally disembodied agents, species differences, viewer independent affordances, a 'disembodied' grasping concept, perceiving processes, understanding causation [Sloman et al., 2006] [Sloman, 2009].

If we accept these design goals for robotic cognition, cognitive robots should have the following practical skills: attention, perception, sensory-motor coordination, memory, decision-making, learning, internal rehearsal, and interaction. All of the skills will involve certain types of behaviors.

Attention

Attention mechanism in the cognitive process of selectively concentrating on one thing while ignoring others [Deutsch and Deutsch, 1963] [Purdy and Olmstead, 1984] [Driscoll et al., 1998] [Begum and Karray, 2011] .

Perception

Perception includes not only obtaining the images, audio, and other sensory data from the environment, but also extracting useful information from them. Image

processing is a tool for this; however, the goal is to map the sensory data into the task-related data [Crick et al., 2011]. For cognitive robots, the perceptual information is used for the processing in a cognitive way. For example, Chella designed a cognitive architecture for robots, which describes the connection between perceptual system and the cognitive process in robots [Chella et al., 1997].

Due to the imprecision of sensors, disturbances and noises in the environment and systems, it is necessary to estimate the state of the environment and systems using some probabilistic methods. State estimation is done through two types of filter: Bayesian Filter and Kalman Filter.

Bayesian Filter is a commonly used method to find the estimation with the fixed probability [Russell and Norvig, 2003]. It is not an optimum one, but it can give overall understanding of the process. Kalman Filter is an optimum filter to estimate the current state of the process and the future state which can be used in a variety of systems [Kalman, 1960]. It originates from the Bayesian Filter.

Sensory-Motor Coordination

When information in the environment and on the robotic body is collected, cognitive robots should set tight connections between the sensory information and the possible actions [Mataric, 2002]. Sometimes, this type of connections could be established through interactions [Mirza et al., 2007]. Such connections enable robots to response rapidly, especially for the emergency.

Memory

Memory can be divided into two types: Long-Term Memory, and Short-Term Memory.

Long-Term Memory stores procedural, semantic, and episodic memories; Short-Term Memory stores information (mainly environmental information from sensors) for a short-term.

Decision-Making

Decision-Making enables robots to choose a reasonable and suitable action based on the collected information [Estlin et al., 2001].

Learning

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E [Mitchell, 1997] .

Interaction

Humans and robots should interact through some communication mechanisms [Rahimi and Karwowski, 1992] [Trafton et al., 2005] [Goodrich and Schultz, 2007]. The interaction can be implemented through GUI, animations, sounds, touching and other methods [Tan, 2011]. Humans can give orders to robots, and robots can send the feedback to humans. This is the simplest way of communicating. However, for cognitive robots, it is necessary to make the communication more interactive. Especially for robots which can be used among humans [Nakauchi and Simmons, 2002], the interaction between robots and humans are more important. For service robots, robots should also get the feedback from humans [Bien and Lee, 2007].

Development of cognitive robotics largely depends on cognitive architectures. Brooks proposed “Subsumption Architecture” for robotic control systems [Brooks, 1986]. It is a strong connected architecture including several sensory-motor sub-modules

called behavior units. In it, behaviors with higher-level priority can suppress the ones with lower-level priority. All of the information is processed in parallel to provide fast response for robots.

There are several ways to classify the architectures; one is by the approach, the other is by the robotic related information. Cognitive architectures can be divided into three types: Symbolic, Connectionist and Hybrid.

For symbolic type, some well-known architectures are: ACT-R [Anderson and Lebiere, 1998], SOAR [Lehman et al., 1998], EPIC [Keiras and Meyer, 1997], Chrest [Gobet et al., 2001], and Clarion [Sun, 2003]. These architectures use classical AI methods for symbolic computation and manipulation in information processing. Logical methods are used for robots to generate actions through reasoning.

Connectionist type includes: Darwinism [Edelman, 1987] and CAP2 [Schneider, 1999]. These architectures construct a large inter-connected network to process the sensed data and generate actions.

Many modern researchers have begun to recognize the need of both deliberative interaction and reactive interaction for cognitive robots, which motivates the research on hybrid architectures [Kawamura et al., 2004]. Hybrid type includes: Subsumption [Brooks, 1986], RCS [Albus and Barbera, 2005], CHIP [Shrobe and Wilson, 2006], JACK [Winikoff, 2005], and ISAC [Kawamura and Browne, 2009].

With the skills and mechanism described above, a cognitive robot is able to interact with human beings and other robots in a dynamic environment. Brooks [Brooks et al., 1998] strongly stresses the importance of embodiment. Beer proposed a field based method to solve the interaction between robots and the environment [Beer, 2000]

[Erlhagen and Bicho, 2006]. Khatib also used the potential field-based method to solve the interaction problem [Khatib et al., 2001]. Interaction requires cognitive robots not only to utilize the information in the environment but also to have abilities to communicate with humans and other robots [Fong et al., 2003]. Embodiment not only lies in the software but also in the hardware [Pfeifer et al., 2007].

Overview of Cognitive Control

"Cognitive control" is a construct from contemporary cognitive neuroscience that refers to processes that allow information processing and behavior to vary adaptively from moment to moment depending on current goals, rather than remaining rigid and inflexible [Ragland et al., 2007]. Cognitive control processes include a broad class of mental operations including goal or context representation and maintenance, and strategic processes such as attention allocation and stimulus-response mapping [Ragland et al., 2007]. Cognitive control provides a method for robots to interact with humans and robots, and executes suitable behaviors in respond to the emergencies and uncertainties in the environment.

Recent research on cognitive control for robots focuses on regulating robotic behaviors by analyzing sensory information [Burattini et al., 2011], mapping between action observation between appropriate complementary actions [Bicho, 2012], behavior control based on Self-Awareness [Gorbenko et al., 2012], control task planning [Insaurralde et al., 2012], sensorimotor coordination [Maye and Engel, 2011], optimization for multiple control task [Karaoguz et al., 2011], etc.

In order to implement cognitive control, robots need to learn the knowledge about control strategies first. Reinforcement Learning [Khamassi et al., 2011], ANN [Sánchez Boza et al., 2011] [Hamker, 2012], etc. for robots to learn strategies.

Cognitive control architectures still receive broad attention from cognitive robotics community [Malfaz et al., 2011] [Munoz et al., 2011] [KangGeon et al., 2011] [Huntsberger, 2011] [Conforth and Yan, 2011] [Glas et al., 2011] [Haazebroek et al., 2011].

A cognitive control framework was developed in our lab [Kawamura and Gordon, 2006] and implemented as a part of a cognitive robotic architecture in 2008 [Kawamura et al., 2008]. Cognitive control functionalities implemented were attention control, working memory and internal rehearsal.

SES, inspired by the egosphere concept defined by Albus [Albus, 1991], serves as a spatio-temporal Short Term Memory for a robot [Peters II et al., 2001]. STM handles sensor-based percepts that are assigned the focus of attention or gating by the Attention Network [Hambuchen, 2004]. Perceived sensory inputs that have a high emotional salience, i.e. task-related chunks, will cause the Attention Network to post them to Working Memory System (WMS). WMS stores task-related information [Kawamura et al., 2004] [Dodd, 2005]. Internal Rehearsal is a tool for the decision-making. Robots do not need to apply the actions in the real environment. They can estimate the results using the established action model and the environment model. The decisions are making before actions using internal rehearsals[Lee and Thompson, 1982] [Erdemir et al., 2008]. This is different from simulation because simulation is only a display or duplicate of the real experimental process. Additionally, robots can estimate the results of its actions so as

to improve performance. In some circumstances, the robot could even use internal rehearsal to learn how to perform new tasks. Such functions cannot be completed by simulations.

Overview of Imitation Learning

In the past, robots could do little without programming by the operators. In order for robots to generate behavior, researchers pre-designed behaviors for robots [Mahadevan and Connell, 1992] [Brooks et al., 1999]. Generated behaviors are then modified through machine learning process. Beginning from the 1970s, researchers started to explore possible methods to teach a robot to learn knowledge, skills, and behaviors.

One robot learning area is in which robots learn mappings from states to actions, called a policy, enabling a robot to act based upon the current state. A particular promising approach to policy learning is called Imitation Learning (IL), also referred to as Learning from Demonstration (LfD) or Programming by Demonstration (PbD) [Argall, 2009]. In imitation learning, after observing demonstrations, robots generate reasonable solutions to solve similar problems by either searching a good solution or eliminating a bad solution in the knowledge base [Billard et al., 2007]. This method was found efficient for transferring simple skills from human to robot.

The first exploration in imitation learning was by Uchiyama who trained robots to learn motion patterns through trials [Uchiyama, 1978]. The subsequent research focused on teaching robots to learn skills or transferring skills between robots.

Atkeson and his college in 1986 proposed a prototype of imitation learning system and trained robots to learn motions through practice [Atkeson and McIntyre, 1986]. He started the research in 1980s to teach the robot to move its arm following the demonstration as close as possible. This is considered to be a significant prototype of the modern imitation learning.

Earlier in the 1990s, imitation learning still focused on the learning of the dynamics of robotic control system [Horowitz, 1993]. Atkeson's famous inverted pedulumn experiment started in the 1990s [Atkeson and Schaal, 1997], in which robots were shown a demonstration of swing-up a pendulum. Robots then tried to modify the control parameters and control policies to balance it in a upright position with similar trajectory to the demonstration as shown in Figure 1.

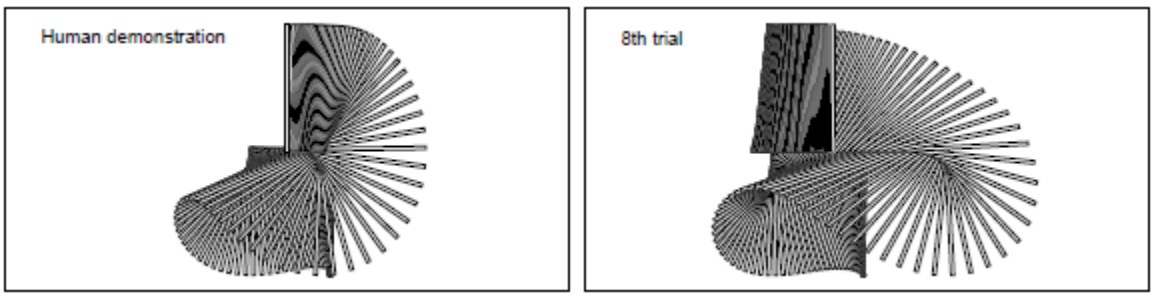


Figure 1 Swing-Up Experiment [Atkeson and Schaal, 1997]

Imitation learning can be implemented in two stages: Learning from Demonstration, and Generation of Behaviors. Typically, an imitation learning process consists of the following steps: 1. A human teacher or a robot demonstrates how to complete a task to another robot (often as a form of a behavior sequence) and another robot records the behavior sequence and segment it into a set of behavior primitives; 2. Given a similar but different task, the robot generates the same behavior sequence or the same behaviors to complete this task.

Current research on imitation learning can be divided into two categories [Calinon et al., 2007]: one is trying to train robots to extract and learn the motion dynamics [Ijspeert et al., 2003], and the other is trying to train robots to learn higher-level behaviors and action primitives through imitation [Dillmann et al., 1995] [Mataric et al., 1998].

Since the 1990s, researchers began to develop humanoid robots (such as MIT Cog [Brooks et al., 1999], NASA Robonaut [Ambrose et al., 2000], Aldebaran's NAO [Gouaillier et al., 2009], Vanderbilt ISAC [Kawamura et al., 2008] and others. Because there are many similar physical characteristics between humanoid robots and humans, it is also expected that humanoid robots allow to investigate complex truly human-like intelligence [Atkeson et al., 2000]. Two problems have been emerged: one is how to program the motions of a humanoid robot which has many degrees-of-freedom [Kuffner Jr and LaValle, 2000] [Yang et al., 2006], and the other is how to implement human-like intelligence on this robotic platform. The first problem lies in the control area and the latter lies in the AI area.

Schaal proposed that imitation learning is a possible solution for the first problem [Schaal, 1999]. Billard analyed the cognitive process of behavior generation in the human brain and presented a biologically inspired model for motor skill imitation [Billard, 2001].

Billard stated that robots should learn the skills from multiple demonstrations, extract the common feature from the demonstratoins, and reproduce the skills in different situations [Billard et al., 2007].

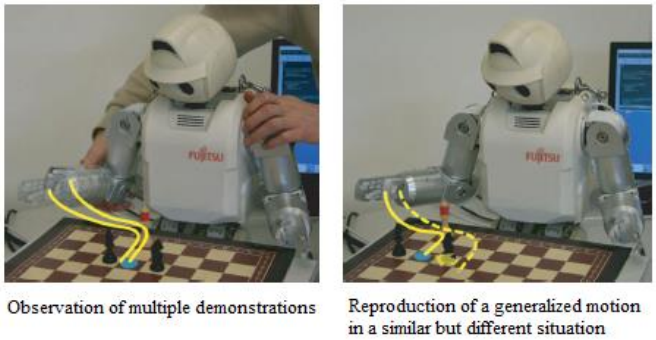


Figure 2 Reproduction in Different Situations [Billard et al., 2007]

Gradually, imitation learning research has been formulated into four stages: *what-to-imitate*, *how-to-imitate*, *when-to-imitate*, and *who-to-imitate* [Calinon et al., 2007] [Nehaniv and Dautenhahn, 2002].

What-to-imitate deals with the problem of acquisition and representation of demonstrations, how-to-imitate tries to find a method to incorporate the demonstration into a policy-making process(partly decision-making), *when-to-imitate* focuses on the regression model and prediction of the data, and *who-to-imitate* determines the skills transfer between bodies with dissimilarity [Nehaniv and Dautenhahn, 2007].

In the *what-to-imitate* stage, demonstrations are represented by mathematical models: Fuzzy method[Dillmann et al., 1995], Hidden Markov Model (HMM) [Yang et al., 1997], Locally Weighted Regression [Atkeson et al., 1997], Gaussian Process [Wang et al., 2008], and Gaussian Mixture Model (GMM) [Calinon et al., 2007]. Because of the high DOFs, dimension reduction is applied before the establishment of these mathematical models through Principal Component Analysis (PCA) [Wood et al., 1987; Calinon et al., 2007], Factor Analysis[Bartholomew, 1984], Principal Curves[Tibshirani, 1992], ISOmap [Jenkins and Matari 2004] [Jenkins and Matari 2004], or Locally Linear Embedding (LLE) [Roweis and Saul, 2000].

In the *how-to-imitate* stage, Dynamic Movement Primitives (DMP) [Ijspeert et al., 2003] is widely used, and Calinon proposed a lagrange-equation method to compute solutions [Calinon et al., 2007].

In the *when-to-imitate* stage, the data prediction is based on the models used in *what-to-imitate*. Additionally, if dimension reduction is applied in *what-to-imitate*, data reconstruction should be taken into consideration. Bishop proposed the Generative Topographic Map (GTM) to train the recontruction matrix [Bishop et al., 1998]. Lawrence proposed Gaussian Process Latent Variable Model (GPLVM) by using the Expectation-Maximization (EM) method in PCA to interatively compute the reconstruction matrix [Lawrence, 2005].

who-to-imitate is still under investigation.

From a perspective of system implementation, we can also divided the imitation learning framework in to following parts: demonstration acquisition, behavior segmentation, dimension reduction, behavior representation, and behavior generation.

Demonstration Acquisition

This is the first stage of skill learning by imitation. In this stage, robots record the demonstrated motions to complete certain tasks. Demonstrations are given by humans or other robots. As shown in Figure 3, robots could use a variety of devices to sense and record the demonstrated motions.

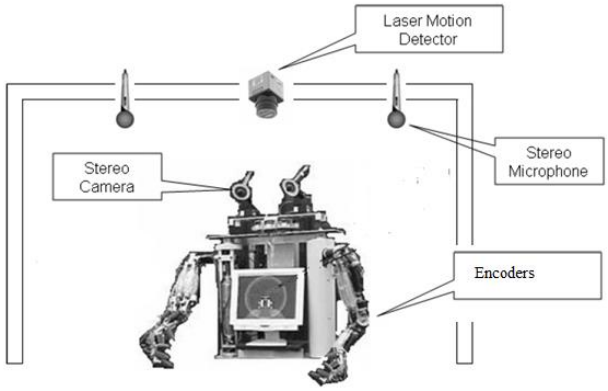


Figure 3 Robot Senses the World [Kawamura et al., 2008]

--*Recording from Observing*

The method in which robots use cameras to observe the demonstrations from humans is the most common motion recording method [Bentivegna and Atkeson, 2001] [Ogawara et al., 2003] [Ude et al., 2004].

Kuniyoshi proposed a general framework for robots to learn reusable knowledge through observation [Kuniyoshi et al., 1994].

The observed information was gathered by cameras and sent to the vision server for visual and action recognition. The recorded information of the demonstration was the movement of the hand of humans and the movement of the objects.

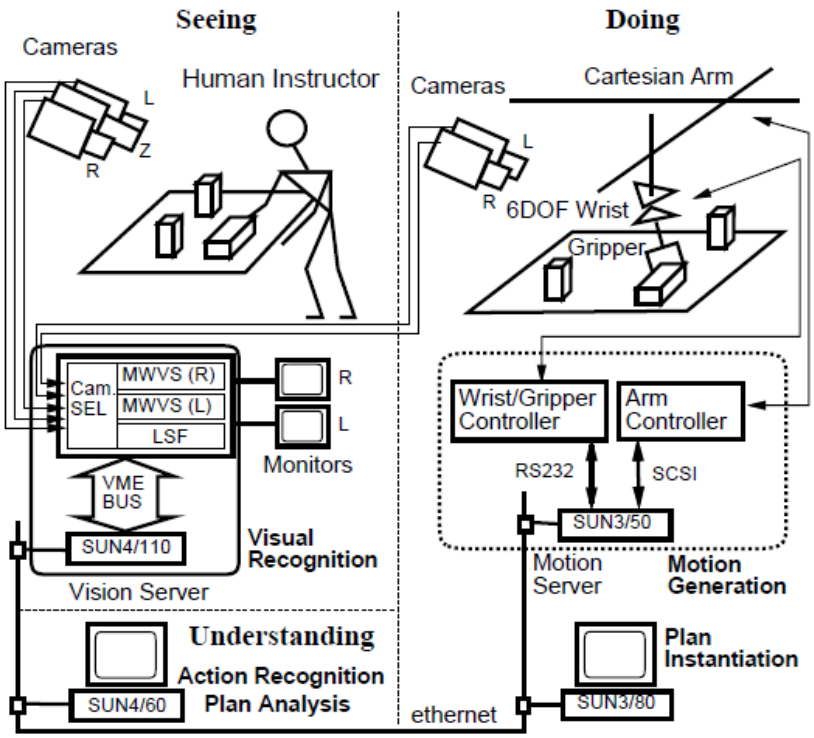


Figure 4 A Learning from Observation Framework [Kuniyoshi et al., 1994]

In [Yeasin and Chaudhuri, 2000], robots not only tracked the objects in the environment, but also tracked the hand, and in [Mataric, 2002] the vision information was straightly coupled with motor information.

In our lab, Begley used two cameras to obtain the demonstrations [Begley, 2008]. Thronton also used this method [Thornton, 2009]. Using two cameras, robot could obtain the positions in a X-Y plane and the depth information. In Begley's method, the movement of a object in a human's hand was tracked and in Thronton's method, the movement of a hand of a human was tracked.

--*Recording through Manipulating*

In traditional programming of robotic motions, operators use teach-pendants or other devices to send control information to robots. The manipulation information is

recorded by a device for future use. In [Inamura et al., 1999], a human teacher used a joystick to demonstrate how to complete a task.

--*Encoder Recording*

There are many sensors on robots, e.g., encoders on the joints. Calinon and Billard trained a robot to learn how to complete a Chess-Moving task called Bucket task by manually moving the arm of the robot [Calinon et al., 2007]. The angular information is recorded and used to compute the position and orientation of end-effectors.

--*Recording through Sensing the Forces*

Force information has been used for robots to record the demonstration information from human teachers. In [Skubic and Volz, 2000], human teachers manually moved the end-effector on the arm of a robot and robot computed the trajectory of movement using the recorded force information.

--*Recording using Sensors on Humans*

In order to reduce noises and obtain better recognition or tracking results, researchers put sensing devices on human teachers, the movement of which is designed to be easily captured by cameras or other electronic sensor devices. In [Voyles and Khosla, 2001], sensors were mounted on the hand of a human, and the movement information is recorded by the sensors on the hand of a human. In [Ijspeert et al., 2002], sensors were mounted on the arm and the body of a human teacher, and the robot has been shown a demonstration of hitting a ball using a racket.

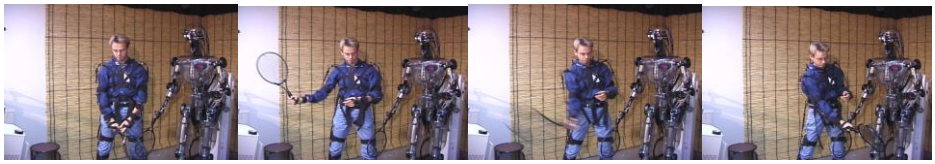


Figure 5 Sensing using Sensors on the Human Body [Ijspeert et al., 2002]

Behavior Segmentation

Simple tasks can be completed by several behaviors in a behavior sequence. Therefore, it is reasonable to segment the behavior sequence into several primitive behaviors [Tan, 2012].

--*Hidden Markov Model (HMM)[Rabiner, 1989]*

The sensed information from the demonstrations is both temporal and spatial. Therefore, it is necessary to find a mathematical model to describe it temporally and spatially. In robotics research, HMM is widely used for behavior segmentation either shown by human teachers or executed by robots [Pook and Ballard, 1993] [Hovland et al., 1996] [Rybski and Voyles, 1999] [Wilson and Bobick, 1999] [Inamura et al., 2003] [Herzog et al., 2008].

Yang et al. used a HMM model to segment the demonstrated gestures [Yang et al., 1997]. Rybski used this method to segment the behavior of a mobile robot [Rybski and Voyles, 1999]. In [Pook and Ballard, 1993], behaviors were also segmented using HMM models. The difference between Pook's method and Yang's method is that Pook segments the measured behaviors in the joint space. In Yang's method, the 6-dimensional information has been combined into 1-dimension.

The advantages of the HMM method are: robust, flexible, extensible, and easy to implement. The disadvantage of HMM method is that the number of states in a behavior sequence should be predefined. If a new behavior sequence is generated, it is difficult for designers to determine how many states there are in the behavior sequence.

--*Extension of HMM Method*

In order to avoid the disadvantages of the HMM method, many researchers proposed some extensions of the original HMM method. The basic idea is to modify and improve the segment criterion or the representation of states.

--*Gaussian Mixture Model (GMM)*

Billard used GMM [Chernova and Veloso, 2007] to describe the states to construct the HMM model [Billard et al., 2006]. The basic idea is that different segmented parts in the demonstration have different shapes of distributions if GMM is used to describe the distributions of the sampled data points.

As shown in Figure 6, there are eight Gaussian models in the sampled data points, each model having different means and variants. Therefore, the states can be described upon the means and variants.

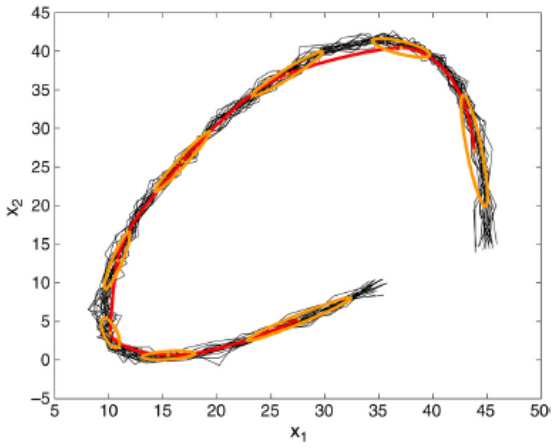


Figure 6 Segmentation Results using a GMM Based HMM Model [Billard et al., 2006]

--*Cluster Based Method (CBM)*

Kulic and Nakamura proposed a segmentation method based on the optical flow in the environment, which is a cluster based method [Kulic et al., 2008]. This method tries to find the similarity between two HMM models. If the Kullback-Leibler distance

calculated from is larger than a predefined value, it is considered as a new behavior sequence and inserted into the tree as a node. Otherwise, the demonstration is segmented based on the model which has the smaller Kullback-Leibler distance to it.

--*Fuzzy Method*

Dillman applied the Fuzzy method in segmenting the behaviors in [Dillmann et al., 1995]. The advantage of Fuzzy method is that it is not necessary to predefine the number of states in a behavior sequence of a demonstration. However, the disadvantage is that it is easy to be affected by the noise. Although some noise reduction methods can be used to avoid this bad influence, it is still not easy to use this method in a practical environment.

Dimension Reduction

The recorded demonstrations are sampled points on the trajectory in data arrays. There are many DOFs on humanoid robots. Therefore, the collected data points have high dimensions [Bitzer et al., 2009].

Dimension reduction methods, which project the data from a higher-dimensional space to a lower-dimensional space [Sammon, 1969] [Kambhatla and Leen, 1993] [Carreira-Perpinán, 1997] [Rahman and Xu, 2004] [Benner et al., 2005] are widely used in climate analysis and control, oil data analysis, population analysis, and geographic analysis [Carreira-Perpinán, 1997] [Bishop et al., 1998], and can provide a visual representation of the data in the 2-D or 3-D space.

The data come from a single inner space, which is normally called the latent space. Dimension reduction methods can extract the internal relationships or features of the sampled data points. This is especially useful for robotic research [Bitzer et al., 2008]

[Bitzer and Vijayakumar, 2009]. In robotic imitation learning, it is necessary to find the internal features of the demonstrated behaviors, because the goal of the imitation learning is to learn the key features of the demonstrations [Calinon et al., 2007].

Suppose that we have a collected data set $\boldsymbol{Y}$. $\boldsymbol{Y}$ is represented as a matrix: $\boldsymbol{Y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_n)^T$. $\boldsymbol{y}_i$ in this data set is a p-dimensional vector and can be represented as a row vector: $\boldsymbol{y}_i = (y_{i1}, y_{i2}, \dots, . y_{ip})$ .The objective of dimension reduction is to obtain a data set in the low-dimensional space, named the latent space where obtained data set is $\boldsymbol{X} = \{\boldsymbol{x}_i | i = 1,2, \dots, n\}$. Each element $\boldsymbol{x}_i$ is a q -dimensional vector and can be presented as $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \dots, . x_{iq})$, where $q \ll p$. Normally, $\boldsymbol{X}$ is represented as a matrix $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n)^T$.

--*Principal Component Analysis (PCA)*

The basic idea of PCA is to reduce the dimensionality of a data set inside to find out variables are related to each other, while keeping as much as possible the variations lie in the original analyzed data set [Jolliffe, 2002] [Kambhatla and Leen, 1997] [Schölkopf et al., 1998] [Belkin and Niyogi, 2003].

PCA is widely used for representing the distribution of data on a 2-dimensional plane or finding the internal features of the sampled data points [Li and Wang, 2002] [Verbeek et al., 2002]. Calinon used this method to represent the sampled position values, joints values, hands-object relationship values using a humanoid robot [Calinon et al., 2007].

Tipping and Bishop extended the classical PCA method to a probabilistic method[Tipping and Bishop, 1999]. If the sampled data points are represented by using

kernel functions in the original data space, the classical PCA is extended to the kernel PCA [Williams, 2002].

--*Factor Analysis (FA)*

The PCA method tries to find the principal components to keep the information, such as the variance, as much as possible. However, in the FA, a model is established first and the target is to estimate the variance of the factors in the model [Bartholomew, 1984].

--*Topology Methods*

Topology methods analyze the sampled data points (in the original high-dimensional space) by their internal connections and place the data in suitable positions in the latent space. The distributions of the data points in the latent space reflect the internal topology of the data points in the original space.

Roweis proposed Locally Linear Embedding (LLE) [Roweis and Saul, 2000] to utilize the neighborhood information to reduce the dimension of the sampled data points. The target of this method is to construct a topological manifold and the distribution of the data points on this manifold in the latent space that is strongly related to the distribution of the neighbors.

--*Multi-Dimensional Scaling (MDS)*

MDS [Mardia et al., 1980] [Šarić et al., 2011] is a method to analyze the dissimilarities between data points. MDS can be divided into the Metric (Classical) MDS and the Non-Metric MDS. The Metric MDS analyzes the distance between the data points and constructs a distance matrix. The distribution of the data points in the latent space reflects the chief spatial topology of the original data set.

--*ISOMAP*

ISOMAP is an extension of the Metric MDS. It constructs the distance matrix by connecting the neighbors [Tenenbaum et al., 2000] [Tan et al., 2012]. That means, a neighbor distance matrix is constructed first, and then the overall distance matrix is computed by connecting the neighbors and adding the distances between the neighbors.

--*Self-Organizing Map (SOM)*

SOM is a recursive method, which analyzes the similarity of the data points and automatically places them into a 2-dimensional latent space along with their mutual similarity [Kohonen, 1982] [Kohonen, 1990] [Kohonen et al., 2002].

--*Generative Topology Mapping (GTM)*

An important problem in dimension reduction is how to project the data points from the latent space back to the original high-dimensional data space. Only PCA can provide the function of the projecting the data points from the original data space to the latent space, and from the latent space back to the original data space. Bishop proposed the Generative Topology Mapping (GTM) [Bishop et al., 1998], which constructs a probabilistic mapping from the latent space to the original data space.

Behavior Representation

The behavior representation methods are strongly related to the targets of the imitation learning. If the target of the learning is to imitate demonstrated behaviors, the behaviors are often described as components, policies or rules in a decision making architecture or mechanism; if the target of the learning is to generate behavior sequences which are similar to the demonstrations, the behavior sequences are often represented as semantic knowledge, etc., and the behaviors are often represented as semantic

descriptions, components, etc.; if the target of the learning is to generate motion trajectories and the dynamics of the motion trajectories are similar to the demonstrations, the motions are often represented as mathematical regression models or probabilistic models in a path planning module or mechanism.

Behaviors are often represented as the raw data to generate motions [Kawamura et al., 2008]. These motions can be considered as sequence of sampled points along trajectories in Cartesian, Joint space, etc.. Some mathematical models are needed to represent the motion trajectories using some mathematic models. We divide the current representation methods into several categories: probabilistic methods, semantic methods, learning methods and others.

--*Probabilistic Methods*

Gaussian Process (GP)

GP describes the probabilistic distribution over functions. Forte and Ude used GP regression models to teach a humanoid robot to learn reaching [Forte et al., 2010].

Gaussian Mixture Model (GMM)

Calinon and Billard used the GMM method to represent motion trajectories in robotic imitation learning [Calinon and Billard, 2007]. This method is applied in the research of Billard's group for grasping experiments [Sauser et al., 2011] [Shukla and Billard, 2012] .

In our lab, Erdemir used GMM models to describe the objects in the environment [Erdemir et al., 2008]. As shown in Figure 7, the robot is required to reach the objects on the table.

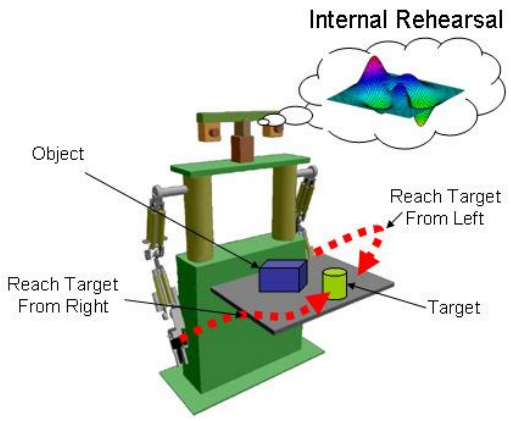


Figure 7 ISAC Simulator and Internal Rehearsal [Erdemir et al., 2008]

First, ISAC tried to reach points in the environment. As shown in the left picture of Figure 8, when the end-effector collides with an object, robot records the position of the collision in the environment. These collision points can be described using a GMM model as shown in the right picture of Figure 8. The target is a 'hole' in the GMM model and the objects are represented as impedances.

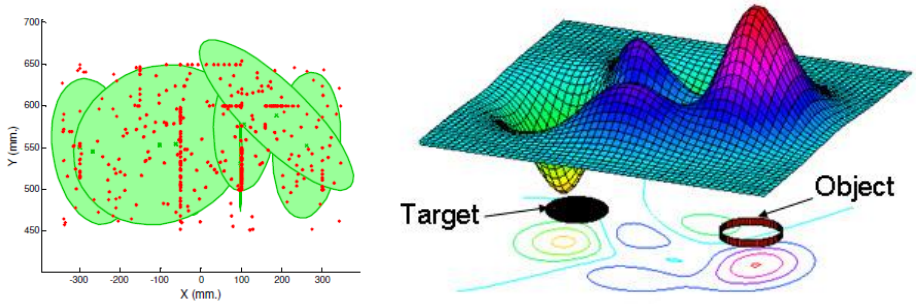


Figure 8 Environmental Modeling using GMM [Erdemir et al., 2008]

--*Hidden Markov Model (HMM) Based Method*

Billard and Calinon proposed that the motions of the behaviors can also be described as a HMM model [Billard et al., 2006]. The HMM method was also used by Inamura in [Inamura et al., 2003]. In this paper, Inamura tried to train a robot to learn the

demonstrated body gestures as shown in Figure 9. Vakanski also applied HMM in robotic imitation learning [Vakanski et al., 2012].

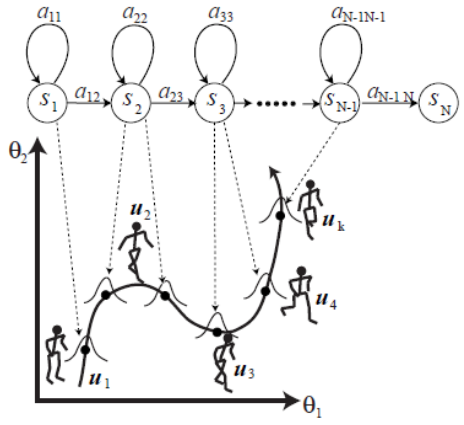


Figure 9 A HMM Model for Representing Human Gestures [Inamura et al., 2003]

--*Semantic Methods*

Robots often need to execute a sequence of behaviors (or actions) to achieve the task. In such situations, it may be easier to represent behaviors as symbols or semantic words/phases in sequences. Such behaviors are often called behavior primitives and robots should learn how to assemble behavior primitives into behavior sequences to complete a task.

--*Behavior Sequences*

Many researchers try to represent behavior primitives as semantic names with parameters [Tan, 2012]. In Rybski's research, primitives are predefined with corresponding descriptions about the specific actions with parameters [Rybski and Voyles, 1999]. The demonstrated behaviors are described as a sequence of behavior primitives and sequences. Bentivegna and Atkeson also used similar methods to represent the behavior primitives and to teach a robot to learn to play table hockeys [Bentivegna

and Atkeson, 2001]. Mataric used a sequence learner for robots to learn behavior sequences from human demonstrations using HMM methods [Amit and Matari, 2002].

--*Behavior Graph*

Arikan used simple nodes to represent the behavior primitives in a behavior graph [Arikan and Forsyth, 2002]. The behavior sequence is represented by connecting the nodes. The connected nodes are assembled in a temporal order and robots understand that the behaviors are in a behavior sequence. The lengths of the node in the lower assembled behavior sequence reflect the required time of the execution for the behavior primitive.

Mataric used a distributed graph to describe the traveled path a robot has learned [Mataric, 1992]. Nicolescu also used simple symbols to represent the behavior primitives [Nicolescu and Mataric, 2003]. In his method, several demonstrations are given to complete the same task. If two demonstrations are given, a generalized behavior sequence is to combine them together while keeping the existing time constraints. The robot only keeps the list that embeds the longest of the possible subsequences. In Ogawara's method, several demonstrations are given and robots need to find the common essential segments from all the demonstrations [Ogawara et al., 2003]. Each segment is represented as a mean and a variance of the positions of the sampled points on the trajectory. Steil describes the representation of the demonstrated behavior as a behavior sequence [Steil et al., 2004]. Additionally, the segments in the behavior sequence described are not only the movement of the arm but also the actions of other actuators. Salem used a chunked verbal method to represent the behaviors in a behavior sequence[Salem et al., 2011]. Muench tried to teach a robot to pick and place an object in the working space [Muench et al., 1994]. The observed behaviors are represented as a tree. Similar applications are found in

[Pardowitz et al., 2007] and [Kulic et al., 2008]. Tenorth used a hierarchy graph to describe the learned behaviors [Tenorth and Beetz, 2009]. Each behavior in his method is described a combination of several logic operations.

--Learning Methods

Artificial Neural Network (ANN)

An artificial neuron receives the weighted input and generates an output according to the following equation:

$$y = f\left(\sum_{i=1}^{n} w_i x_i\right) \quad (1)$$

where x_i is the input of the neuron, w_i is the related weight, and f is an activation function. The neurons are connected and placed in multiple layers as shown in Figure 10. A hidden layer is designed inside this ANN model. The neurons in each layer receive weighted inputs and generate corresponding outputs. The training process is to use the current data set to modify the weights.

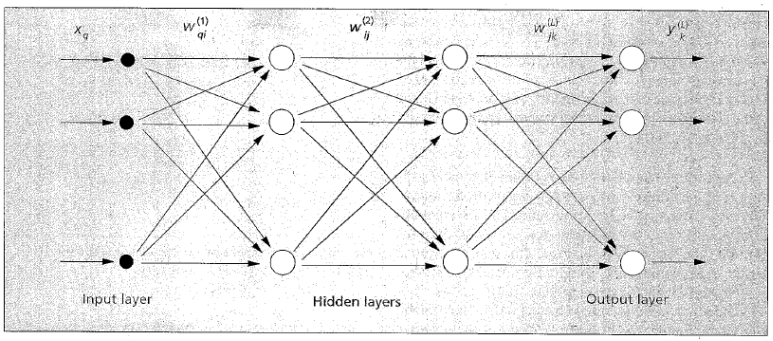


Figure 10 A Multi-Layer Neural Network Model [Jain et al., 1996]

Normally, the Back-Propagation algorithm is applied to learn the weights in an ANN model [Marsland, 2009].

Tani applied ANN to teach a robot to learn gestures from humans [Tani, 2003] [Tani et al., 2004]. He described the behavior sequence as a reactive policy making mechanism.

Nonmonotonic Neural Network (NNN)

In 1990s, Morita proposed an extension of ANN, called NNN to represent sequential patterns [Morita, 1993] [Morita, 1994] [Morita, 1996]. Kuniyoshi used NNN for robotic imitation learning [Kuniyoshi et al., 2003] [Nabeshima et al., 2006]. In Kuniyoshi's method, the states are represented as the combination of the sensory neurons, where environmental states are stored, and the motor neurons, where the motor data are stored. After learning, when the system observes a similar visual motion, it is encoded as a temporal sequence of visual feature vectors. It drives the network state close to the previously learned trajectory attractor. The state is trapped into the attractor and moves along it. Then a NNN model is constructed to record the sequential information about the demonstration, which includes the environmental information and the motor information.

--*Topology Coordinate Space Method*

Topology coordinates is the basic representation used for the control of tangling motions [Ho et al., 2010]. There are three attributes within this method: writhe, center and density.

--*States-Actions Coupling*

In Kaiser and Dillman's method, the trajectory is considered as a pair description between a function of current state $x(t)$: $C\big(x(t)\big)$,and the corresponding actions state $u(t)$ [Kaiser and Dillmann, 1996]. Given a state $x(t) = \{x(t-d), x(t-d-$

$1), \ldots, x(t-d-p)\}$, the target is to use the sensed data in the demonstrations to train the Radial-Basis Functions (RBF).

--*Regression Methods*

The general idea of the *Locally Weighted Regression (LWR)* [Atkeson et al., 1997] is to find the distance function between the query point and input vector of the data points. This method has been extended to Receptive Field Weighted Regression (RFWR) [Atkeson et al., 1997] and Locally Weighted Projection Regression (LWPR) [Vijayakumar and Schaal, 2000] for modeling the non-linear functions and noisy models, in which the distance function $d(\boldsymbol{t}_i, \boldsymbol{t}_e)$ is modified to calculate the distance between the enquiry point and the center of the radial functions scattered in the space. Ijspeert [Ijspeert et al., 2003] [Gams et al., 2009], Schaal, and Theodorou [Theodorou et al., 2010] [Theodorou et al., 2010] applied this method to represent the learned behaviors.

--*Genetic Programming Method*

In An's research, dynamic of motion in the demonstration are represented using Genetic Process (GP) [An et al., 2007] models.

--*Crucial Points Methods*

Because the noises or errors always exist in the demonstrations, some researchers try to use minimum data points to describe the demonstrated motion trajectory.

Miyamoto [Miyamoto and Kawato, 1998] used "Via-Points" method to teach a robot to learn hitting a tennis ball. The number of "Via-Points" is predefined, and the algorithm is to find certain number of points. By connecting the "Via-Points", the error between the demonstrated trajectory and the required generalized trajectory should be minimized. Chen [Chen and Zelinsky, 2003] used the same method to represent the

demonstrated behavior by recording the crucial points in a configuration space. In his experiment, a robot is taught to put the spindle into the support. The demonstration trajectory is also segmented by finding the crucial points.

In robotic control, trajectories are represented as sampled points on the trajectory. The required trajectory is generated by connecting these sampled points.

--*Tunnel Methods*

Because of the measurement error caused by the sensor or the manipulation error caused by humans, trajectories of demonstrations of a task are not always exactly the same. Therefore, some researchers tried to describe the required trajectory as a 'tunnel' and the demonstrated trajectories lie in this tunnel [Delson and West, 2002].

Delson proposed that the description of the demonstrated trajectories is a region which includes the observed trajectories [Delson and West, 2002]. Naksuk [Naksuk et al., 2005] used a similar method by defining boundary to restrict the demonstrated trajectory in a small region. Brock used the same method by expanding the tunnel with a certain parameter δ [Brock and Kavraki, 2001].

Behavior Generation

There are two types of behavior generation methods: one is to generate behaviors exactly the same as the demonstrations and the other is to generate similar behaviors in a similar but slightly different environment. Both methods can be applied in three situations: first is to imitate same behaviors (replicating methods), second is to generate behavior sequences, and the third is to generate similar movement trajectories of behaviors (adapting methods).

--*Task Description*

At the generation stage, a task is given by humans and robots need to understand given commands to generate required behaviors and their related parameters. A typical method is to use natural language processing methods to extract essential units from voice commands based on predefined grammars.

Simmons and Apfelbaum used the Task Description Language to describe given tasks [Simmons and Apfelbaum, 1998]. In their method, tasks are defined by using constraints and tags. In a given command, robots extract these constraints by matching the predesigned grammars.

--*Replicating Methods*

The main goal of the generated motion trajectories in imitation learning is that the distances between the generated trajectories and the demonstrated trajectories are minimized or that the dynamic characteristics of generated trajectories are similar to the demonstrated trajectories based on some metrics. Humans show robots trajectories, and robots record them. Robots generate same movement as the demonstrations by driving the joints or end-effector strictly following the demonstrated trajectory. This method has been applied in many imitation learning cases where robots only imitate the demonstrated movements in the same situation.

Figure 11 displays a diagram of imitation learning proposed by Bentivegna and Atkeson[Bentivegna and Atkeson, 2001]. The “learning from observation” module segments the observed behavior into predefined primitives, and the segment criterion is flexible. This segmented data is then used to provide the encoding for the primitive selection, sub-goal generation, and action generation modules. When a new situation is

given, robots extract the current context from the observation (normally the current environmental information of the task-relevant situation,) and compare the obtained context to the stored contexts. A nearest neighbor lookup process is used to find the context similar to the current context and its corresponding behavior primitive is selected to complete the task.

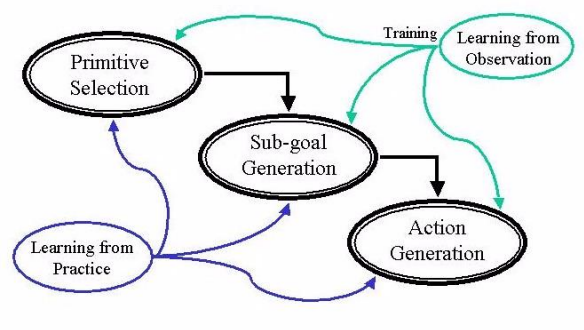


Figure 11 Imitation Learning of Reactive Primitives [Bentivegna and Atkeson, 2001]

Using this framework, Chernova trained a mobile robot to learn behaviors [Chernova and Veloso, 2007]. The policies are represented as a Gaussian Mixture Model (GMM). When robot is placed in the environment, it needs to select a policy which is related to its current situation context. In our lab, Begley and Thronton applied this method for gesture imitation [Begley, 2008] [Thornton, 2009]. Tracked movements of an object or a hand have been transformed to the coordinates of the ISAC robot. ISAC then generate the same movements by following the transformed movements in its own coordinates.

Arikan used a simplified Genetic Algorithm (GA) to generation motions for a robot [Arikan and Forsyth, 2002].

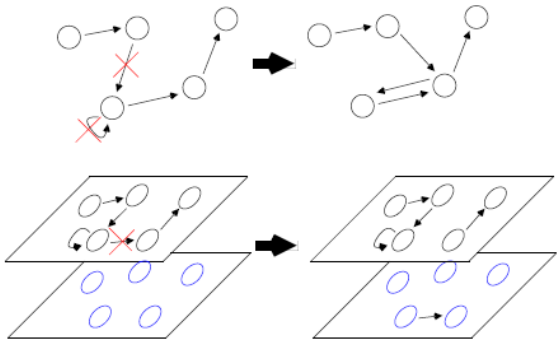


Figure 12 Path Searching[Arikan and Forsyth, 2002]

The path is generated by connecting the nodes in the graph.

An used GA to generate the motions of a robotic arm, which are similar to the demonstrations [An et al., 2007].

Lee proposed to use Continuous Hidden Markov Model to generate the required actions which is constrained by the observation of the demonstrations [Lee and Nakamura, 2006].

Atkeson proposed that robots should learn the demonstrations through practice [Atkeson and McIntyre, 1986]. Figure 13 displays the control architecture of Atkeson's method. X_d is the required position or joint angle which the joint of a robot should move to, X is the actual measured position of joint angle, τ is the torque on the joint, and τ_{ff} is the feedforward torque.

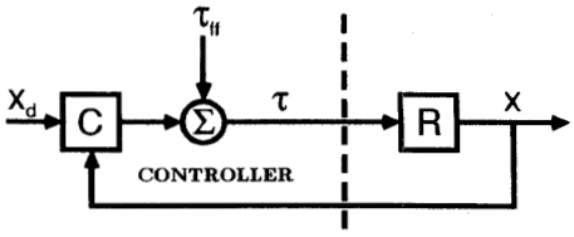


Figure 13 Control Architecture in Atkeson's Method [Atkeson and McIntyre, 1986]

Demiris proposed to use a predictive architecture to generate required behaviors [Demiris and Hayes, 1996]. Figure 14 displays his architecture. The behavior module receives current state information and the target goal(s), and a motor command is generated to achieve the goal state. The forward model provides an estimate of the next state to adapt the PID controller.

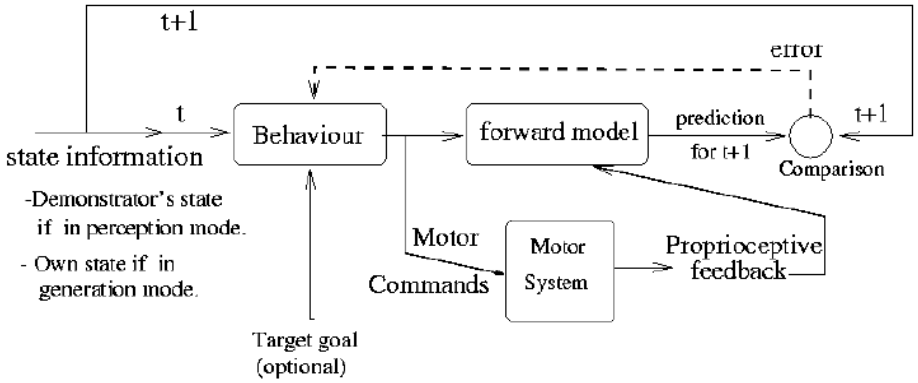


Figure 14 Demiris's Imitation Learning Architecture [Demiris and Hayes, 1996]

Khatib used the potential field used to generate motions for robots to avoid obstacles [Khatib, 1985]. Brock represents the demonstrations using "tunnels". The generation of a path in a tunnel is accomplished by imposing a local-minima free potential function in the tunnel. Miyamoto used Via-Points in the representation of the demonstrations[Miyamoto and Kawato, 1998]. At the generation stage, given a task, a trajectory is generated to pass all Via-Points [Miyamoto and Kawato, 1998]:

Bitzer proposed that the learned demonstrations are stored in the Latent space, and at the generation stage, robots simply map the points in the Latent space to the original data space as shown in Figure 15 [Bitzer and Vijayakumar, 2009].

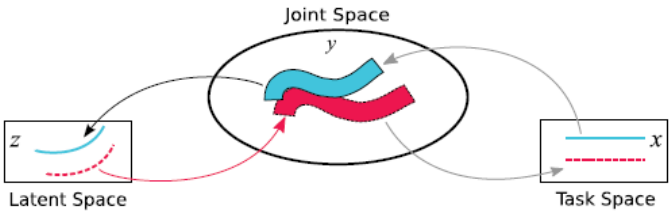


Figure 15 Relationship between Different Spaces [Bitzer and Vijayakumar, 2009]

As shown in Figure 15, a robot needs to complete a task in the task space, which is normally described in the Cartesian space. In the task space, the position and the orientation of the end-effector of the robot and the task-relevant objects are described. Using inverse kinematics (assuming that they are available), the corresponding joint angles are then computed to generate the required positions and orientations of the end-effector. The data points in the joint space are often projected to a low-dimensional space, i.e., latent space. At the generation stage, robots only need to project the data points from the latent space to the joint space to generate required movement trajectories. Shon applied the latent space approach for robotic imitation learning [Shon et al., 2005].

Adapting Methods

--*"Lagrangian Method"*

Calinon and Billard proposed a method to minimize the distance between the generated trajectory and the demonstrated trajectory with some constraints [Calinon et al., 2007].

Let $X_s = \{\theta_s, x_s, y_s\}$ where θ_s is joint angles of the two arms and the torso, x_s is the Cartesian positions of the two hands, and y_s is the hands-object relationships. Let o_s be the positions of the objects.

$$y_s = x_s - o_s \quad (2)$$

The dimension of the sampled data is reduced using PCA:

$$X_s - \bar{X}_s = A\xi_s \tag{3}$$

The elements in $\xi_s = \left\{\hat{\xi}_s{}^{\theta}, \hat{\xi}_s{}^{x}, \hat{\xi}_s{}^{y}\right\}$ are, respectively, the generalized joint angle trajectories, the generalized hand path and the generalized hands-object distance vectors extracted from the demonstrations in the latent space. The generated trajectory is represented as$\left\{\xi_s{}^{\theta}, \xi_s{}^{x}, \xi_s{}^{y}\right\}$.

The metric of imitation performance is H and is given by:

$$\begin{aligned} H = &\left(\xi_s{}^{\theta} - \hat{\xi}_s{}^{\theta}\right)^T W^{\theta}\left(\xi_s{}^{\theta} - \hat{\xi}_s{}^{\theta}\right) + \left(\xi_s{}^{x} - \hat{\xi}_s{}^{x}\right)^T W^{x}\left(\xi_s{}^{x} - \hat{\xi}_s{}^{x}\right) \\ &+ \left(\xi_s{}^{y} - \hat{\xi}_s{}^{y}\right)^T W^{y}\left(\xi_s{}^{y} - \hat{\xi}_s{}^{y}\right) \end{aligned} \tag{4}$$

Where W^{θ}, W^{x}, W^{y} are symmetric weighted matrices, and the target is to find a minimum H.

$$c_{1,i,j} = \hat{\xi}_{s,i,j}{}^{\theta} - \xi_{s,i,j-1}{}^{\theta} \tag{5}$$

$$c_{2,i,j} = \hat{\xi}_{s,i,j}{}^{x} - \xi_{s,i,j-1}{}^{x} \tag{6}$$

$$c_{3,i,j} = \hat{\xi}_{s,i,j}{}^{y} - \xi_{s,i,j-1}{}^{y} \tag{7}$$

So equation (4) is rewritten as

$$\begin{aligned} H = &\left(\dot{\xi}_s{}^{\theta} - c_1\right)^T W^{\theta}\left(\dot{\xi}_s{}^{\theta} - c_1\right) + \left(\dot{\xi}_s{}^{x} - c_2\right)^T W^{x}\left(\dot{\xi}_s{}^{x} - c_2\right) \\ &+ \left(\dot{\xi}_s{}^{y} - c_3\right)^T W^{y}\left(\dot{\xi}_s{}^{y} - c_3\right) \end{aligned} \tag{8}$$

The Lagrangian is defined as:

$$L = H + \lambda_1{}^T\left(\dot{\xi}_s{}^{x} - J\dot{\xi}_s{}^{\theta}\right) + \lambda_2{}^T\left(\dot{\xi}_s{}^{y} - A^z\dot{\xi}_s{}^{x} + (A^y)^{-1}\dot{o}_s\right) \tag{9}$$

where J is the Jacobian matrix of the dynamics model of a robot.

In order to minimize the Lagrangian, L is differentiated to obtain the gradient ∇L and set the gradient equal to zero,

$$-2W^{\theta}\left(\dot{\xi}_s^{\ \theta}-c_1\right)-J^T\lambda_1=0 \quad (10)$$

$$-2W^{x}\left(\dot{\xi}_s^{\ x}-c_2\right)+\lambda_1-(A^z)^T\lambda_2=0 \quad (11)$$

$$-2W^{y}\left(\dot{\xi}_s^{\ y}-c_3\right)+\lambda_2=0 \quad (12)$$

Then,

$$\lambda_1=2W^{x}\left(\dot{\xi}_s^{\ x}-c_2\right)+2(A^z)^TW^{y}\left(\dot{\xi}_s^{\ y}-c_3\right) \quad (13)$$

$$\begin{aligned}\dot{\xi}_s^{\ \theta}=&\left(W^{\theta}+J^TW^{x}J+(A^zJ)^TW^{y}A^zJ\right)^{-1}\\&\times\left(W^{\theta}c_1+J^TW^{x}c_2+(A^zJ)^TW^{y}((A^{y})^{-1}\dot{o}_s)+c_3\right)\end{aligned} \quad (14)$$

Iteratively, the joint angle in the latent space can be calculated as:

$$\xi_{s,i,j}^{\ \theta}=\xi_{s,i,j-1}^{\ \theta}+\dot{\xi}_{s,i,j}^{\ \theta} \quad (15)$$

The joint angles in the original data space are:

$$\theta_s=A^{\theta}\xi_s^{\ \theta}+\bar{\theta}_s \quad (16)$$

An example of generating behavior of grasping-moving an object using this method is shown in Figure 16.

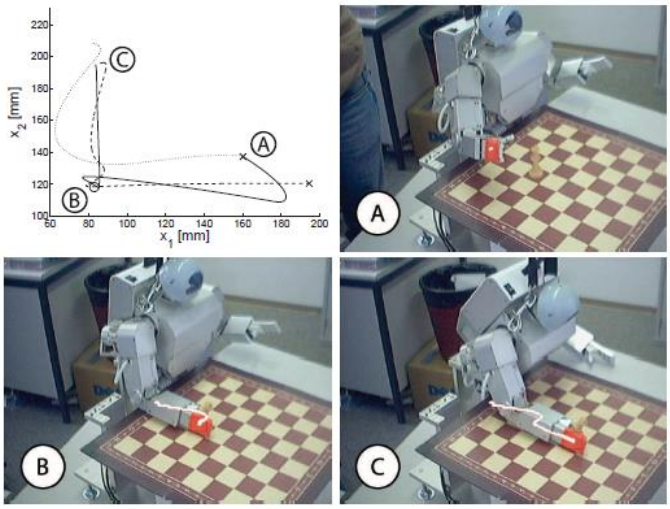


Figure 16 Generation Results in Calinon's Method [Calinon et al., 2007]

The solid line is the generalized demonstration of the position of the end-effector, the dot line is the reconstructed positions of the end-effector from the sampled angles, and the dashed line is the generated trajectory using the above algorithm.

Figure 17 displays the generation results when the objects are placed at different locations.

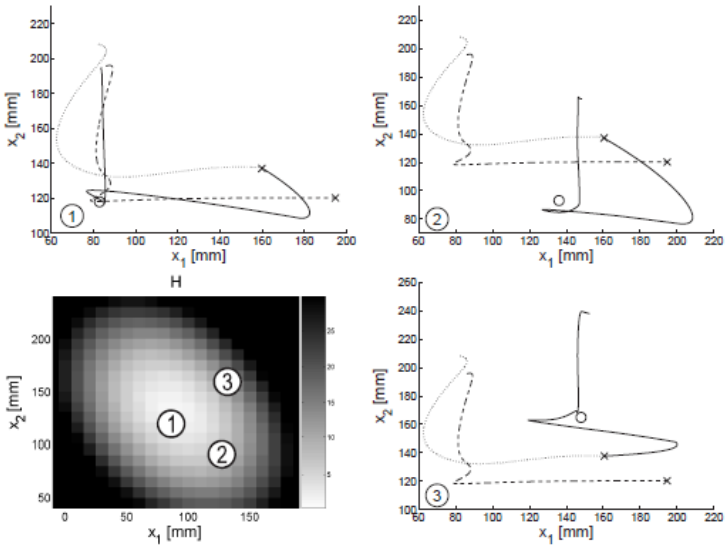


Figure 17 Generation Results in Calinon's Method in Similar Situations[Calinon et al., 2007]

--Dynamic Movement Primitive Method

Schaal proposed an imitation learning architecture using Movement Primitives as shown in Figure 18.

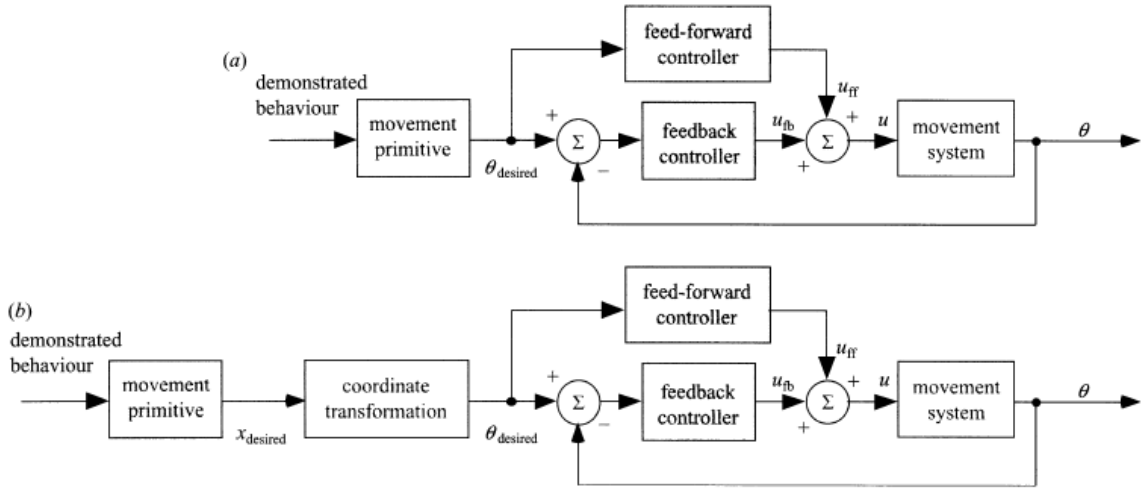


Figure 18 Schaal's Imitation Learning Architecture [Schaal et al., 2003]

The demonstrated behavior is mapped onto a movement primitive that is defined in internal coordinates of the robot: joint angular coordinates θ are a good candidate as they can be extracted from visual information, a problem addressed as pose estimation in computer vision. Such internal coordinates can directly serve as the desired input to a motor-command execution stage here and assumed to be composed of a feedback and a feed-forward control block.

Ijspeert proposed to use Dynamic Movement Primitives (DMP) [Ijspeert et al., 2002] [Ijspeert et al., 2003]. The DMP algorithm describes the generated trajectory as a combination of a second-order attractor and a nonlinear function which describes a generalized demonstration ("generalized" means that this function could be constructed from several demonstrations for one task.) and modulates the trajectory in the generation process.

The formulation of the DMP algorithm is shown as differential equations:

$$\tau \dot{z} = \alpha_z(\beta_z(g - y) - z) \quad (17)$$

$$\tau \dot{y} = z + f \quad (18)$$

where g is the goal state, z is the internal state, f is calculated to record the dynamic of the demonstration and to guarantee convergence of the new generated trajectories, y is the position generated by the DMP differential equations, and $\dot{y}$ is the generated velocity correspondingly. α_z, β_z, and τ are the constants in this equation. f is a Receptive Field Weighted Regression (RFWR) model [Atkeson et al., 1997].

This method has been extended to rhythmic DMP in 2003. Similar to the discrete DMP, rhythmic DMP is also represented as differential equations.

$$\tau \dot{z} = \alpha_z(\beta_z(y_m - y) - z) \quad (19)$$

$$\tau \dot{y} = z + f \quad (20)$$

where,

$$f = \frac{\sum_{i=1}^{N} \Psi_i w_i v}{\sum_{i=1}^{N} \Psi_i} \quad (21)$$

$$\Psi_i = exp(-h_i(mod(\varphi, 2\pi) - c_i)^2) \quad (22)$$

where y_m is an anchor point for the oscillatory trajectory.

Figure 19 displays the generation results using discrete DMP in a Reaching experiment.

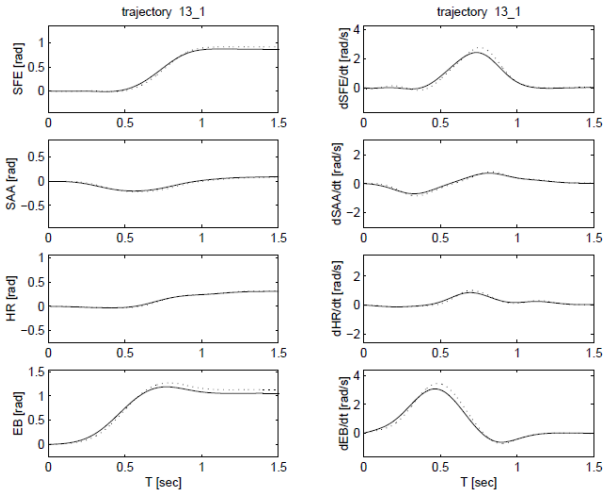


Figure 19 Generation Results using Discrete DMP [Ijspeert et al., 2003]

In Figure 19, the dotted lines are the demonstrated trajectories, and the solid lines are the generated trajectories using the DMP method. From the graph, it is obvious that the generated trajectories fit the demonstrations well.

Figure 20 displays the generation results using rhythmic DMP in a Drum-Hitting experiment.

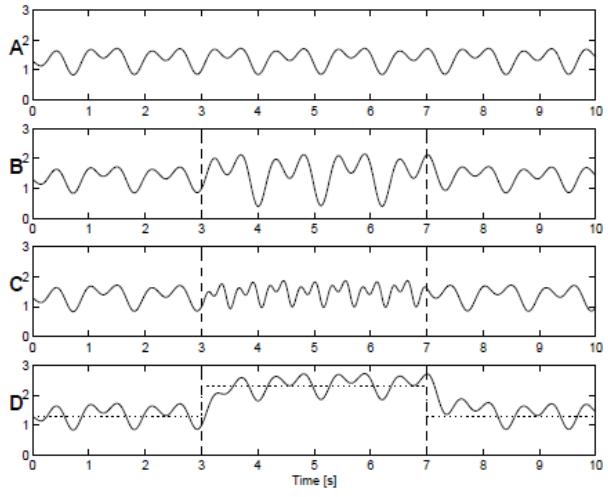


Figure 20 Generation Results using Rhythmic DMP [Ijspeert et al., 2003]

In Figure 20, picture A is the demonstrated rhythmic trajectories; picture B, C and D are the generated rhythmic trajectories. When increasing the constant τ, the generated

result of picture B and C from time 3s to 7s displays that the robot successfully learned the tempos of the demonstrated trajectory and can increases the tempo by changing the time constant τ. Picture D displays that the robot can learn to generate the trajectories at different locations and keep the tempos.

--*Reinforcement Learning Based Methods*

Reinforcement Learning is based on the Markov Property Assumption [Sutton and Barto, 1998] that the probability of the current transition only depends on the previous state. Based on the basic assumption of the Markov property:

$$P(s_{t+1} = s', r_{t+1} = r | s_t, a_t) \quad (23)$$

Equation (23) means that the state at the next time step is determined by the current state and the current action.

Some important definitions in Reinforcement Learning are explained as follows [Sutton and Barto, 1998].

Given the current state s with action a, the probability of transition from s to s' is:

$$P_{ss'}^{a} = P(s_{t+1} = s' | s_t = s, a_t = a) \quad (24)$$

And the expected reward from this transition is:

$$R_{ss'}^{a} = P(r_{t+1} | s_t = s, a_t = a, s_{t+1} = s') \quad (25)$$

Given the current state s at time step t and a policy π, the expected reward is:

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s\right\} \quad (26)$$

where γ is a discount factor. Given the current state s at time step t, current action a at time step t, and a policy π, the expected reward is:

$$Q^{\pi}(s,a) = E_{\pi}\{R_t | s_t = s, a_t = a\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s, a_t = a\right\}$$
$$= \pi(s,a) \sum_{s'} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma V^{\pi}(s')\right] \quad (27)$$

Rewrite equation (26),

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s\right\}$$
$$= \sum_{a} \pi(s,a) \sum_{s'} P_{ss'}^{a} \left[R_{ss'}^{a} + \gamma V^{\pi}(s')\right] \quad (28)$$

Theodorou proposed applying reinforcement learning methods in optimal control to teach robots to learn and generate motion trajectories [Theodorou et al., 2010].

CHAPTER III

METHODOLOGY

Motivation

Imitation learning provides a possible solution for task-specific behavior generation. However, researchers gradually found that although that it is possible to design methods for robots to learn how to complete a specific task in a specific situation, it is very difficult to design a general imitation learning method to generate situation-specific behaviors in a large number of different situations. A possible solution is to teach robots basic behaviors and let robots to complete new and complex behaviors through some cognitive processes. Then, intelligent robots should be able to apply these learned skills in different situations. For example, given a task, robots should know whether they can complete the task using the learned behaviors or by adapting these behaviors to similar but slightly different situations. Behavior selection and generation should be based on the current situation, sensed environmental information, the capabilities of the robot, etc. This process could be carried through internal rehearsal or by trials in the real environment.

Researchers look for inspirations from the cognitive science, because cognitive science investigates learning processes in human or animal brains and possibly it can provide solutions to current imitation-based robotics research, especially for the research on humanoid robots. In a dynamic environment, a robot should analyze the environment

and use its owned skills to make a decision whether it is possible to complete the required task by flexibly and adaptively switching processes, strategies etc.

One way for a robot to perform tasks using learned behaviors in a complex, dynamic, and unstructured environments may be to integrate cognitive control with robotic imitation learning. The motivation of this dissertation is to investigate how imitation learning can be used in robotic cognitive control. Imitation learning provides a method of learning behaviors from demonstrations and generating behaviors in similar task-relevant situations without being preprogrammed. Cognitive control provides a framework of switching strategies, processes, etc., and adaptively completing tasks [Banich et al., 2009] using the learned behaviors. In this dissertation, imitation learning methods are generalized to behavior generation, the storage of generalized behaviors and the description of the relationship among these learned behaviors, and learned behaviors will be used to generate similar behaviors or behavior sequences in different task-relevant situations. By integrating a cognitive control framework, the robot is able to switch strategies based on current environment situations.

In this dissertation, we limit our proposed skill learning to the tasks of "object handling". Examples of "object handling" include: Reaching, Pushing, Grasping, Playing, Yo-Yo Playing, Assembling, Tower of Hanoi, etc.

Imitation Learning Framework

In this dissertation a robotic imitation learning framework is divided into the following main parts: behavior acquisition, behavior segmentation, behavior generalization, behavior representation, and behavior generation.

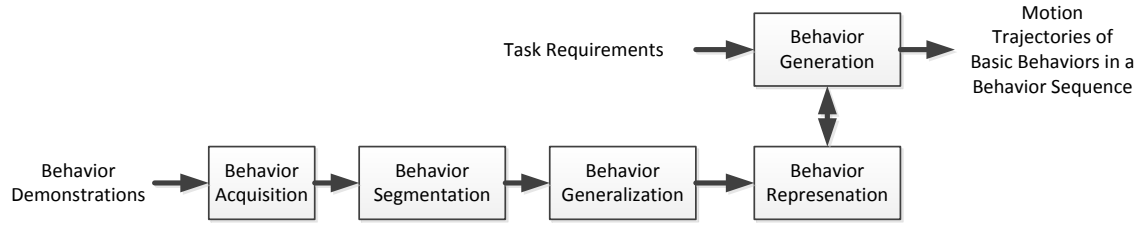


Figure 21 Imitation Learning Framework

Figure 22 displays the designed overall system framework for imitation learning.

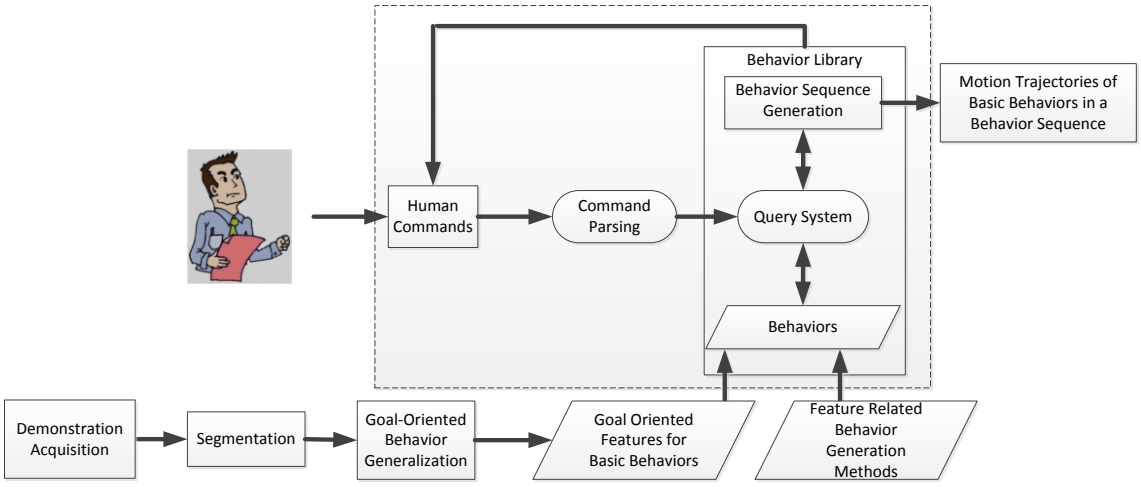


Figure 22 Detailed System Framework for Imitation Learning

The “Demonstration Acquisition” block records the motion trajectories of a hand of human teachers.

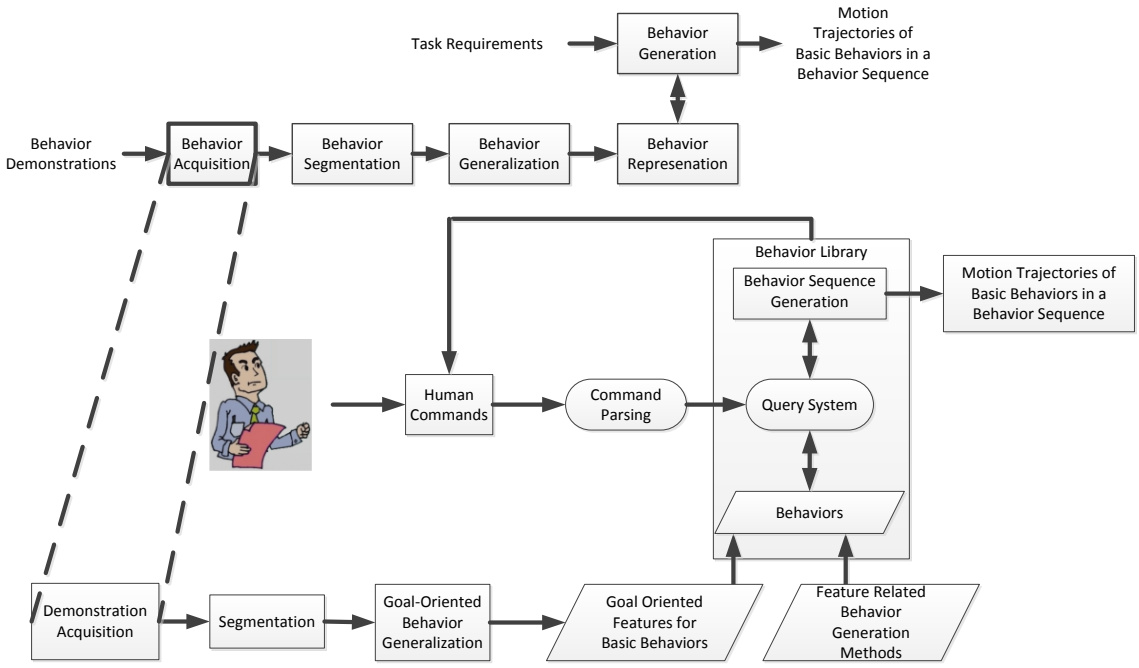


Figure 23 Behavior Acquisition

The “Segmentation” block segments the observed behavior sequences into several basic behaviors.

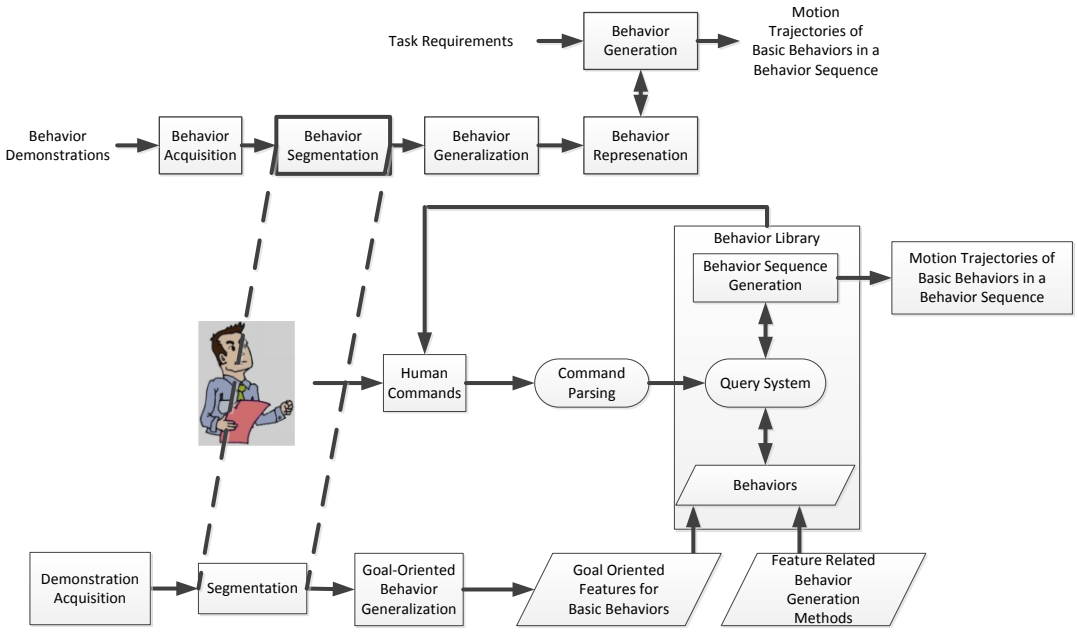


Figure 24 Behavior Segmentation

The “Goal-Oriented Behavior Generalization” block generalizes the common features of demonstrated motions.

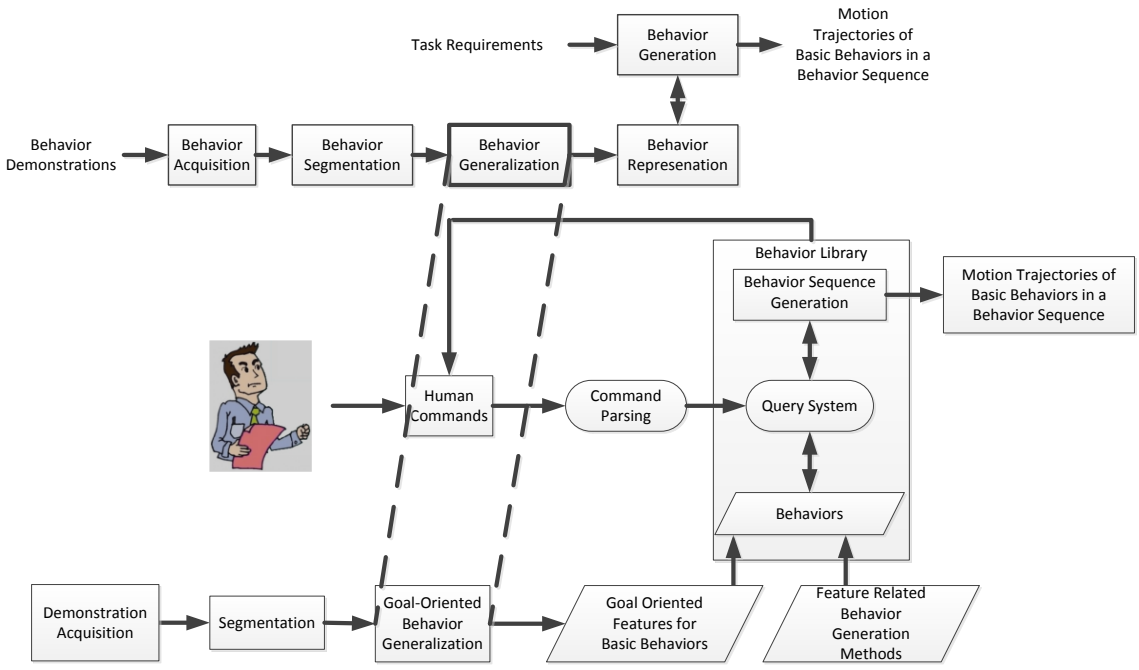


Figure 25 Behavior Generalization

Generalized features (represented as a group of attributes) are standardized as described in the section of “Behavior Generalization” and stored in the “Behavior Library” in the memory system. The “Behavior Library” includes both semantic (behaviors) and numeric (trajectories and corresponding dynamic parameters).

The “Feature Related Behavior Generation Toolbox” contains predefined behavior generation methods and also stored in the memory system.

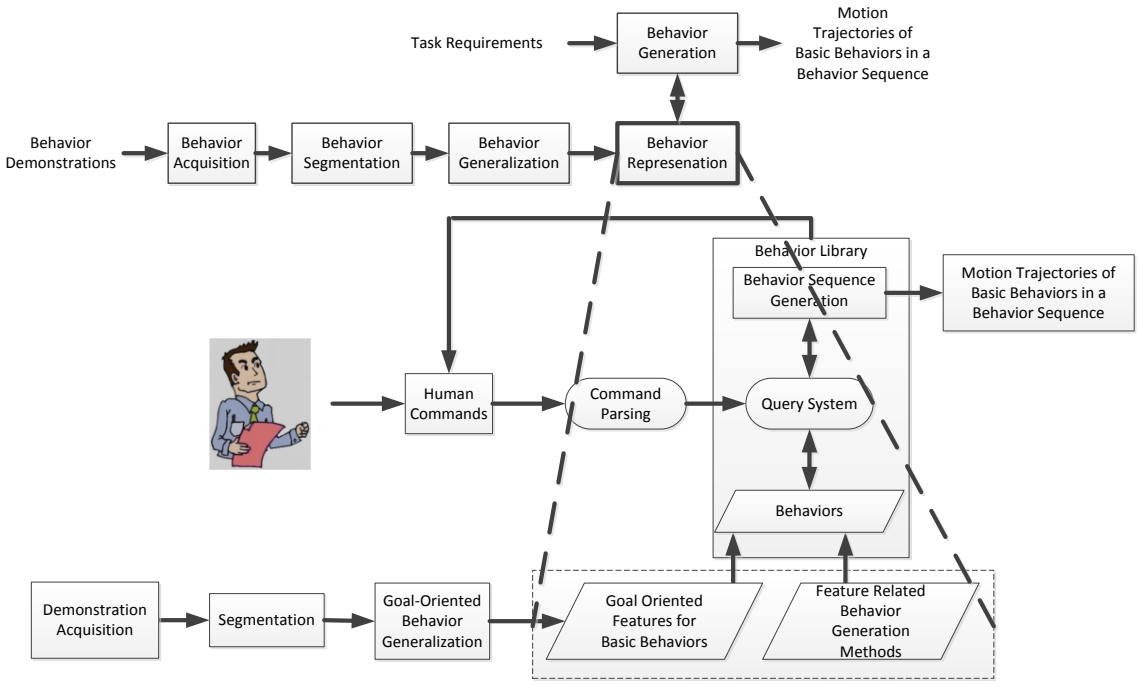


Figure 26 Behavior Representation

In the generation stage, given a new human command, robot searches the "Behavior Library" using a "Query System". If robot finds a matching behavior in the "Behavior Library", robot retrieves it and change parameters to fit into the new command. Using Dijkstra's algorithm [Dijkstra, 1959], a behavior sequence will be constructed by finding a shortest path from the "starting" behavior to the required behavior. We add "starting" and "ending" behavior in our behavior database. This could enable robots to generate a behavior sequence by starting from "starting" behavior to the "ending" behavior and simplifies the behavior sequence generation. If there is no match, it means that the behavior sequence cannot be generated using the shortest path searching algorithm, and the robot has to ask a human teacher to demonstrate the unlearned and required behavior, generalizes the demonstrated new behavior, and adds it to the Behavior Library by itself. After learning the new behavior and adding it into the

behavior library, a behavior sequence can be generated using Dijkstra's algorithm. When the behavior sequence is generated, the robot uses pre-defined behavior motion trajectories generation methods in the memory system, such as second-order attractor, DMP, Potential Field, RRT, etc., as described in Table 5 on page 91, to generate motion trajectories for all the behaviors in the generated behavior sequence. The motion trajectories are then assembled for robot to complete the required task.

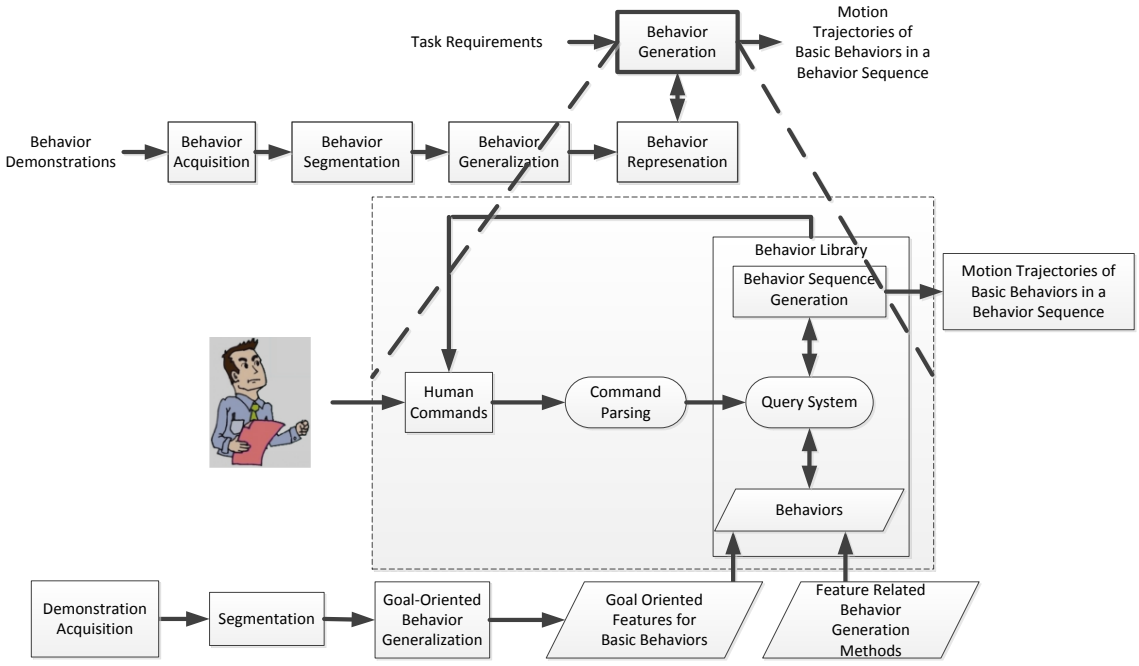


Figure 27 Behavior Generation

Cognitive Architecture

Recently, cognitive architectures are receiving broad attention from the robotics community, because they provide a process for high-level cognitive activities, such as cognitive control [Badre, 2008]. Figure 28 is the system diagram of the ISAC Cognitive Architecture developed in our lab, which is a multi-agents hybrid architecture. This cognitive architecture provides three control loops for cognitive control: *Reactive*,

Routine and *Deliberative*. Behaviors can be generated through this cognitive architecture. Imitation learning basically should be involved in the Deliberative control loop. Three memory components implemented in this architecture are: Working Memory System (WMS), Short Term Sensory Memory (STM), and Long Term Memory (LTM) [Kawamura et al., 2008].

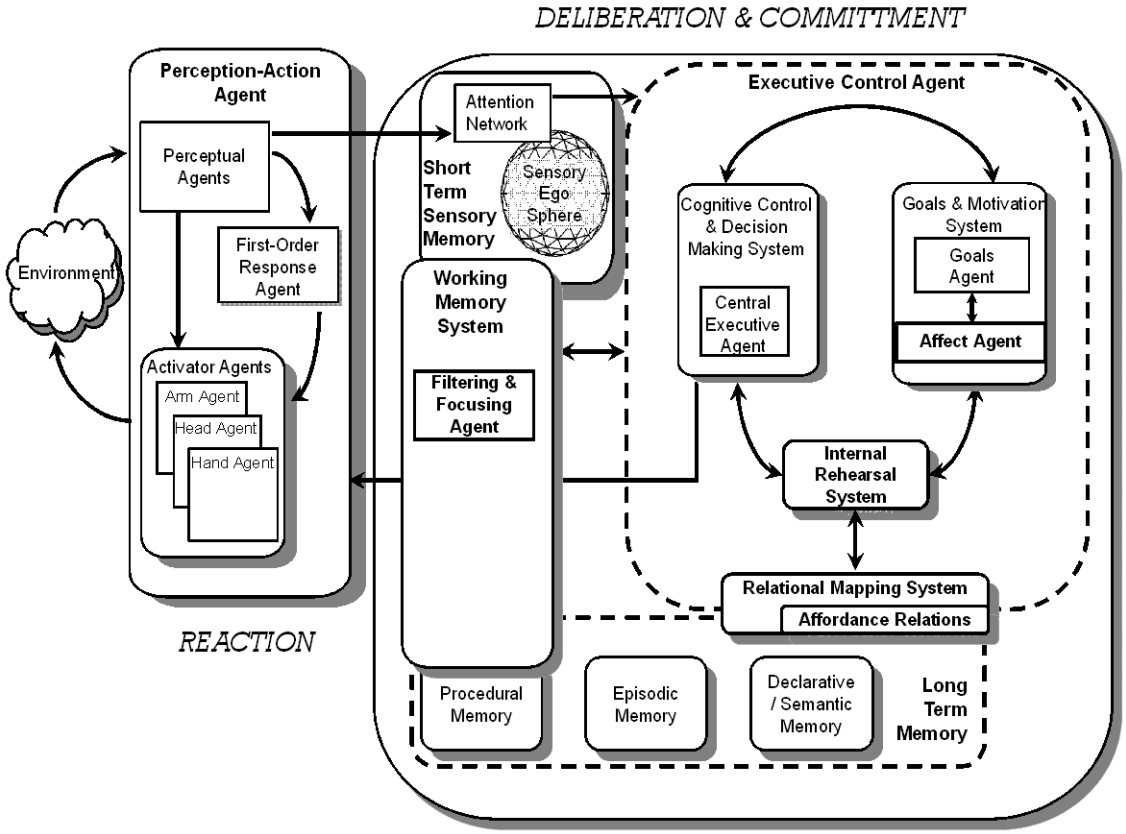


Figure 28 ISAC Cognitive Architecture [Kawamura et al., 2008]

In our lab, Joe Hall used this cognitive architecture to implement cognitive control experiments [Hall III, 2007]. In his method, ISAC used the IRS to evaluate whether it can reach an object in the environment. The reaching behavior is defined by interpolating the points between the starting point of the arm and the position of the object. Using the IRS, ISAC finds a collision between the obstacle in the environment

and its arm, and determines whether the object is reached. If ISAC determines that it can reach the object, it executes a behavior sequence; if not, it returns failure information. One interesting aspect of Hall's work is to evaluate a set of successful sequence and pick one of them.

Based on our ISAC cognitive architecture, I proposed a simplified hybrid cognitive architecture [Tan and Liang, 2011] as shown in Figure 29. In order to integrate imitation learning framework, the agents which are related to behavior generation are kept and some agents are deleted. For example, *Relational Mapping* is not used in our system, so it has been removed.

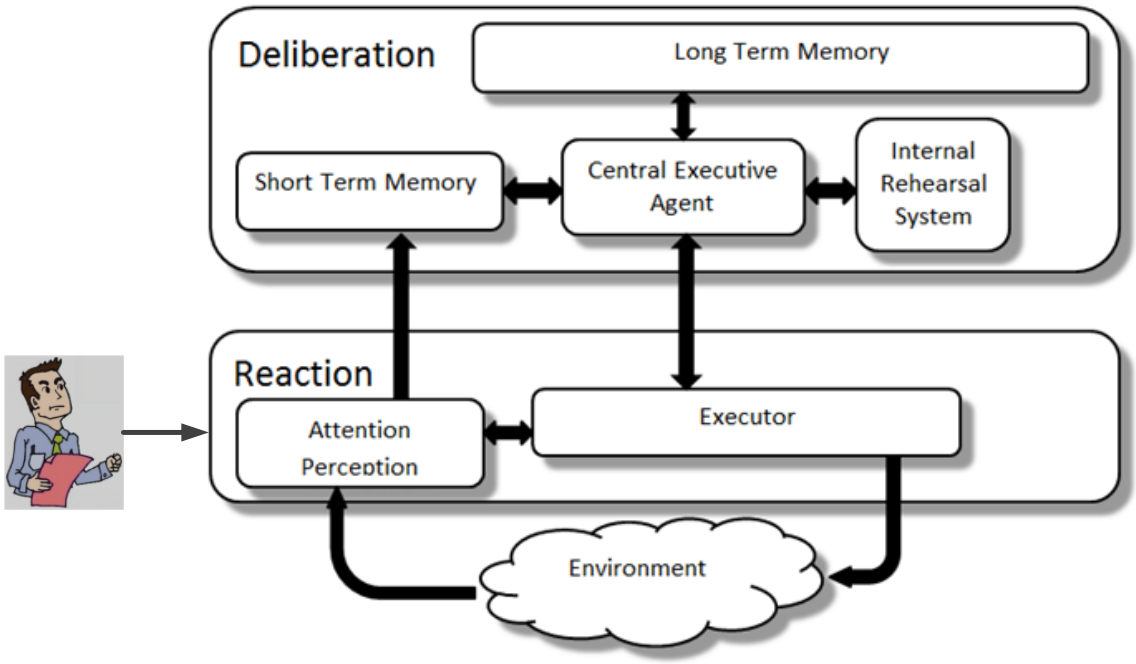


Figure 29 A Simplified Hybrid Cognitive Architecture

--*Reactive Layer*

In this layer, the robot senses the environment and the status of the robot body and moves the end-effector along the generated motion trajectories.

Executor

Executor receives the motion trajectories from the CEA, which is described by points on the trajectories and moves the end-effector of the robot by passing these points.

Attention-Perception (AP)

The AP gathers information from the environment and the robotic body.

--Deliberation Layer

In this layer, the robot can complete high-level cognitive processes to learn demonstrated behaviors by generalizing common features of the demonstrations as described in the section of "Behavior Generalization", store the learned behaviors in the LTM, generate behavior sequences by finding a path in a constructed behavior graph as described in the section of "Behavior Graph Construction", and generate motion trajectories for all behaviors in behavior sequences as described in the section of "Motion Trajectory Generation" to complete tasks.

Short Time Memory (STM)

The STM stores the environmental information including the position and the sizes of the target object and the obstacle in the environment, the joint angles of the robot arms (designed as shown in the Demonstration Acquisition section).

Long Term Memory (LTM)

The LTM stores the learned behaviors and the semantic description of the objects or obstacles in object handling tasks.

Internal Rehearsal System (IRS)

The IRS [Hall III, 2007] [Erdemir et al., 2008] evaluates the current behavior sequence and sends the evaluation results to the CEA for decision making. The Kinematics model and dynamics model are stored in the IRS for the robot to use.

Central Executive Agent (CEA)

The CEA is responsible for cognitive control and decision making process.

In the learning stage, the CEA receives the information of sensed states from the STM. Basic behaviors are generalized to find its goal-oriented common features (as the output of the behavior generalization) and stored into the LTM.

In the generation stage, the CEA receives the task command from the STM and generates behavior sequences and motion trajectories for all the behaviors in the behavior sequence. The decision-making mechanism, which is rule-based, switches strategies to complete the given task. The explanation of the key components in the CEA will be discussed on page 66 in this Chapter.

Integration

We propose to integrate our imitation learning framework with cognitive architecture. The basic idea is shown in Figure 30. Behaviors can be learned, generated and generated through our cognitive architecture.

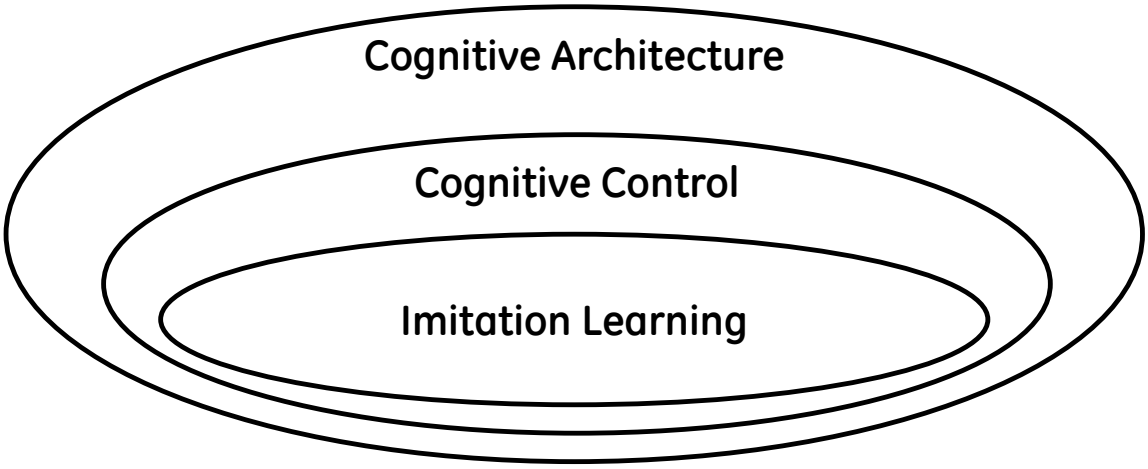


Figure 30 Integration of Imitation Learning with Cognitive Control

Figure 31 displays a cognitive control system block diagram developed in our lab [Kawamura and Gordon, 2006].

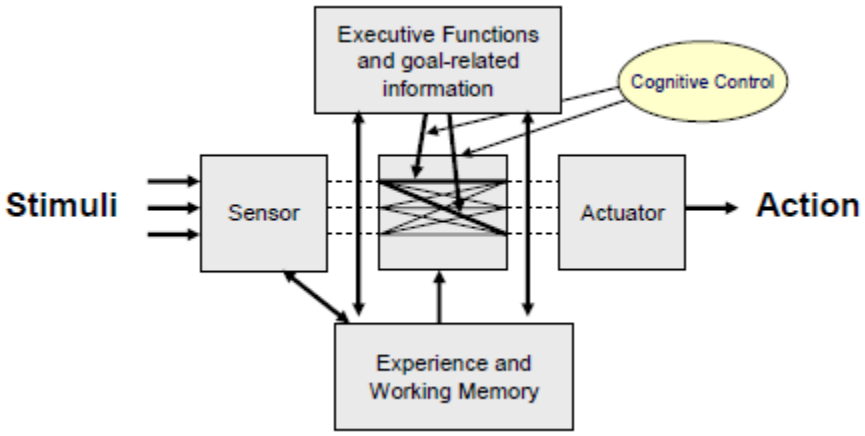


Figure 31 Cognitive Control Block Diagram

Based on this model and the cognitive architecture shown in Figure 9, a modified cognitive control system block diagram is shown in Figure 32.

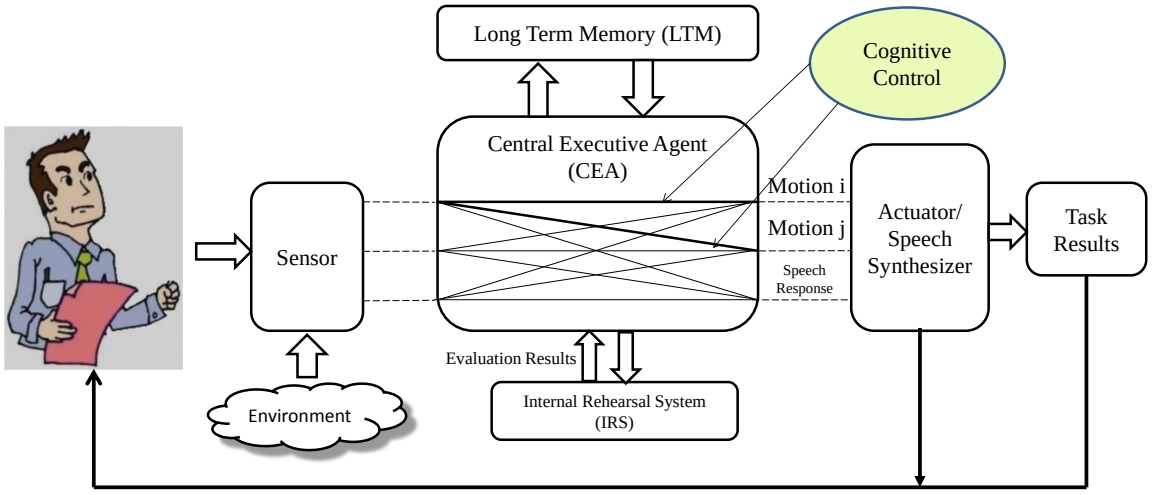


Figure 32 Modified Cognitive Control System Diagram

The integration of the imitation learning framework and the cognitive control block diagram is shown in in the following figures.

The *Behavior Acquisition* of the imitation learning framework is integrated with the *Sensor*.

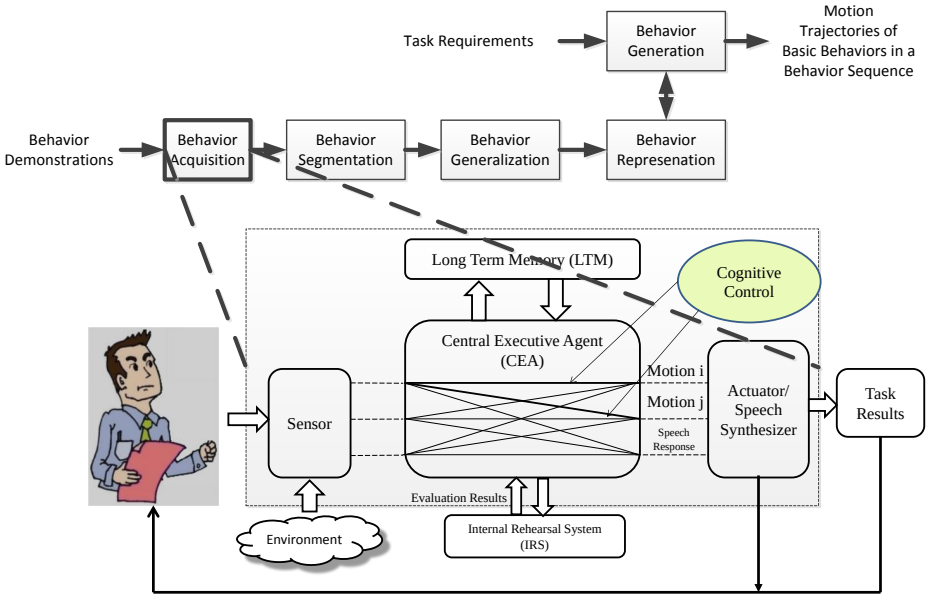


Figure 33 Integration of Behavior Acquisition

The *Behavior Segmentation* and the *Behavior Generalization* is integrated with the *CEA*.

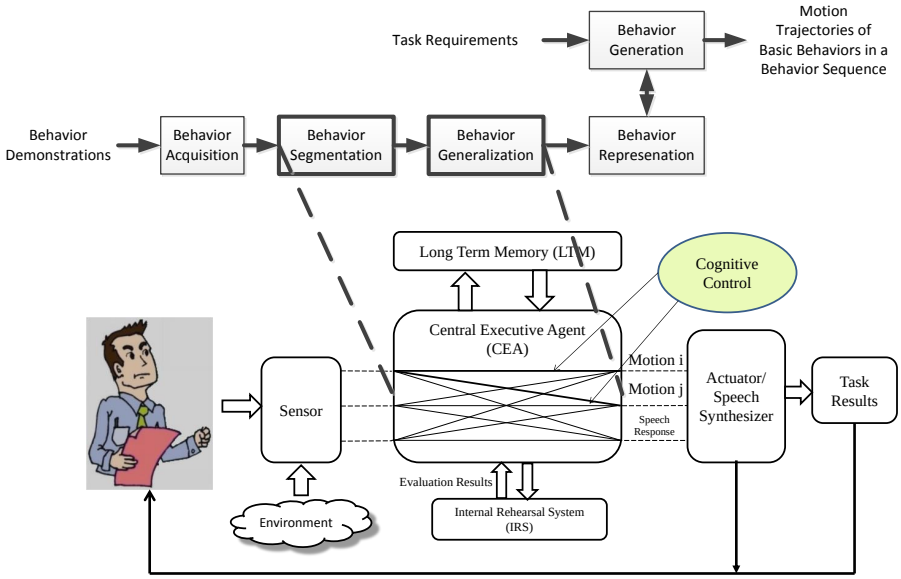


Figure 34 Integration of Behavior Segmentation and Generalization

The *Behavior Representation* is integrated with the *LTM*.

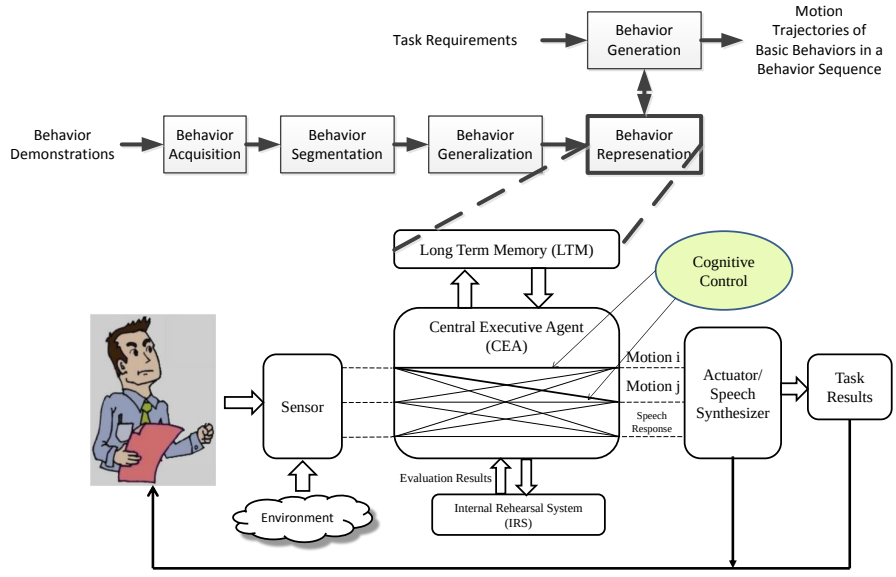


Figure 35 Integration of Behavior Representation

The *Behavior Generation* is integrated with the *Perception/Attention*, the *STM*, the *CEA*, the *IRS* and the *Executor*.

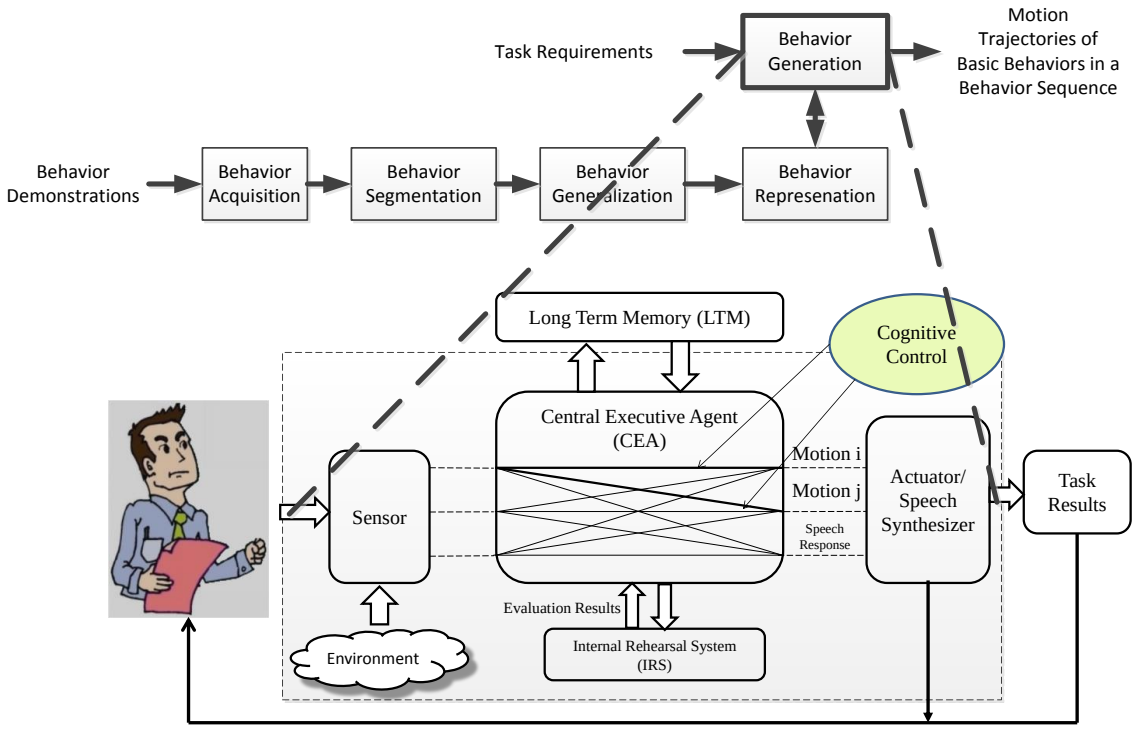


Figure 36 Integration of Behavior Generation

An integrated system diagram is shown in Figure 37. The italicized individual components are described in detail subsequently. Refer to Figure 37 in the following description.

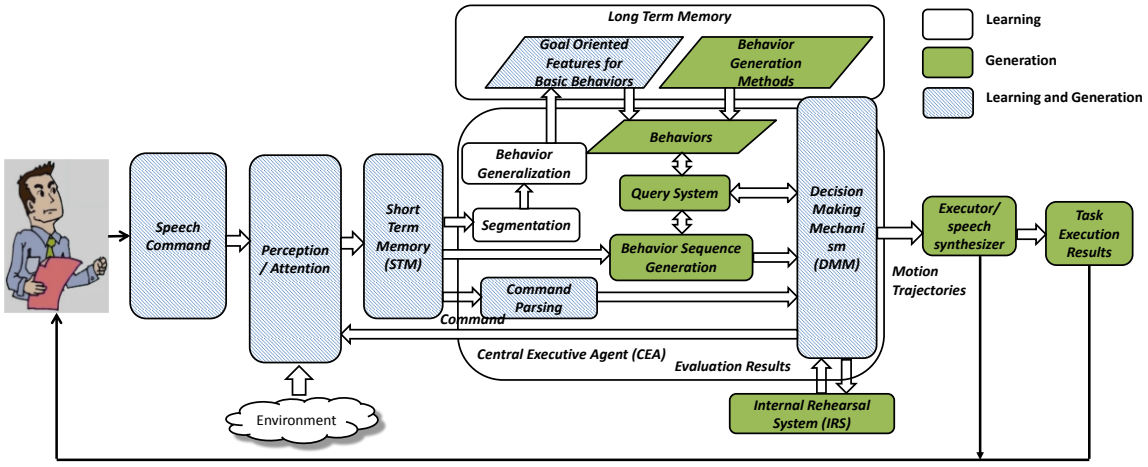


Figure 37 Integrated System

The *Perception/Attention* block collects sensory information from the environment and stores it in the *Short Term Memory* (STM). Speech commands are obtained by the *Perception/Attention* module and parsed by the *Command Parsing* module, which is described in the section of Command Parsing on Page 33-35. Based on the parsing results, the *Decision Making Mechanism* (DMM) decides to switch the control process to either learning or generation.

In the learning stage, a human teacher demonstrates a behavior or a behavior sequence and the robot uses the *Perception/Attention* block to record the motion trajectories. After segmentation the motion trajectories are sent to the *Behavior Generalization* block to extract common features which are stored in the *Goal-Oriented Features for Basic Behaviors* block in the *Long Term Memory* (LTM).

In the generation stage a new command causes the robot to search the *Behavior Library* using a query system. No match implies that the behavior sequence cannot be generated by searching from the "Starting" to the required behavior. The robot must ask a

teacher to demonstrate the required behavior (not the overall behavior sequence). The human teacher checks the behavior library of the robot and decides which behavior needs to be demonstrated. The learning stage is invoked by the DMM and the *Behavior Library* is updated. If the robot finds a matching behavior in the *Behavior Library*, it is retrieved and the parameters changed to fit the new command. The *Behavior Sequence Generator* uses the *Query* component to search the *Behavior Library* then constructs, via Dijkstra's algorithm, a sequence that follows shortest path from the current behavior to the required behavior. After a behavior sequence is identified, the system selects the appropriate motion trajectory generator from the memory system. (These include second-order attractors, DMPs, Potential Fields, RRTs, etc.) Thus the robot motion trajectories for the behaviors that comprise the task are generated. They are sent to the *Internal Rehearsal System* (IRS) for evaluation. The DMM uses the evaluation result to determine if the task can be completed. If so, the motion trajectories are assembled and sent to the *Executor* whereby the robot performs the task. If the DMM finds that the robot cannot complete the task with the selected arm, it transfers the behavior sequence to the other arm and causes the motion trajectories to be recomputed. Experiment 3.1 and 3.2 are used to validate this part. If the DMM finds that the robot cannot complete the task with either of its arms, it tries to generate behavior sequence for both arms. The DMM uses the IRS to evaluate the result of the generated behavior sequence. If it is successful, the newly generated behavior sequence and motion trajectories will be sent to the *Executor*. If this second evaluation finds the task still cannot be completed, the robot demurs the task.

The data flow of the learning stage is depicted with white arrows and arrows filled with slashes with in Figure 37. That of the generation stage is displayed with solid filled arrows and arrows filled with slashes.

Central Executive Agent

The CEA is responsible for cognitive control and decision making process.

Input/Output

In the Learning stage, the input of the CEA is the stored information in the STM and the output is the generalized basic behaviors to be stored in the LTM. In the generation stage, the input of the CEA is the speech command and the environmental information in the STM and the output is the generated motion trajectories sent to the executor or a speech response sent to the speech synthesizer.

Implementation

There are several key components inside the CEA: *Decision Making Mechanism, Segmentation, Behavior Generalization, Query System,* and *Behavior Sequence Generation.*

In the learning stage, the CEA receives the information of sensed states from the STM. Behavior Sequences are segmented into basic behaviors in the *Segmentation* block. Basic behaviors are generalized in *Behavior Generalization* to find its goal-oriented common features (as the output of the behavior generalization) and stored into the LTM.

In the generation stage, the CEA receives the speech command from the STM to trigger the *Decision Making Mechanism* which is rule-based. The *Decision Making*

Mechanism uses *Query System* to check whether robot has learned a complex behavior or a basic behavior to complete this task. If yes, it constructs a behavior graph and generates a behavior sequence finding a path in the behavior graph from the "Starting" to the required behavior; if no, it requires new demonstrations from human teachers. Motion trajectories are generated in *Behavior Sequence Generation* for all the behaviors in the behavior sequence. Then the environmental information and the generated behavior sequence are sent to the IRS for evaluation. If the robot finds that it can complete this task from the evaluation, it sends the generated motion trajectories to the Executor; if it finds that it is impossible to complete this task, it uses the IRS to evaluate the execution results by using the same behavior sequence for the other arm; if the robot finds that it cannot complete this task by using either of its arms; it tries to generate a behavior sequence by using both of its arms. If the robot finds that it cannot complete the task by using either or both of its arms, it displays a message on the screen.

The components in the CEA will be discussed in detail in following sections of this Chapter.

Decision Making Mechanism

The DMM controls the cognitive processes. According to current situations, the DMM make decisions and choose suitable behaviors to respond to requests from humans and to deal with uncertainties or emergencies in the environment.

Input/Output

In the learning stage, the input is parsed commands, and the output is a signal sent to the AP to start recording the demonstrated behaviors. In the generation stage, the input is parsed commands, generated behavior sequences, and evaluated results from the IRS. The output is the generated behavior sequence to be evaluated in the IRS and the generated motion trajectories which will be sent to the Actuators.

Implementation

The decision making process in the DMM is shown in Figure 38.

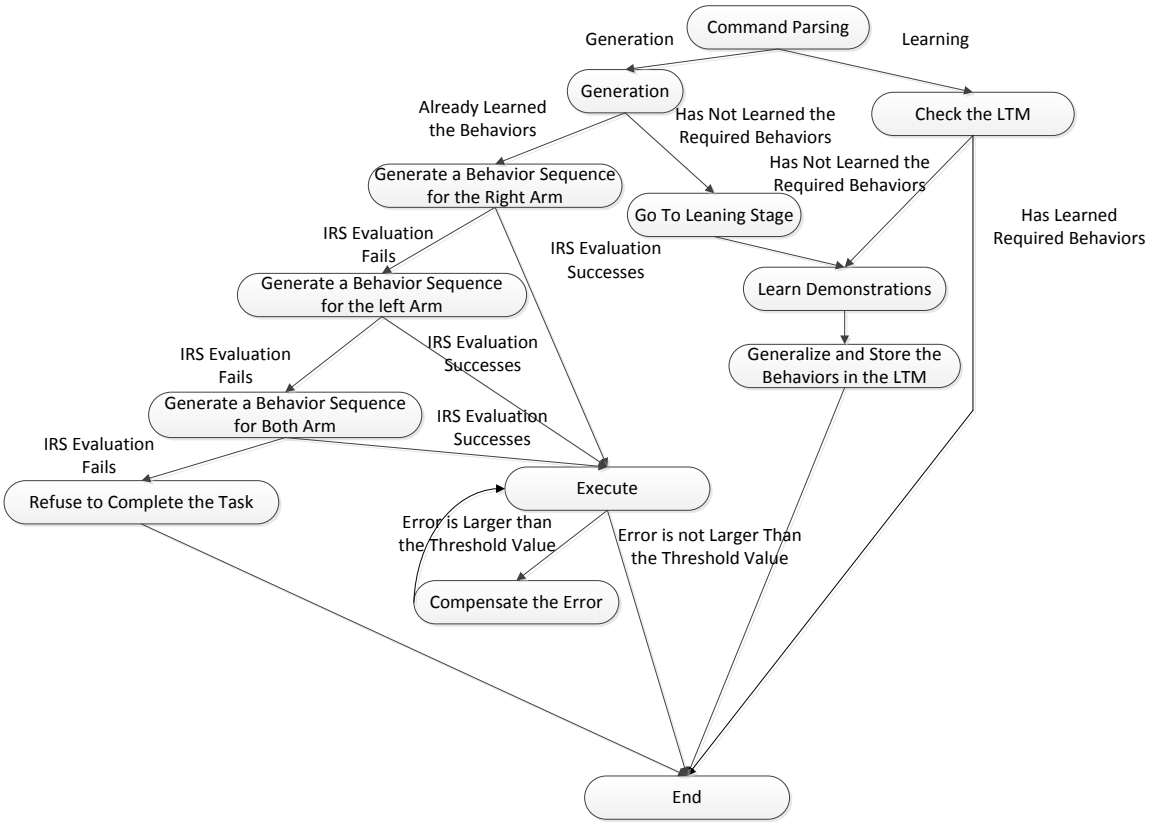


Figure 38 Decision Making Mechanism

Based on the output of the *Command Parsing* block, the DMM decides whether the robot should enter the learning stage or the generation stage.

In Figure 38, the decision making process is routed into two branches: learning and generation, by analyzing speeches and the current situation. In the learning stage, a robot requires demonstrations from human teachers. By observing the demonstrations, the robot analyzes and generalizes the common features of the demonstrated behaviors. Generalized behaviors are stored in the Behavior Library. In the generation stage, give a new command, the robot parses the command to behaviors and related parameters. The robot constructs a behavior graph using the stored information in the Behavior Library, and searches the shortest path from the "Starting" to the required behavior to generate a behavior sequence. If it finds it has not learned any of the required behaviors, the cognitive process turns to the learning stage. When the robot finds that it has all the required behaviors, it requires the environmental information such as the positions and sizes of the target object and the obstacle, and the joint angles of the robot, from the STM. Then the DMM uses the environmental information and sends the generated behavior sequence to the IRS for evaluation. IRS uses the behavior generation methods, which are described in the earlier section, to generate motion trajectories. If it finds that it is impossible to complete this task, it uses the IRS evaluate the execution results by apply the same behavior sequence using the other arm; if the robot finds that it still cannot complete this task by using its arms, it displays a message on the screen; if the robot finds that it can complete this task from the evaluation, it sends the evaluation information to the DMM. In the execution using one of its arms to complete the task, ISAC checks the error generated by each behavior. If the error is smaller than the threshold value, the error

is acceptable; if the error is larger than the threshold value, ISAC uses a compensation method to overcome the error and redo this behavior.

Compensator

The reasons of using compensator to overcome the error generated from the hardware of ISAC are:

1. The errors are generated from the hardware of ISAC. ISAC cannot exactly drive the end-effector to the required position.
2. The errors are generated from the vision module. The reaching point found by the vision module has some errors.
3. Comparing the size of the object used in Experiment, which is 18cm × 18cm × 12 cm, the error values are small. Thus the distance less than 12 cm was judged as that the gripper reached the object.

Input/Output

The input of the compensator is the obtained error. When motion trajectories are needed to be generated, ISAC checks the stored information in the compensator and changed the goal of the motion trajectory according to this information.

Implementation

The stored error is computed using the following equation:

$$error = P_{Desired} - P_{Actual} \quad (29)$$

where $P_{Desired} = \{X_{Desired}, Y_{Desired}, Z_{Desired}\}$ is the desired position of the end-effector, and $P_{Actual} = \{X_{Actual}, Y_{Actual}, Z_{Actual}\}$ is the actual position of the end-effector.

According to the design in the DMM, if the error is larger than the threshold value, the BSG will compensate it by changing the target position of the behaviors.

$$Goal_{new} = Goal_{original} + error \quad (30)$$

For example, if the desired position of the end-effector is $P_{Desired} = \{10,20,30\}$ and the actual position is $P_{Actual} = \{20,40,60\}$, the error is $\{-10, -20, -30\}$. Given a goal position of a motion trajectory $Goal_{original} = \{100,200,300\}$, the new goal position of the motion is $Goal_{new} = \{100,200,300\} + \{-10, -20, -30\} = \{90,180, -270\}$.

Acquisition

The system takes observed behaviors as the bases for generating motions to complete tasks. Demonstrations are given by one or more teachers several times. Demonstrated motion trajectories and task-relevant information (e.g., the distances between the end-effector and a target object in a manipulation task) are recorded for modeling and analysis. Demonstration data is recorded with time-stamped vectors. The robot uses the AP to record the demonstrations and store the recorded information in the STM.

Input/Output

The data set from D observed demonstrations can be represented as:

$$S = \{S^1, S^2, \dots, S^d, \dots, S^D\} \quad (31)$$

where d is the index of the observed demonstrations.

For each demonstration, R types of information are chosen to record. They could be the motion trajectories of human hands, the Cartesian poses of task-related objects, the distance between the robot's end-effector and the target objects, etc. Represent them as:

$$S^d = \{{s_1}^d, {s_2}^d, \dots, {s_R}^d\} \quad (32)$$

Each element in this group is a time-stamped vector:

$${s_i}^d = \left({s_i}^d(1), {s_i}^d(2), \dots, {s_i}^d(T)\right)^T \quad (33)$$

where i is the ordinal index of the trajectory in the demonstration group S, and numbers 1, 2,…, T are time steps.

Recorded information is stored in the STM and will be sent to the *Segmentation* and the *Behavior Generalization* module for processing.

Implementation

As stated in the literature review, there are several types of methods to demonstrate behaviors to robots. In our system, we used a Kinect sensor to record the position and orientation of the hand of a human teacher, and used a camera to record the position and the size of an obstacle in the environment.

Kinect is a motion sensing input device by Microsoft for the Xbox 360 video game console and personal computers (Figure 39). Based around a webcam-style add-on peripheral for the Xbox 360 console, it enables users to control and interact with the Xbox 360 and PCs, through a natural user interface using gestures and verbal commands.

Figure 39 Kinect

There are four major functions of Kinect: obtaining the original RGB images from cameras, obtaining the depth information from the infrared sensors, obtaining the audio information using the microphones, and generating skeleton data of human bodies in the environment.

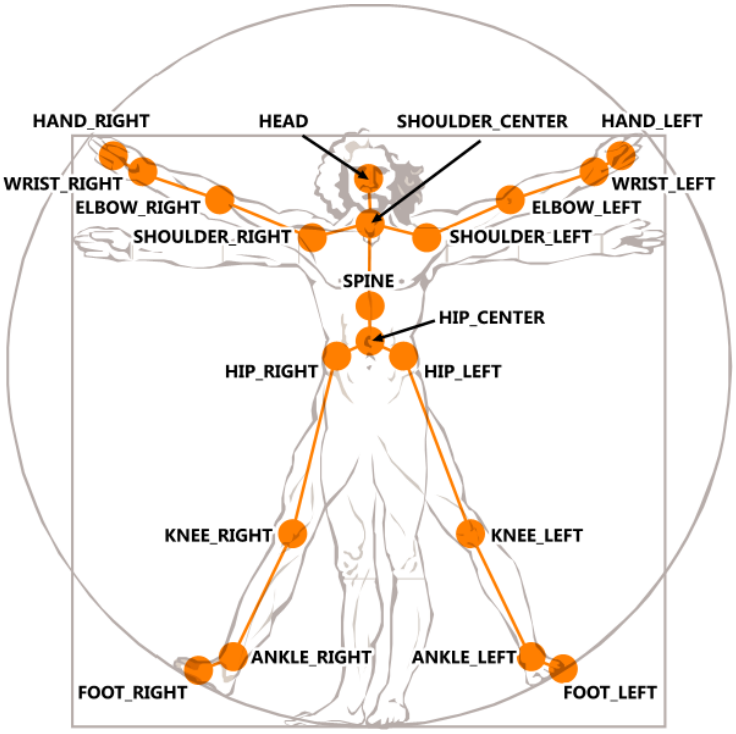


Figure 40 Obtained Skeleton Data from Kinect Software

Figure 40 displays the skeleton data designed in the Kinect Software. The position values of 20 joints on a human body could be obtained from the Kinect Software. Each

position value of a joint is a three-dimensional vector including the X and Y position in an image and the depth information of the joint.

Because of the difference of the configurations between human bodies and our robot, the coordinates we designed for humans' bodies and robotic bodies are totally different. That means that we cannot directly apply obtained position values of joints to robots. Therefore, it is necessary to find a transformation between the coordinates of human bodies and robotic bodies.

Currently, demonstrations are observed using Kinect sensor in a task-space (mostly in the Cartesian space). Because our robot has two arms, we need to observe the positions and the orientations of the hands of a human teacher for our experiments. Here we assume that the human teacher does not move his torso during the demonstration process. This assumption can give us a good reference coordinates. The base coordinates for the human teacher and our robot are both located on the shoulder.

Figure 41 displays the kinematics model used for ISAC.

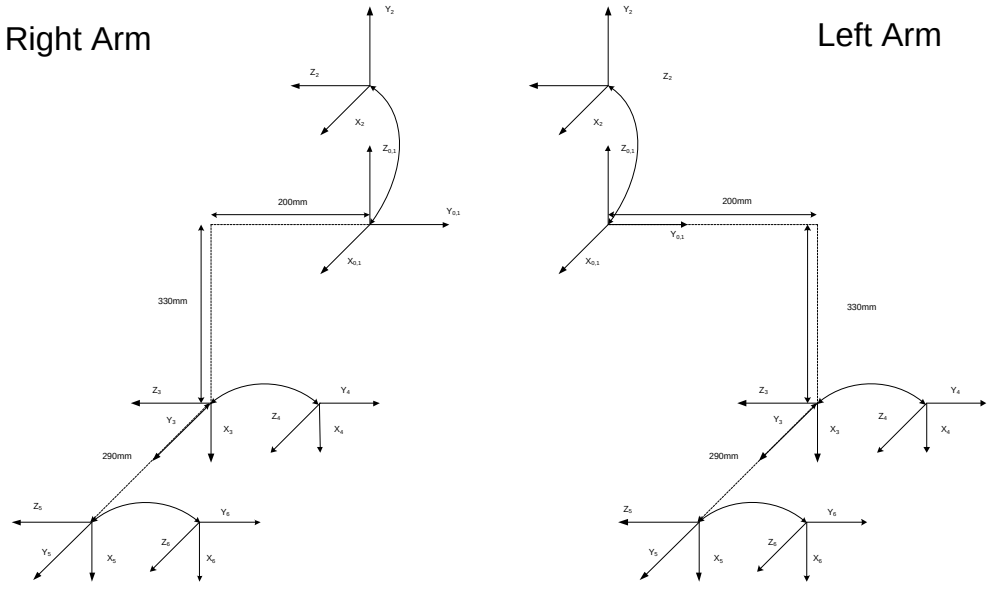


Figure 41 Kinematics Model of ISAC

The data set from D observed demonstrations can be represented as:

$$S = \{S^1, S^2, \dots, S^d, \dots, S^D\} \quad (34)$$

where S^d is a vector representing the d-th observed demonstration. Details on S are summarized in Appendix A.

Behavior Generalization

One key assumption in current research on robotic imitation learning is that human teachers are well-trained and are capable of generating similar trajectories for certain behavior demonstration. In some situations, however, this is not true. I.e., due to different experiences or habits of human teachers, the demonstrations are not always "similar". For example, consider a case where three teachers demonstrate a "Reaching" behavior to reach an object as shown in Figure 42. The circles are the starting points and the stars are the ending points.

The motion trajectories look quite different, which start from different locations, stop at different locations, and have different styles. So how is a robot able to choose a correct trajectory? Since the goal (i.e. reaching a target) is achieved using such different trajectories, it is reasonable to assume that there should be some common features lying within these demonstrations. In our proposed method, we propose to analyze these common features to generalize demonstrations. In Figure 42, the common feature of the two demonstrated "Reaching" behaviors is to minimize the distance between the end-effector of the robot (or the hand of the human teacher) and the target object. From Figure 42, although three motion trajectories start from different locations labeled with

stars, they end with different styles at the same location which is the target object. So that is the common feature for the “Reaching” behavior.

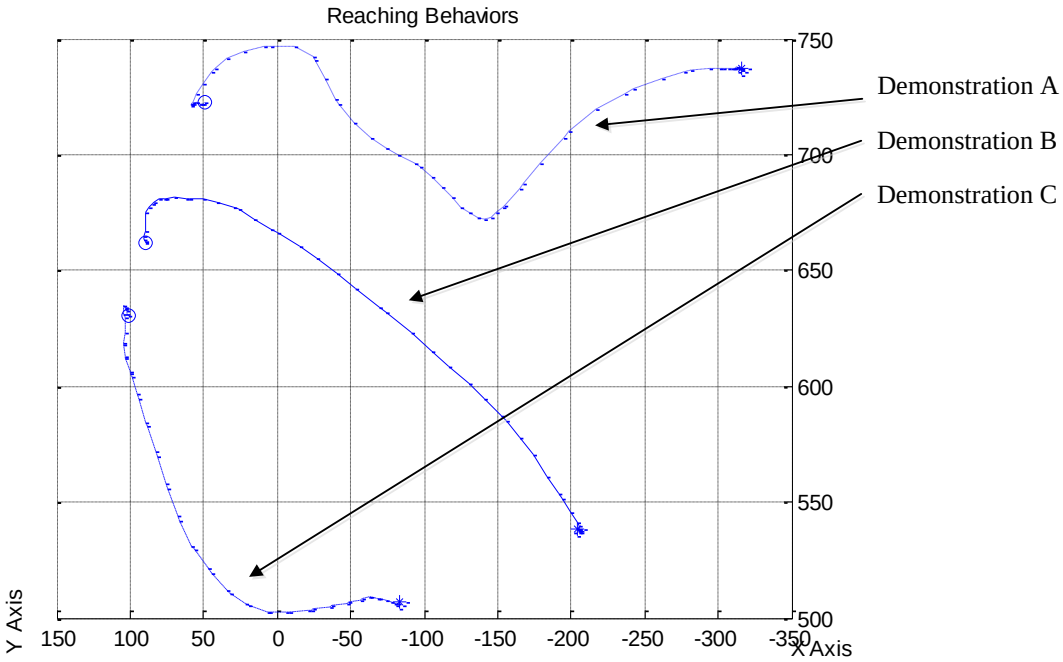


Figure 42 Different “Reaching” Behaviors

Current imitation learning research focuses on how to generate a motion trajectory which is similar to one of the demonstrated trajectories. As explained earlier, an assumption is that all human teachers are well trained and thus all demonstrations are similar. However, all the demonstrations are expected to be different according to different human teachers. We in fact do not need to assume that all the teachers are well trained and all the demonstrations are similar. Instead, we only need to assume the demonstrations are related to the same task. These demonstrations could be described using many task-related features. From this assumption, it is reasonable to analyze the common features of the demonstrations because they are all for the same task.

An important capability of robots is to apply the learned behaviors in new task-relevant situations. If robots can only learn motion trajectories and generate similar motion trajectories, it is very difficult for them to apply the low-level knowledge in different task-relevant situations. In this dissertation, through behavior generalization, robots find the common features of the demonstrated behaviors. Based on that, robots can use higher-level methods to utilize the generalized behaviors and to apply these behaviors to new task-relevant situations, e.g., constructing behavior graphs and generating behavior sequences. This behavior generalization method enables robots to learn new behaviors and apply generalized behaviors in new task-relevant situations flexibly and adaptively.

As stated before, tasks are goal-oriented. If we can find goal-related features from the demonstrations, a task could be described with several features.

Tasks demonstrated by human teachers comprise a set of low-level behaviors. Different tasks require different behaviors. Due to the measurement errors, noise in the environment and inconsistencies in demonstrations of the same task, the obtained motion trajectories could be different. There may be, however, common features latent within the demonstrations. An appropriate analysis and comparison of demonstrated tasks could find the common features hidden in the sampled motion trajectories.

This method considers behaviors to be attribute-based. That is, common internal features found for demonstrated behaviors are represented as a set of attributes. A labeled or named behavior can be described in terms of three attributes: (1) the requisite preconditions or task-specific environmental conditions for execution, (2) internal

constraints which confine the behavior during execution, and (3) post results that characterize the outcome of a behavior:

$$Behavior: \{Name, Pre-Condition, Intenal\ Constraint, Post\ Result\}$$

At the behavior generalization stage, the target is to find the most common feature for pre-condition, internal constraints and post results respectively. The design of the required common features is flexible, and researchers can define their own features. An example of how to design features and generalization methods will be discussed in detail later in this paper.

Before the generalization, a group of features are predefined and stored in the memory system for robots to use. We define three groups of features for pre-condition, internal constraints and post results respectively:

$$X = \{X_{pre}, X_{interna}, X_{post}\} \quad (35)$$

$$X_{pre} = \{{X_{pre}}^1, {X_{pre}}^2, \dots, {X_{pre}}^l\} \quad (36)$$

$$X_{internal} = \{{X_{internal}}^1, {X_{internal}}^2, \dots, {X_{internal}}^m\} \quad (37)$$

$$X_{post} = \{{X_{post}}^1, {X_{post}}^2, \dots, {X_{post}}^n\} \quad (38)$$

l, m, and n are numbers of features for pre-condition, internal constraints and post results respectively.

Using D demonstrations (for one task) from humans, we can compute a probability score for each feature:

$$P_{pre} = \{{P_{pre}}^1, {P_{pre}}^2, \dots, {P_{pre}}^l\} \quad (39)$$

$$P_{internal} = \{{P_{internal}}^1, {P_{internal}}^2, \dots, {P_{internal}}^m\} \quad (40)$$

$$P_{post} = \{{P_{post}}^1, {P_{post}}^2, \dots, {P_{post}}^n\} \quad (41)$$

The most common feature for pre-condition, internal constraints and post results could be found with the following equations:

$$C_{pre} = argmax\left({P_{pre}}^{1}\right) \quad (42)$$

$$C_{internal} = argmax\left({P_{internal}}^{1}\right) \quad (43)$$

$$C_{post} = argmax\left({P_{post}}^{1}\right) \quad (44)$$

Feature $X = \left\{{X_{pre}}^{C_{pre}}, {X_{internal}}^{C_{internal}}, {X_{post}}^{C_{post}}\right\}$ is used to describe a behavior:

$$Behavior = \left\{name, {X_{pre}}^{C_{pre}}, {X_{internal}}^{C_{internal}}, {X_{post}}^{C_{post}}\right\} \quad (45)$$

$$c_{2,i,j} = {\hat{\xi}_{s,i,j}}^{x} - {\xi_{s,i,j-1}}^{x} \quad (46)$$

$$c_{3,i,j} = {\hat{\xi}_{s,i,j}}^{y} - {\xi_{s,i,j-1}}^{y} \quad (47)$$

Input/Output

The input of the Behavior Generalization is the recorded states *S* in equation (32), and the output is the generation results for Pre-Condition, Internal Constraints, and Post Results of the demonstrated behaviors.

Implementation

In our system, pre-conditions, internal constraints and post results are pre-defined as shown in Table 1.

The target of behavior generalization stage is to find suitable conditions, constraints and results for each behavior from demonstrations. From the demonstrations, we used the criterions in Table 2 and Table 3 to find the most common feature for pre-conditions, post results and internal constraints.

Table 1 Pre-Conditions, Internal Constraints, and Post Results

Pre Conditions	Internal Constraints	Post Results
0. Unnecessary	0. Unnecessary	0. Unnecessary
1. Minimize the distance between the hand and the target position	1. Keep similar dynamics	1. Minimize the distance between the hand and the object-related position
2. Keep the distance between the hand and the target position	2. Generate the same trajectory	3, Keep the distance between the hand and the target position
3. Object in hand	3. keep the distance between the hand and the obstacle larger than a predefined value	3. Grasp the object
4. Object not in hand		4. Release the object

Table 2 Criterions for Pre Conditions and Post Results

Pre Constraints/ Post Results	Criterions
0. Unnecessary	None
1. Minimize the distance between the hand and the target position	The distance between the hand and the target position is smaller than a threshold value
2. Keep the distance between the hand and the target position	The distance between the hand and the target position is larger than a threshold value
3. Object in hand	The closed signal from the gripper and the distance between the hand and the target position is smaller than a threshold value
4. Object not in hand	The opened signal from the gripper

For Feature 1 and 2 in Table 2, the distance d between the hand and the target position is computed directly using Euclidean distance and then normalized.

These distances are then normalized to probability values.

$$F_1 = e^{-\frac{(d-d_0)^2}{k_1}}, l = 1,2 \quad (48)$$

$$F_2 = e^{\frac{(d-d_0)^2}{k_2}} \quad (49)$$

where F_1 and F_2 are similarity scores for Feature 1 and 2, k_1 and k_2 are the normalization parameters.

F3 and F4 are determined by the measurements of control signal from the grippers. If the gripper is closed and object is in hand, $F_3 = 1$; otherwise, $F_3 = 0$. the gripper is opened and object is not in hand, $F_4 = 1$; otherwise, $F_4 = 0$.

The most common feature for the Pre-Condition and Post-Results are determined by the corresponding largest value of F_1, F_2, F_3, and F_4.

Table 3 Criterions for Internal Constraints

Internal Constraints	Criterions
1. Keep similar dynamics	DTW similarities between normalized motion trajectories are larger than a threshold value
2. Generate the same trajectories	DTW similarities between motion trajectories in original data space are larger than a threshold value

For Feature 1 in Table 3, Dynamic Time Warping (DTW) [Berndt and Clifford, 1994] distances between two demonstrations are computed first. Then we have a matrix to describe the distances.

$$d^{DTW} = \begin{bmatrix} 1 & d^{DTW}{}_{1,2} & \cdots & d^{DTW}{}_{1,D-1} & d^{DTW}{}_{1,D} \\ 0 & 1 & & d^{DTW}{}_{2,D-1} & d^{DTW}{}_{2,D} \\ \vdots & & \ddots & \vdots & \\ 0 & 0 & \cdots & 1 & d^{DTW}{}_{D-1,D} \\ 0 & 0 & & 0 & 1 \end{bmatrix} \quad (50)$$

where D is the number of demonstrations.

The elements of first row are normalized using the following equation:

$$\bar{d}^{DTW}{}_{i,j} = e^{-\frac{d^{DTW}{}_{i,j}{}^2}{k_v}} \quad (51)$$

Then the normalized variance of the first row of $\bar{d}^{DTW}$ is computed as F1, i.e. the probability score of this feature.

For Feature 2, the DTW distance is computed in a normalized range [0,1] first, and then the following steps are the same.

The most common feature for internal constraints is determined by the maximum value of F_1, and F_2.

By analyzing the demonstrated behaviors, the robot tries to assign a most common feature to the pre-conditions, internal constraints, and post-results respectively. The generalized results can be represented as shown in equation (43). Each common feature is described using a number which is related to the pre-designed features table in Table 1.

Representation and Storage

Generalized behaviors are stored in the LTM for future use. When required, the CEA can query the stored information in the LTM to construct a behavior graph and searches required behavior in the behavior graph to generate behavior sequences.

Input/Output

The input of the "Goal-Based Features for Basic Behaviors" is the results obtained in equation (43): $\left({X_{pre}}^{C_{pre}}, {X_{internal}}^{C_{internal}}, {X_{post}}^{C_{post}}\right)$ and the semantic name assigned to the generalized behavior by a human teacher.

Implementation

Microsoft Access 2010 is used as a database tool to store the generalized behaviors and their related features. Table 4 displays stored basic behaviors and their related Pre-Conditions, Internal-Constraints, and Post-Results.

Table 4 Stored Basic Behaviors

Behavior ID	Behavior Name	Pre Cond	Post Cond	Internal Cond	Prerequisite
1	reaching	0	1	0	null
2	pushing left	1	1	0	null
3	pushing right	1	1	0	null
4	pushing forward	1	1	0	null
5	moving	0	1	0	null
6	lifting	0	1	0	null
7	droping	0	1	0	null
8	releasing	3	4	0	null
9	grasping	1	3	0	null
10	yo-yo motion	3	0	2	null
11	starting	0	0	0	null
12	ending	999	0	0	null

Semantic names for these names are assigned by a human teacher. From the generalization results of the demonstrated behaviors, the numbers related to Pre-Conditions, Internal-Constraints, and Post-Results which are predefined in Table 1 are assigned to the behaviors and stored in the database. In the section of "Behavior Graph Construction", a behavior graph will be constructed using the information stored in Table 4 by finding the matching between the pre-conditions and post-results.

Command Parsing

An example of a generation command is described as:

Push (the box) to the right

The content in the brackets could be modified according to the requirements of tasks. For example, *the box* could be replaced with *the pen*, *the toy*, etc.

The mail goal of this task is to push the box to the right in the environment. The robot needs to complete a task which is constrained by this goal.

In this dissertation, the description of a given command is represented as:

$$G_1\ (\theta_1)\ while\ \{G_2(\theta_2)\ \&\ G_3(\theta_3)\ \&\ (G_4(\theta_4)|G_5(\theta_5)), \dots, \&\ G_n(\theta_n)\} \qquad (52)$$

G_1 is the main goal, while G_i $(i = 2,3, \dots, n)$ are subordinated goals. All goals have their related parameters which are represented in the brackets as θ_i $(i = 1,3, \dots, n)$. The symbols of "& "and "|" represents the "and" and "or" relationships between the subordinated goals.

An example of a learning command is described:

I will show you how to use the reaching the object behavior

In this dissertation, the description of a given learning command is represented as:

$$L(b, p_1, \dots, p_n) \qquad (53)$$

b is the name of the behavior to be learned, and $p_1, \dots, p_n$ are parameters related to this behavior. In the above example, b is *reaching* and p_1 is *the object.*

Input/Output

The input of the Command Parsing is a speech command which obeys the form of equation (50) and (51). After parsing the command described in equation (50), the output of the Command Parsing is a required main goal behavior and several subordinated goal s with task-related parameters; After parsing the command described in equation (51), the output of the Command Parsing is the name of the behavior to be learned

Implementation

Microsoft Speech Recognition Library is used for speech recognition in our system.

The grammars of the commands are pre-defined as follows:

$$verb\ 1 + Object + Parameter\ 1 \quad (54)$$

$$verb\ 1 + Parameter\ 1 + Parameter\ 2 \quad (55)$$

$$verb\ 1 + Parameter\ 1 + while + verb\ 2 + Parameter\ 3 \quad (56)$$

$$verb\ 1 + Parameter\ 1 + Parameter\ 2 + while + verb\ 2 + Parameter\ 3 \quad (57)$$

$$learn + behavior + parameter\ 1 \quad (58)$$

$$learn + behavior + parameter\ 1 + Parameter\ 2 \quad (59)$$

In these grammars, $verb\ 1$, $verb\ 1$, $Parameter\ 1$, $Parameter\ 2$, and $Parameter\ 3$ are pre-designed lexicons which are defined as:

$$verb\ 1 = \{reach, push, move, lift, drop, wave, shake, play, release, graps\} \quad (60)$$

$$verb\ 2 = \{avoid, keep\} \quad (61)$$

$$Parameter\ 1 = \{box, ball, toy\} \quad (62)$$

$$Parameter\ 2 = \{to\ the\ left, to\ the\ right, forward, backward\} \quad (63)$$

$$Parameter\ 3 = \{obstacles, dynamics\} \quad (64)$$

In the recognition process, a given speech command is categorized into different grammars by using the Microsoft Speech Recognition Library and the key lexicons are extracted in different grammars. $verb\ 1$ is the main goal behavior and is extracted first.

$Parameter\ 1$ and $Parameter\ 2$ are extracted as the related parameter for $verb\ 1$. $verb\ 2$ is the subordinated goal and $Parameter\ 1$ is extracted for this subordinated goal.

The extracted lexicons are sent to the CEA as the searching criterion to search the behavior graph and the required parameters to generate behavior sequences.

Behavior Library

Behavior Library stores the learned behaviors and constructs a behavior graph based on the generalized features to describe the relationships among these generalized behaviors. All learned behaviors are represented as vertexes in a behavior graph, and the edges are defined by matching the pre-condition of a behavior and the post-result of another behavior. If the robot finds that the pre-condition of a behavior and the post-result of another behavior match, an edge is added.

Input/Output

The input of the Behavior Library is the generalized behaviors stored in the LTM, and the pre-defined behavior motion trajectory generation methods. The constructed behavior graph is stored in the Behavior Library for the *DMM* and the *Behavior Sequence Generation* to use.

Implementation

Figure 43 displays the pseudo code of the construction stage.

```
1    for i=1 to number of behaviors
2        vertex.add(behavior[i])
3    end
4    for i=1 to number of behaviors
5        for j=1 to number of behavior
6            if (behavior [j] !='ending')
7                if (the post condition behavior[j] == the pre constraints of
                 behavior [i])
8                edge.add(behavior[j], behavior[i])
9            end
10           else if (behavior[i]== "starting")
11               edge.add(behavior[j], behavior[i])
12           end
13           else if(behavior [j]== "ending")
14               edge.add(behavior[j], behavior[i])
15           end
16       end
17   end
```

Figure 43 Pseudo Code of Constructing a Behavior Graph

Line 1-3 add all the learned behaviors into the behavior graph as vertexes. Line 4-17 add edges between all vertexes. In Line 6-9, the results of matching the pre-condition and the post-result between two behaviors determine whether an edge can ben added to connect the two behaviors. Line 10-12 add edges to the "Starting" behavior from all other behaviors. Line 13-15 add adges from the "Ending behavior" to all other behaviors.

Based on the matching of the pre-conditions and the post results from the stored information in Table 4, a behavior graph is constructed.

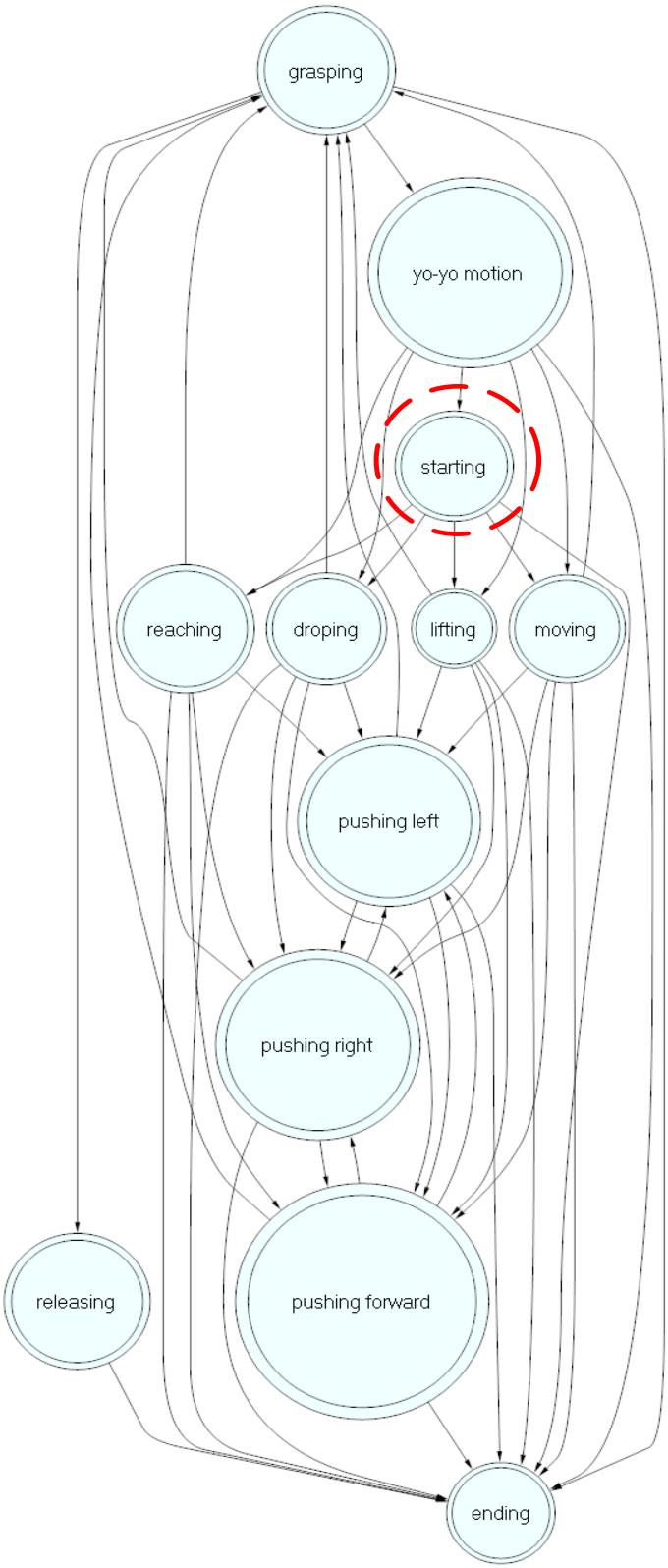


Figure 44 A Constructed Behavior Graph

In Figure 44, learned basic behaviors are represented as vertexes in this behavior graph. The vertexes reflect the transitions among these behaviors. Robots do not need to learn all these edges/transitions from human teachers. They just need to learn the common features of these behaviors and construct the transitions by matching the Pre-Conditions and Post-Results of these behaviors. The advantage of this method is to enable the robot to find a behavior sequence by itself based on this graph.

Behavior Sequence Generation

Given a task, a robot needs to complete the task using some behaviors. If the robot has already learned the required behavior, it can generate a behavior sequence starting from the "starting" behavior, ending at the required behavior. An "ending" behavior is added into the behavior sequence as the last elements for robots to finish the behavior sequence.

Input/Output

The input is a behavior verb which is extracted in the Command Parsing, and the output is a behavior sequence:

$$B = \{behavior\ 1, behavior\ 2, \dots, behavior\ n\} \quad (65)$$

Implementation

In our method, we choose to use Dijkstra's algorithm [Dijkstra, 1959] to find a shortest path in a constructed directed behavior graph. Dijkstra's algorithm, conceived by Dutch computer scientist Edsger Dijkstra in 1956 and published in 1959, is a greedy

searching algorithm to find a shortest path in a directed graph by repeatedly updating the distances between the starting node and other nodes until the shortest path is determined.

Let the node at which we are starting be called the initial node. Let the distance of node Y be the distance from the initial node to Y. Dijkstra's algorithm will assign some initial distance values and will try to improve them step by step.

1. Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes.

2. Mark all nodes unvisited. Set the initial node as current. Create a set of the unvisited nodes called the *unvisited set* consisting of all the nodes except the initial node.

3. For the current node, consider all of its unvisited neighbors and calculate their *tentative* distances. For example, if the current node A is marked with a distance of 6, and the edge connecting it with a neighbor B has length 2, then the distance to B (through A) will be 6+2=8. If this distance is less than the previously recorded tentative distance of B, then overwrite that distance. Even though a neighbor has been examined, it is not marked as "visited" at this time, and it remains in the *unvisited set*.

4. When we are done considering all of the neighbors of the current node, mark the current node as visited and remove it from the *unvisited set*. A visited node will never be checked again.

5. If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the *unvisited set* is infinity (when planning a complete traversal), then stop. The algorithm has finished.

6. Select the unvisited node that is marked with the smallest tentative distance, and set it as the new "current node" then go back to step 3.

Using Dijkstra's algorithm, in a behavior graph, a shortest path can be generated from the “Starting” node to the node related to the $verb\ 1$,. A behavior sequence, which is composed of the nodes on the found path, is generated for the DMM and the IRS to evaluate.

Motion Trajectories Generation

We need to design feature-related generation methods for robot to generate motion trajectories for all the behaviors in a behavior sequence. Since our description of a task is composed of one main goal and several subordinated goals, the generation method for a behavior should also be composed of several generation methods.

Input/Output

The input is a behavior sequence with task-related parameters: $B = \{behavior\ 1, behavior\ 2, \dots, behavior\ n\}$, and the output is a motion vector $P \in \Re^3$, which specifies the via points on a motion trajectory in the Cartesian space.

Implementation

Table 5 Behavior Motion Trajectory Generation Methods

Internal Constraints	Post Constraints	Generation Method
0	1	2^{nd}- order attractor
0	2	Potential Field
0	3	Close end-effector
0	4	Open end-effector
1	1	Dynamic Movement Primitive (DMP)
1	2	Potential Field+ Dynamic Movement Primitive (DMP)
1	3	Close end-effector
1	4	Open end-effector
2	X	Generate the same trajectories

The combinations of the Internal-Constraints and Post-Results determine the generation methods for behaviors. As shown in Table 5, 2nd-order attractor, Potential Field [20], DMP[3], DMP + Potential Field[21], and generating the same trajectories are used as our pre-defined generation methods as well as well as opening and closing end-effector. Using these methods, parameters could be changed to adapt to different situations for robots to generate similar motion trajectories to complete tasks.

--2^{nd}-order Attractor

The end-effector of the robot will be moved to a point where the distance between the end-effector and the manipulated object is the same as the distances in the demonstrations. We applied a second-order spring-damping attractor for robots to reach the target position.

$$\tau \dot{z} = \alpha_z(\beta_z(g - y) - z) \quad (66)$$

$$\tau \dot{y} = z \quad (67)$$

where y is the position values on the generated trajectory, z is the velocity, and g is the

goal position. α_z and β_z are constants, which are selected to ensure the system over-damped.

--*Generating the Same Trajectories*

We can model the motion trajectory in the Cartesian space or in the joint space. In current research on imitation learning, Gaussian Process (GP), Hidden Markov Model (HMM), Receptive Field Weighted Regression (RFWR), and Gaussian Mixture Model (GMM) are used by researchers to generalize and model the demonstrations from humans.

A Receptive Field Weighted Regression (RFWR) method is used for generalizing and modeling the demonstrations. In our model, the temporal information is considered as the enquiry point. Assume N points are sampled in a time period, and t reflects the temporal information. Given the temporal information t, the corresponding predicted data points could be computed using our model. The temporal information could also be understood as timing steps.

Assume that i^{th} observed motion trajectory from i^{th} demonstration could be modeled as a RFWR model as shown below:

$$f_i = \frac{\sum_{j=1}^{N} \Psi_{ij} w_{ij}}{\sum_{j=1}^{N} \Psi_{ij}} \quad (68)$$

where N is the number of the receptive basis functions, j is the index of the receptive basis functions, and i is related to the index of the demonstrations.

Ψ_{ij} is a receptive basis function, which is distributed in the space.

$$\Psi_{ij} = exp\left(\frac{1}{-2h_i j^2}\left(t - c_{ij}\right)^2\right) \quad (69)$$

where, c_{ij} is the center of the j^{th} basis function for i^{th} RWFR model, which is distributed in the input space, and h_{ij} is the bandwidth. w_i is learned by analyzing the demonstrated motion trajectories using the following equation:

$$\boldsymbol{w_i} = \left(\boldsymbol{\Psi_i}^{\boldsymbol{T}}\boldsymbol{\Psi_i}\right)^{-\mathbf{1}}\boldsymbol{\Psi_i}^{\boldsymbol{T}}\boldsymbol{t} \quad (70)$$

$\boldsymbol{t}$ is the observed position vector in the demonstrations.

--*Dynamic Movement Primitives (DMP)*

If the overall goal of the task is to reach to a target position while keeping the dynamics of the generated motion trajectories similar to the demonstrated motion trajectories, we applied DMP to generate motion trajectories.

$$\tau\dot{z} = \alpha_z(\beta_z(g - y) - z) + f \quad (71)$$

$$\tau\dot{y} = z \quad (72)$$

where f is a non-linear RFWR model which can be constructed in equation (66). y is the position values on the generated trajectory, z is the velocity, and g is the goal position. α_z and β_z are constants, which are selected to ensure the system over-damped.

--*DMP + Potential Field*

If the overall goal of the task is to reach to a target position while keeping the dynamics of the generated motion trajectories similar to the demonstrated motion trajectories and avoiding the obstacles in the environment, we applied our potential field based DMP algorithm [Tan et al., 2011] to generate motion trajectories. In this algorithm,

the goal state in equation (69) is modified according to distances between the position of the end-effector and the position of the obstacle.

Because the original DMP method does not provide the function of obstacle avoidance, the motivation was to propose an extension of DMP which can generate new trajectories with similar dynamics to the demonstrations and avoid obstacles in the working environment [Tan et al., 2011].

In DMP, the trajectory is planned step by step in an incremental way. The elements in $\vec{y}$ are the generated points on the trajectory. When the current state $\vec{y}(i)$ is known, the next state $\vec{y}(i+1)$ is calculated by equation (69) and (70).

The basic idea is to move the goal state $\vec{g}$ to a virtual goal position $\vec{g}_{temp}$ by adding an impedance factor $\vec{g}_{impedance}$, when the current state is in the impedance area around an obstacle,

$$\vec{g}_{temp} = \vec{g} + \vec{g}_{impedance} \quad (73)$$

$\vec{g}_{impedance}$ is generated by the impedance field around the obstacle. In our method, we decomposed the force generated by potential field into the tangent direction and the centrifugal direction for simplicity.

Details on this potential field based DMP algorithm are summarized in Appendix B.

Internal Rehearsal System

The IRS not only internally simulates behaviors, but also predicts outcomes based on current situations to obtain evaluations to the DMM.

Input/Output

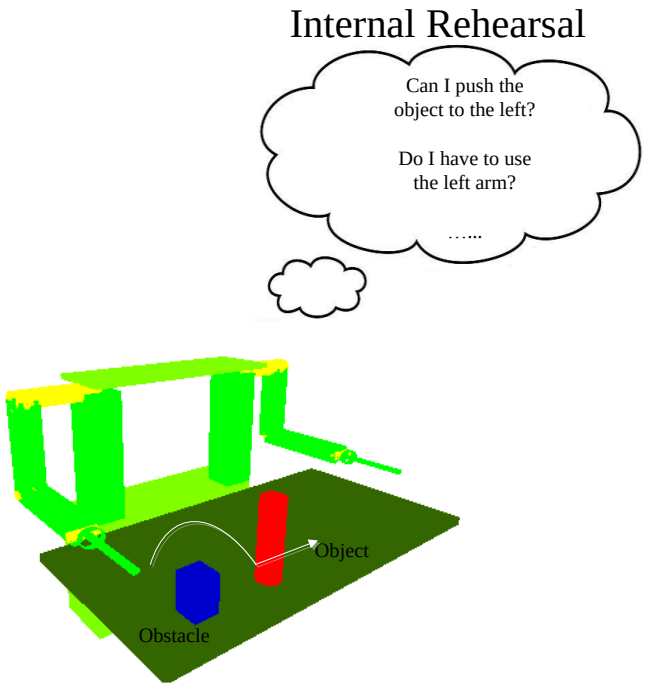


Figure 45 Internal Rehearsal

The input of the IRS is a vector which represents all the via points on a generated behavior sequence and a motion trajectory, task-related parameters, and the environmental information. The output of the IRS is the evaluation results. Figure 45 displays a typical usage of IRS. The robot is asked to push the object and it needs to evaluate whether the object is located in its working space and whether it can successfully push the object to the left while avoiding obstacles. The input is the generated motion trajectories, and the positions and sizes of the object and the obstacle on the table, and the output is a judgment that whether the robot can complete this task.

Implementation

VC# and OpenGL are used to design the IRS component in our system. There are three crucial modules in the IRS: Kinematics Module, Physical Limitation Checking Module, the Obstacle-Collision Detection Module, and Behavior Sequence of Dual Arms Checking Module.

The Kinematics Module map the via points on the motion trajectory to the joint angles of ISAC

The Physical Limitation Checking Module checks whether the operation point is within the working-space of the robot. For each point on the motion trajectory, the robot uses the IRS to check whether it could be reached using the following two equations:

$$P_x{}^2 + P_y{}^2 + P_z{}^2 < length\_shoulder^2 + length\_arm^2 \quad (74)$$

$$P_x{}^2 + P_y{}^2 > length\_shoulder^2 \quad (75)$$

where P_x, P_y, and P_z represents the 3-dimensional position value of the points to be checked, and the $length\_shoulder$ and $length\_arm$ are the length of the shoulder and the arm respectively.

If the position values satisfy both of these equations, the IRS considers that this checked point satisfies the requirement for Physical Limitation Checking.

The Obstacle-Collision Detection Module checks whether the arm of the robot collides with obstacles in the environment. For each point on the motion trajectory, the robot uses inverse kinematics to compute the angular value of each joint and computes the position of each joint in the Cartesian space.

$$\Theta = inverse\ kinematics(P_x, P_y, P_z) \quad (76)$$

$$(P_1, P_2, P_3, P_4, P_5, P_6) = forward\ kinematics(\Theta) \quad (77)$$

The Obstacle-Collision Detection module checks whether Link 2, Link 3, and the end-effector collide with the obstacles in the environment. The widths of Link 2, Link 3, and the end-effector are taken into consideration and defined as: w_2, w_3, and w_e.

The points on Link 2 are represented as P_{23}, the points on Link 3 are represented as P_{34}, and the points on the end-effector are represented as P_e.

The connected lines between joint 2 and joint 3, joint 3 and joint 4, and joint 6 and the manipulation point are computed respectively. P_{23}, P_{34}, and P_e are computed by expanding these lines with w_2, w_3, and w_e respectively.

The robot IRS to check whether the collision happens using the following two equations:

$$\|P_{23}(i) - O\| > size\ of\ Obstacle \quad (78)$$

$$\|P_{34}(j) - O\| > size\ of\ Obstacle \quad (79)$$

$$\|P_3(k) - O\| > size\ of\ Obstacle \quad (80)$$

where i is the index of the points to be checked, O is the position value of the center of the obstacle, and $size\ of\ Obstacle$ is the size of the obstacle.

If the position value of a checked point satisfy both of the two above equations, the IRS considers that this checked point satisfy the requirement for Obstacle-Collision Detection.

If the IRS found that all the via points on the generated motion trajectory satisfy the two requirements, it returns a zero value to the DMM, otherwise it returns 1 for violation of the Physical Limitation Checking and 2 for the violation of the Obstacle-Collision Detection.

In some situations, the robot needs to use its both arms to complete a task. So, given a behavior sequence, the robot needs to decide which arm it should use to execute the behaviors.

Assume we have a sequence: $\{B_1, B_2, B_3, \ldots, B_n\}$. The robot will first try to use one of its arms to execute B_1. If the post-result of B_1 satisfy the pre-condition of B_2 by using the same arm, the robot uses the same arm to execute B_2. If the post-result of B_1 cannot satisfy the pre-condition of B_2 by using the same arm, the robot uses the other arm to execute B_2. The robot then decides to use the left arm or right arm for B_3, ..., and B_n until all the behaviors are assigned to one arm using the same method. In this dissertation, we do not design experiments to use both arms. But we want to point out that our method could be applied to use both arms using the method described above.

Summary

This chapter describes an integrated system which combines cognitive control with imitation learning. Specific designs of the components in this system were explained in detail. In the next chapter, this system is applied for a humanoid robot, ISAC, to carry out three experiments to validate our design approach.

CHAPTER IV

SYSTEM IMPLEMENTATION, EXPERIMENTAL DESIGN AND RESULTS

System Implementation

The designed system was implemented to enable a humanoid robot, named ISAC, to learn graspless behaviors from human demonstrations, generalize behavior features, apply learned behaviors in task-relevant situations, and switch strategies by choosing using its right arm or its left arm to achieve required task goals.

Experimental Design

Hardware

--Humanoid Robot

The hardware platform used for this experiment is the ISAC humanoid robot, shown in Figure 46. ISAC has two arms, each of which has 6 Degree-of-Freedom (DOF), driven by pneumatic air muscles [Kawamura et al., 2000].

Rubbertuators

Rubbertuators [Daerden and Lefeber, 2002], which are pneumatic driven, are used to drive the joints of ISAC. The rubbertuators shorten by increasing its enclosed volume, and they will contract against a constant load if the pneumatic pressure is increased.

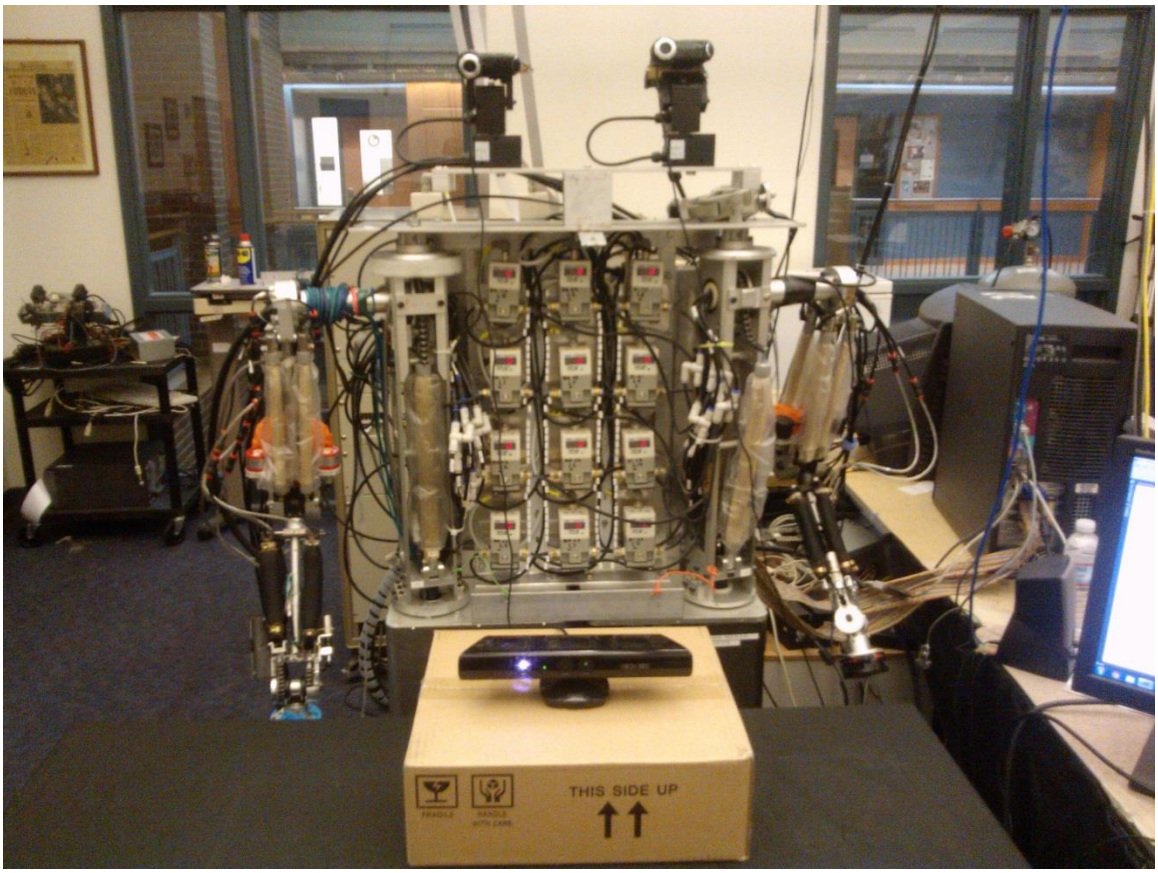


Figure 46 ISAC Robot

Encoders

Optical encoders made by Sumtak [Sumtak] are attached to the joints of each arm for ISAC to obtain the feedback of the current joint angles and to enable closed-loop feedback control. The resolutions of the encoders are 0.087890625 degree.

Gripper

The Gripper used for ISAC is pneumatic driven. Because of the limitation of the hardware and limitation of this system to the graspless “object handling” tasks, the gripper is always closed using tapes.

--*Kinect Sensor*

In our designed experiments, a Kinect sensor is used to assist ISAC to track the position of the hand of a human teacher. The Kinect sensor is used for ISAC to locate the positions and sizes of the target object and obstacle in the environment. Kinect is a motion sensing input device made by Microsoft [Microsoft] for the Xbox 360 video game console and personal computers.

There are four major functions of Kinect: obtaining the original RGB images from one camera, obtaining the depth information from the infrared sensors, obtaining the audio information using the microphones, and generating skeleton data of human bodies in the environment.

Software

Several software modules are implemented for ISAC to perform required tasks in our designed experiments.

--*Operating System*

ISAC

The functions included in the CEA are implemented using Microsoft Visual C++ 2010. LTM is developed using Microsoft Visual Access 2010, and is used for ISAC to store information in a database. A software package, named QuickGraph, is used for ISAC to construct a behavior graph by using the stored information in the LTM, and to visualize the behavior graph on the designed diagram. This program continuously sends out the position and size values of the segmented target object and obstacles to the STM, the information of which is used by the CEA. Command parsing is implemented using

Microsoft Speech Recognition Library, which defines grammars and lexicons, and extracts useful information from recognized commands.

Kinect

The Kinect module is developed using Microsoft Visual C# 2010 in Microsoft Windows 7. It is used for robots to track the position of the hand of the human teacher is based on Natural User Interface (NUI) provided by Microsoft Corporation. As shown in Chapter III, using NUI, ISAC can track the position values of the 20 joints on a human body. In the learning stage, the position of the right hand is used to describe the motion trajectories of demonstrated behaviors.

--*Visual Sensor Processing*

The vision module is used for robots to locate the positions and shapes of the target object and obstacles in the environment. The vision module is developed using Microsoft Visual C# 2010 in Microsoft Windows 7. This module is based on an open source software package, named OpenCV. When a target object and two obstacles are placed on the table in front of ISAC, this module segment the object and obstacle areas using color information in the HSV space. The sizes of the object and obstacles are determined by finding the radius of the segmented area. This program continuously sends out the position and size values of the segmented target object and obstacles to the STM, the information of which is used by the CEA.

--*Arm Control*

The arm control module is developed using Microsoft Visual C++ 6.0 in Microsoft Windows 2000. A closed-loop PID control method is implemented for ISAC to read the encoder value, and to send the control signal to control the air pressure of the air-

muscles which drives the arms of ISAC. This module receives position values of the via points on a motion trajectory as the input, and sends out voltage values for controlling the air-muscles as the output.

--*Simulator*

A simulation environment is required for ISAC to internally evaluate the generated behaviors given some tasks. This environment is developed using Microsoft Visual C# 2010 in Microsoft Windows 7. An open source software package, OpenGL, is used as the basis of implementing such simulation environment. Three modules are implemented in the Simulator: Display Module, Kinematics Module, and Evaluation Module. Evaluation Module receives the motion trajectory to be evaluated from the CEA. The collision checking and working space checking are implemented for the evaluation module as discussed in Chapter III. The via points on the motion trajectories are sent to the Kinematics Module to compute the joint angles of ISAC and then sent to the Display Module for visualization.

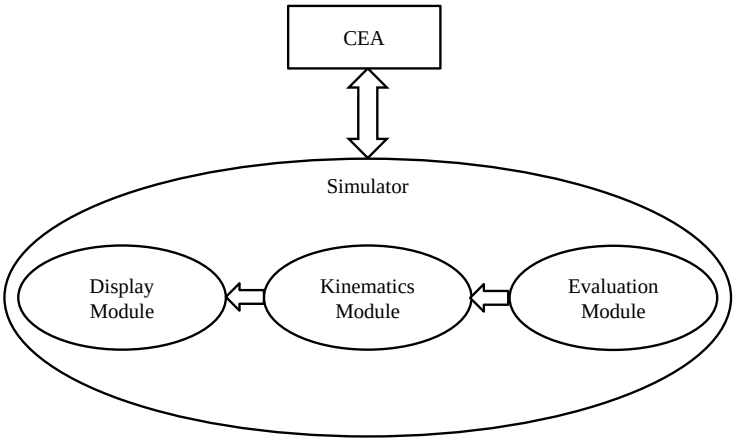


Figure 47 Simulator

In the Evaluation Module, some rules are used to generate Boolean values for the CEA to use. Assume the current being evaluated via points is p_i. Two Boolean values are

defined for these rules. *workingSpaceChecking* is used to check whether p_i is within the working space of ISAC and *collisionChecking* is used to check whether the arm of ISAC collides with the obstacle given the position of the end-effector is p_i.

if p_i is out of the working space of ISAC, then *workingSpaceChecking* is false;

if the arm of ISAC collides with the obstacle given the position of the end-effector isp_i, then *collisionChecking* is false;

if *workingSpaceChecking* and *collisionChecking* are true and p_{i+1} exists, then evaluation continues and start checking p_{i+1}

if *workingSpaceChecking* and *collisionChecking* are true and p_{i+1} does not exist, then evaluation stops and return a Boolean value representing "evaluation successes"

if *workingSpaceChecking* or *collisionChecking* is false, then evaluation stops and returns a Boolean value representing "evaluation fails".

Communication

Figure 48 displays the computers we used for these modules:

Computer Octavia is used for the control Module, computer Sally is used for the vision module, and a laptop is used for the CEA and Memory parts. The communication among these computers is based on TCP/IP socket programming. The cameras are connected to Sally using USB cables and the computer used a PCI card to send the control signals to the regulars on ISAC.

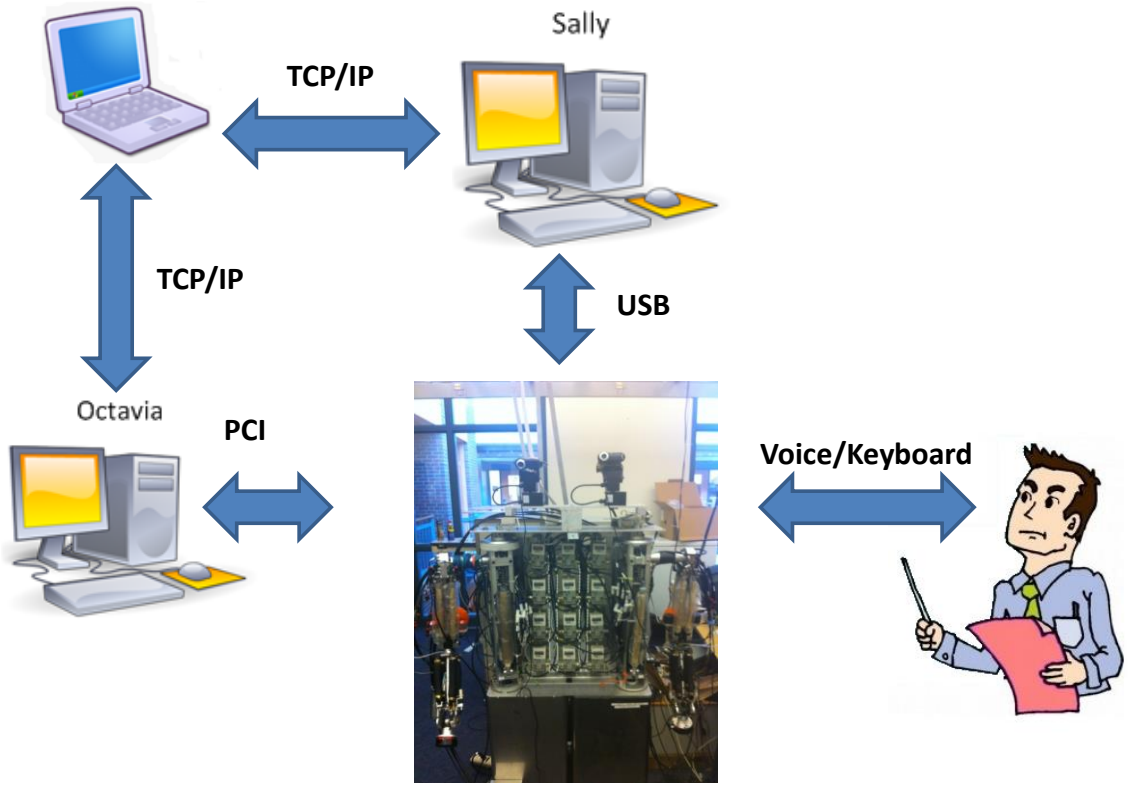


Figure 48 Computers

The communication between the laptop and computer Octavia is based on TCP/IP sockets. The position values of the via points on a motion trajectory are sent to computer Octavia. The X, Y, and Z position values of these points are represented as short variables (2 bytes). Each data packet is composed of 1 operation code and 100 position values are sent from the labtop to computer Octavia. There are 602 bytes in each packet:

Packet[0] = Operation Code

Packet[1], Packet[2], Packet[3] = X,Y,Z coordinates of Point 1;

Packet[4], Packet[5], Packet[6] = X,Y,Z coordinates of Point 2;

…

Packet[208], Packet[209], Packet[300] = X,Y,Zcoordinates of Point 100;

The communication between the labtop and computer Sally is based on TCP/IP sockets. In the learning stage, the position of the hand of the human teacher and the position and sizes of objects are sent to labtop. The X, Y, and Z position values of these points are represented as double variables (4 bytes). Each data packet is composed of 100 position values of the hand of the human teacher, the position value of the target object at the beginning of the demonstration, and the position value of the target object at the end of the demonstration. There are 1224 bytes in each data packet:

Packet[0], Packet[1], Packet[2] = X,Y,Z coordinates of Point 1 of the hand;

Packet[3], Packet[4], Packet[5] = X,Y,Z coordinates of Point 2 of the hand;

…

Packet[297], Packet[298], Packet[299] = X,Y,Z coordinates of Point 100 of the hand;

…

Packet[300], Packet[301], Packet[302] = X,Y,Z coordinates of the object at the beginning of the demonstration;

Packet[303], Packet[304], Packet[305] = X,Y,Z coordinates of the object at the beginning of the demonstration;

Experiments designed were to demonstrate that the following requirements satisfied:

1. Parse speech commands for ISAC to determine the requirements of a given task.
2. Observe the demonstration from a human teacher, record the recorded motion trajectory of the hand of the human teacher.

3. Generalize demonstrated behaviors and store them in the LTM
4. Construct behavior graphs for ISAC to use
5. Generate behavior sequences by finding a path from the "Starting" behavior to the required behavior in the behavior graph
6. Generate similar motion trajectories for behaviors when the common feature of the generated behavior is to keep similar dynamics
7. Switch strategies or processes to achieve required goals

Three types of experiments are designed and implemented to evaluate our designed system. Experiment 1 is used for ISAC to learn and apply "Reaching", "Pushing Left", and "Pushing Right" behaviors. ISAC needs to generalize the demonstrated "Reaching", "Pushing Left", and "Pushing Right" behaviors and store them in LTM. In Experiment 2, ISAC learns how to play a Yo-Yo. ISAC has learned the "Reaching" behavior in experiment 1, so it does not need to learn it again. It just needs to learn "Grasping" and "Yo-Yo Motion" behaviors and combine them with the already learned "Reaching" behavior to assemble a behavior sequence to play the Yo-Yo. Experiment 2 validates the long-term memory part of the system and also generates motion trajectories which are similar to the demonstrations. Experiment 3 evaluates the performance of high-level cognitive control processes on the basis of Experiment 1. A target object is placed on a table in front of ISAC. ISAC needs to determine whether the object can be pushed using either arms while avoiding obstacles in the environment. ISAC switches strategies to complete the task goals if necessary.

Experiment 1: Reaching and Pushing

Experiment 1A: Unsupervised Reaching and Pushing

--*Objective*

The objective of this experiment is to investigate how the system learns the observed behaviors.

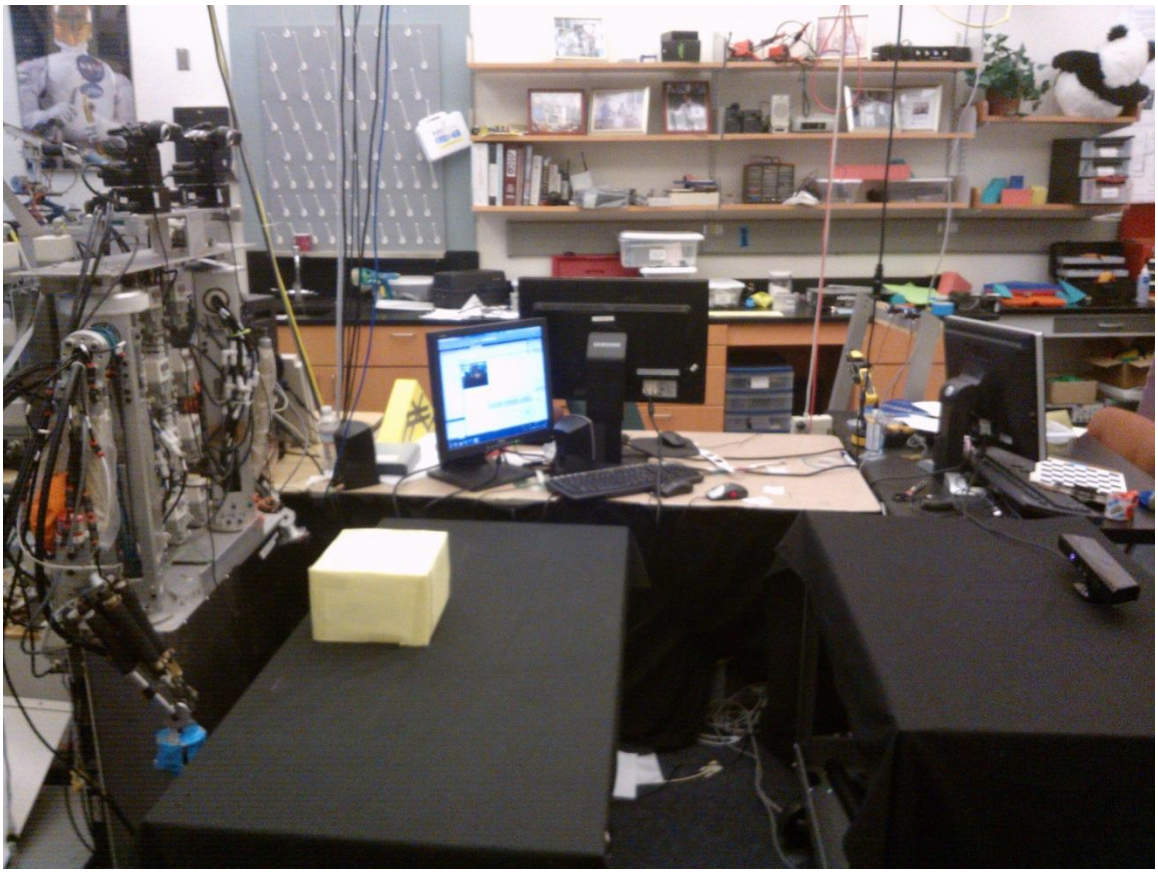
--*Experimental Setup*

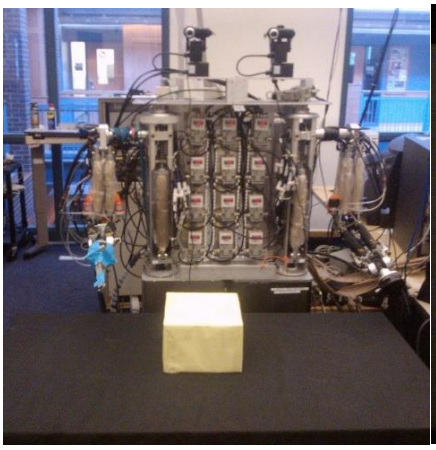
In this experiment, ISAC is asked to push an object which is placed on a table in front of it. In order to complete the task, ISAC needs to generate a behavior sequence, which is composed of the "Reaching" and "Pushing" behaviors. The generated "Reaching" and "Pushing" behavior should have the same feature of the demonstration which minimizes the distance between the end-effector and the target object.

This experiment is to validate the following specifications

1. Parse speech commands for ISAC to determine the requirements of given tasks.
2. Observe the demonstration from a human teacher, record the recorded motion trajectory of the hand of the human teachers.
3. Generalize demonstrated behaviors and store them in the LTM
4. Construct behavior graphs for ISAC to use
5. Generate behavior sequences to complete task

The target object used in the experiments carried out on ISAC is a yellow box with the size: 18 cm (length), 18 cm (width), and 12 cm (height).





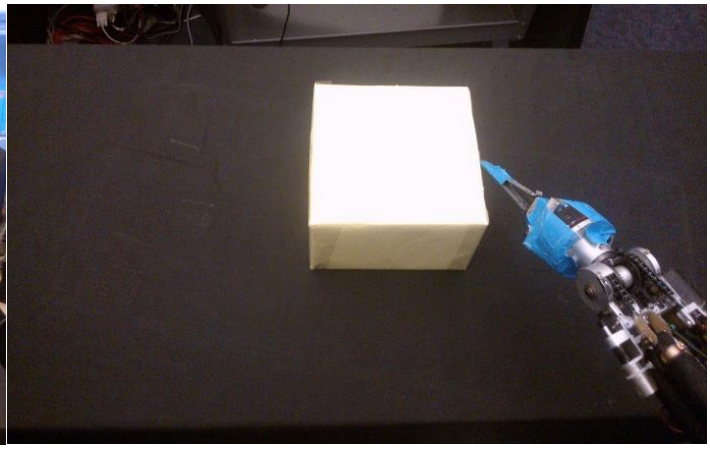


Figure 49 Environmental Setup of Experiment 1

The box is placed on a table in front of ISAC. ISAC is asked to push the box to its left or right. The box is placed at 4 different locations for ISAC to push without grasping. Figure 49 displays the environmental setup. This experiment is to validate that our system

can enable ISAC to learn behaviors from human demonstrations and to generate behavior sequences to complete tasks, so all the objects are placed within the working space of ISAC.

ISAC first checks the stored information in the LTM to find whether it has learned the required behaviors to learn. In order to demonstrate that ISAC could learn and generalize behaviors, initially, the behavior library is blank. Thus, ISAC requires demonstration from human teachers to learn “Reaching” and “Pushing”.

--*GUI Interface*

Figure 50 display the control interface implemented on computer Octavia, which is used to control the voltage of the regulators.

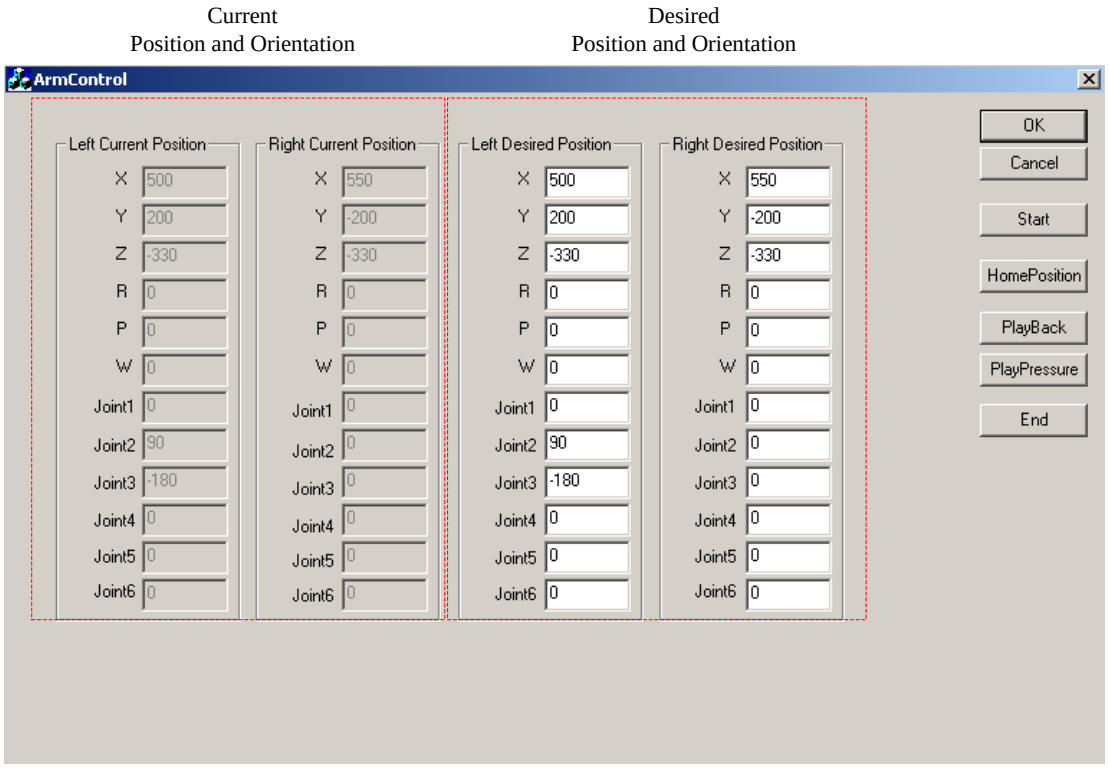


Figure 50 ISAC Arm Control Interface

This program can receive the via points on a desired motion trajectory from another computer. The communication protocol is described in Chapter IV. After receiving the via points on the motion trajectory, this program convert it to a sequence of joint angles using inverse kinematics method. Then the joints of ISAC are driven by the air muscles by changing the voltage of the regulars on ISAC. The closed control loop incorporates a PID control method.

The other method of controlling ISAC is to put the desired position and orientation values in the area of desired position and orientation of the dialog shown in Figure 50. By pressing the "Start" button, this program computes the required joint angles using inverse kinematics and sends the control command to the regulators.

The first method is used in Experiment 1.

Figure 51 displays the GUI interface of the IRS and the speech command communication. The stored basic behaviors are displayed on the top of the dialog. In the "System Status" area, it displays the current system status, e.g., parsing speech command, recording motion trajectory, generalization, generating behavior sequences, generated motion trajectories, etc. The learning status is displayed on the left side of the dialog. Constructed behavior graph and the generated behavior sequence are shown at the bottom of the dialog. Given speech commands described in Chapter IV, ISAC records the motion trajectories of the hand of the human teacher using Kinect, generalizes the learned behaviors, and generates motion trajectories which are sent to the computer Octavia to control the arms of ISAC.

The IRS simulation environment is displayed on the right side of the dialog. The kinematics module converts position values of the via points on the motion trajectory to the joint angles. These angles are sent to the displaying module to update the current joint angle of the simulation model of ISAC. Meanwhile, the position of the box and the obstacle are updated using the information received from the perception module.

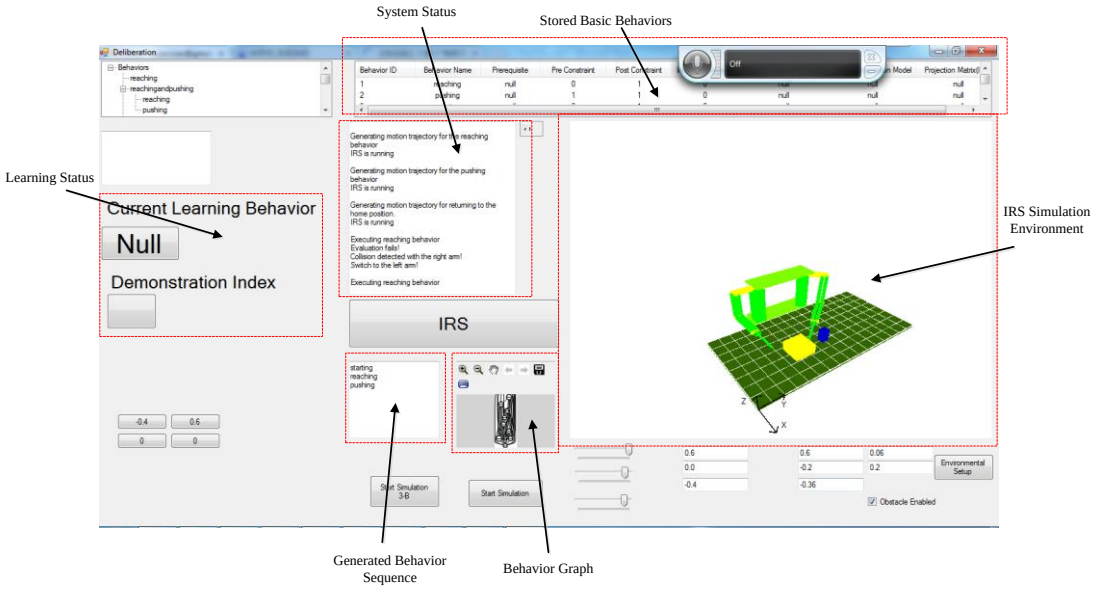


Figure 51 GUI of the IRS and the Speech Command Communication

--*Learning*

Figure 52 displays the experimental setup for the learning stage.

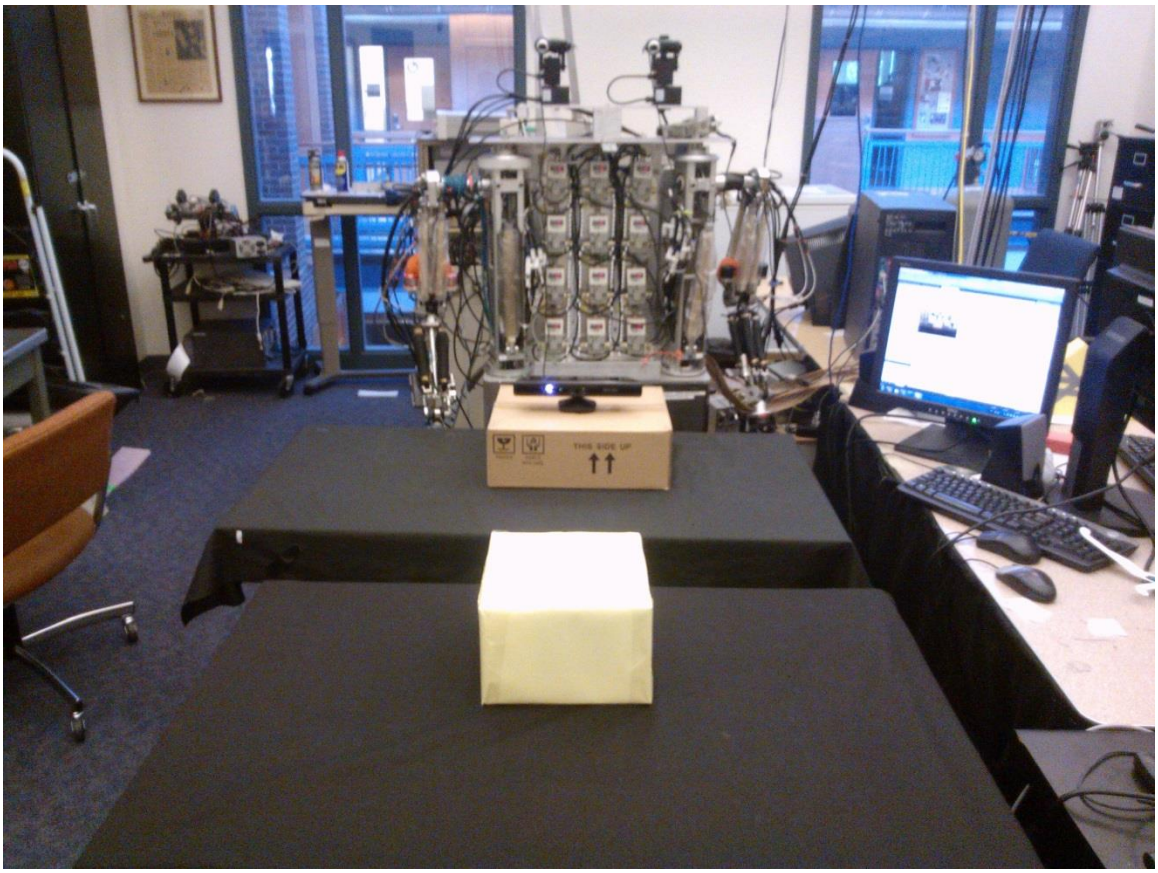


Figure 52 Experimental Setup for the Learning Stage of Experiment 1

In order to teach ISAC to learn the "Reaching" behavior, we put an object at two different locations on the table. A human teacher demonstrates how to reach the object: using the left arm or the right arm, starting from different locations, reaching with two different styles. Thus, there are 16 demonstrations in the learning stage. (8 demonstrations are by using each arm respectively.)

The left upper picture of Figure 53 displays the recorded motion trajectories of "Reaching" behavior demonstrated using the left arm. The starting positions are labeled with stars, and the target positions are labeled with circles. The common features for pre-condition, internal constraints and post-results are predefined in Table 1, Table 2 and

Table 3 in Chapter III. The generalization scores of pre-condition, internal constraints and post-results of the "Reaching" behavior are displayed in the right upper picture, left lower picture, and right lower picture of Figure 53 respectively.

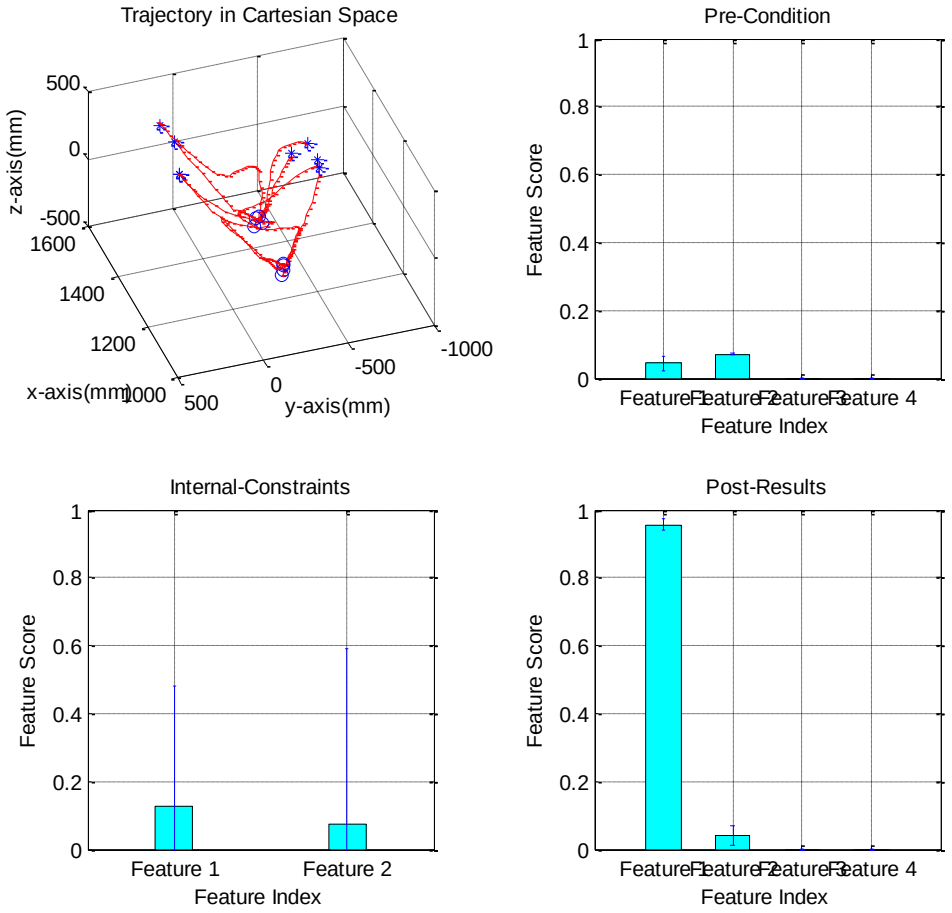


Figure 53 Generalization Results of the "Reaching" Behavior Using the Left Arm

Figure 54 displays the recorded motion trajectories of the "Reaching" behavior demonstrated using the right arm. The starting positions are labeled with stars, and the target positions are labeled with circles.

Although motion trajectories shown in Figure 53 and Figure 54 are different, the generalization results of the “Reaching” behavior using the right arm are the same as the one using the left arm.

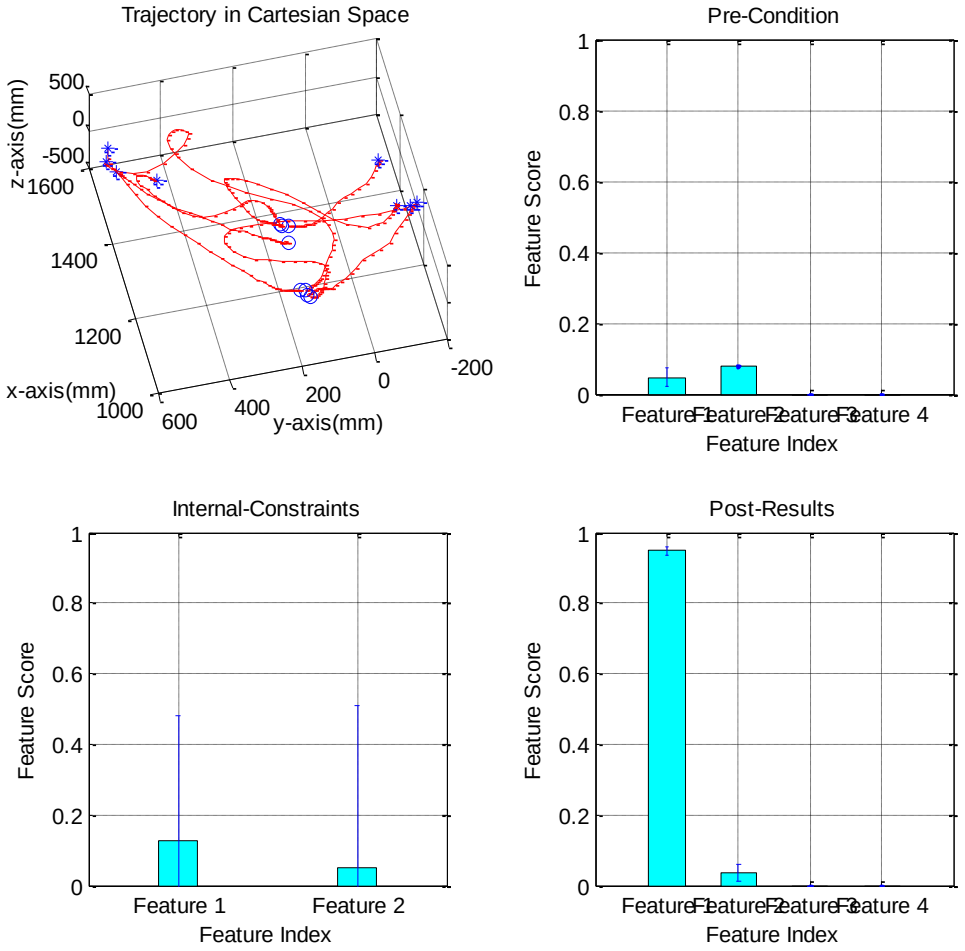


Figure 54 Generalization Results of the “Reaching” Behavior Using the Right Arm

The generalized results reflect that the common feature of the “Reaching” behavior is: minimizing the distance between the hand and the destination position at the end of the motion trajectory. There is no requirement of pre-condition and internal constraint for the “Reaching” behavior.

Figure 55 displays the recorded motion trajectories of the "Pushing left" behavior. The starting positions are labeled with stars, and the target positions are labeled with circles. The generalization scores of pre-condition, internal constraints and post-results of the "Pushing to the left" behavior are displayed in the right upper picture, left lower picture, and right lower picture of Figure 55 respectively.

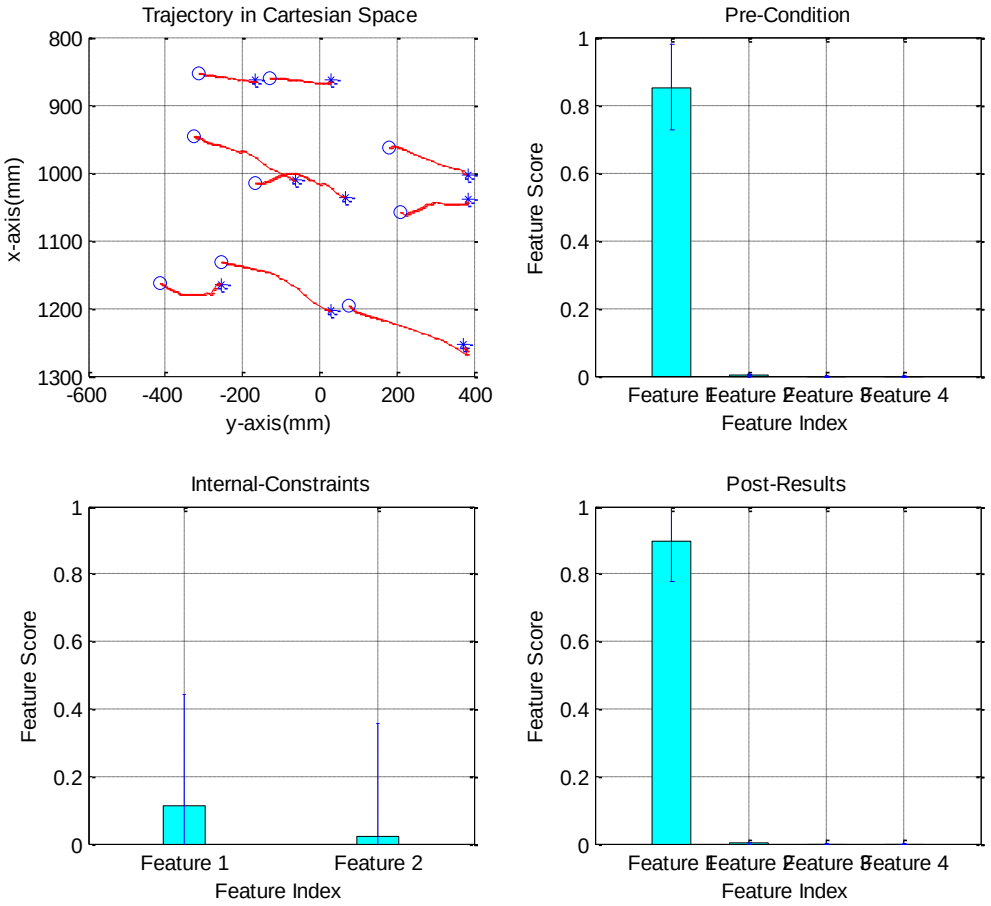


Figure 55 Generalization Results of the "Pushing Left" Behavior

Figure 56 displays the recorded motion trajectories of the "Pushing right" behavior. The starting positions are labeled with stars, and the target positions are labeled with circles. The generalization scores of pre-condition, internal constraints and post-

results of the “Pushing to the right” behavior are displayed in the right upper picture, left lower picture, and right lower picture of Figure 56 respectively.

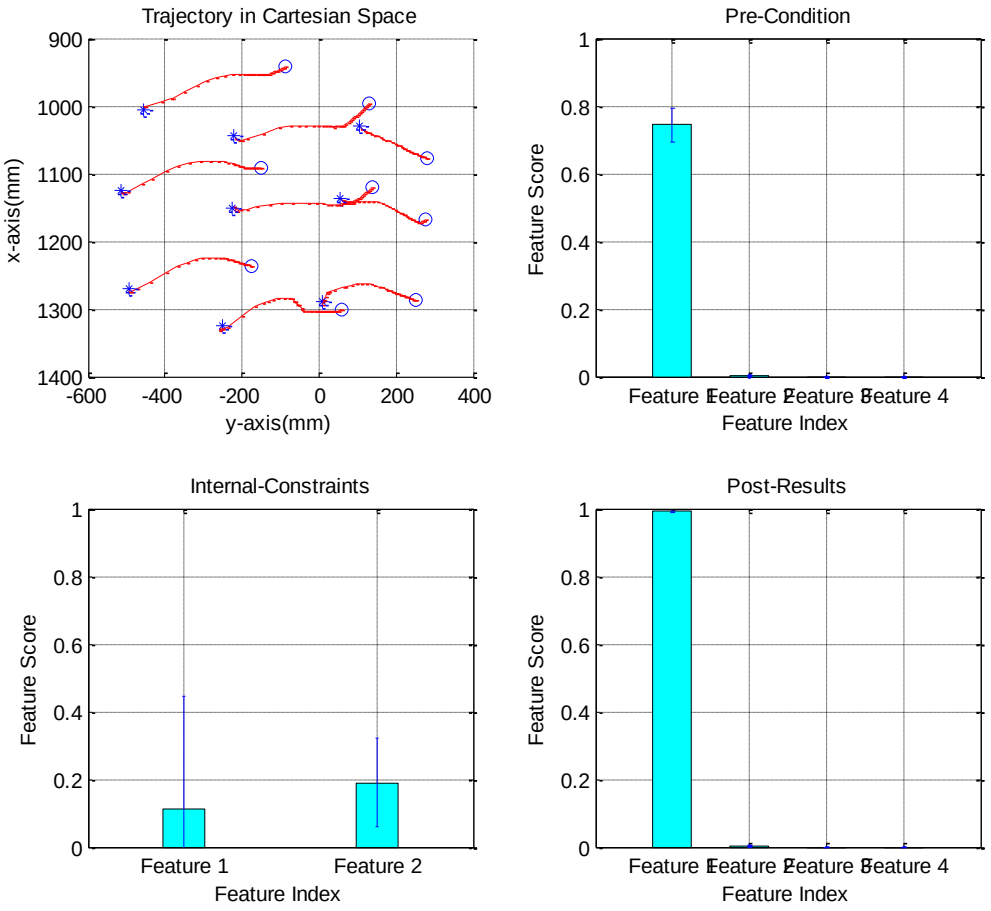


Figure 56 Generalization Results of the “Pushing Right” Behavior

The generalized results reflect that the common feature of the “Pushing Left/Right” behavior is: 1. minimizing the distance between the hand and the target object at the beginning of the motion trajectory; 2. minimizing the distance between the hand and the destination position at the end of the motion trajectory. There is no requirement for internal constraint for the “Pushing to the left/right” behavior.

--*Generation*

Given a command: “Push the box to the left”, ISAC finds the required behavior is “Pushing left” and the parameters are “the box” and the “to the left”. ISAC then generates a behavior sequence by searching the behavior graph to find a shortest path from “Starting” to “Pushing left”. This behavior is composed of “Starting”, “Reaching”, and “Pushing left”.

--*Experimental Results of Experiment 1*

Figure 57 displays the generated motion trajectories in the Cartesian space for pushing the box to the left at 4 different locations labeled by circles. 2nd-Order attractor is used to generate motion trajectories for “Reaching”, and “Pushing left/right”.

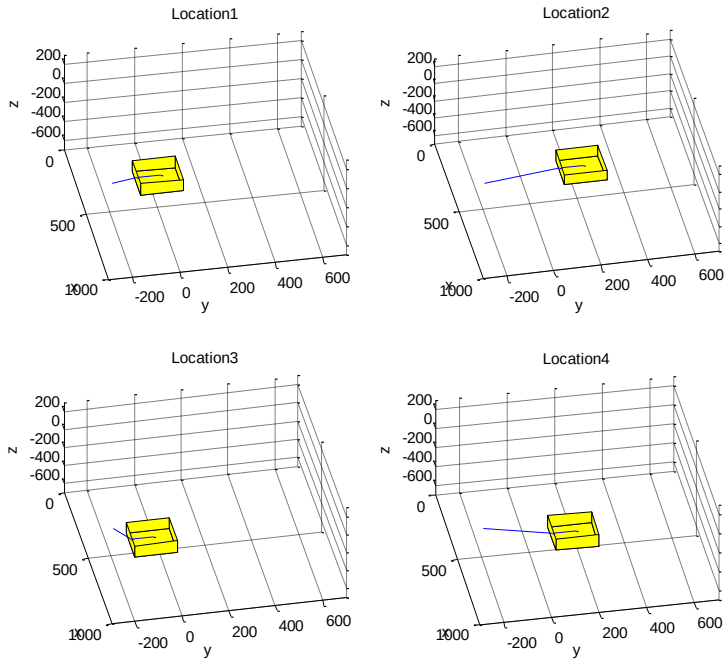


Figure 57 Generated Motion Trajectories for Experiment 1A (Pushing to the Left)

In Figure 58, ISAC pushes the object placed at different locations to the left.

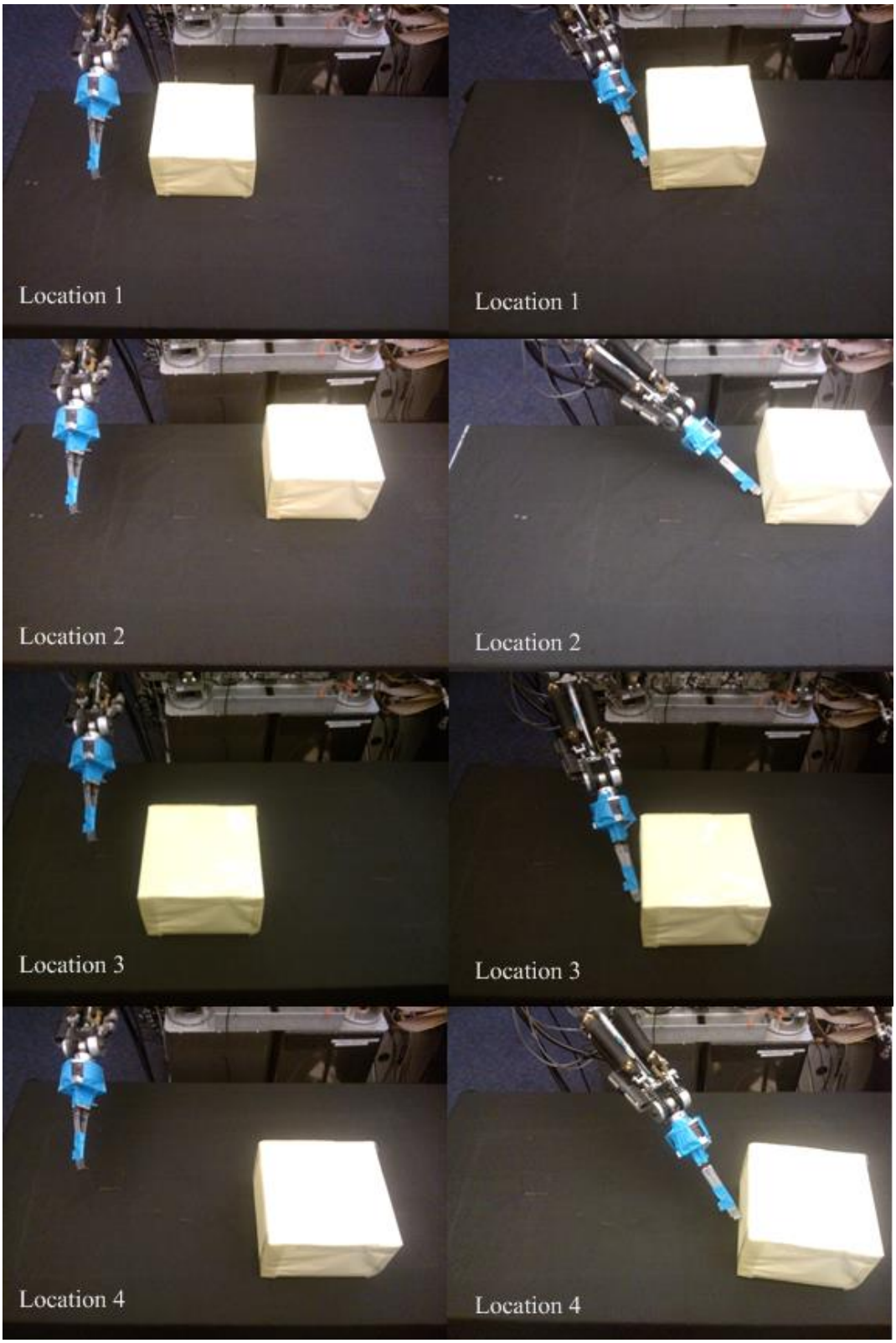


Figure 58 Experimental Results of Experiment I (Pushing to the Left)

When the box is placed at location 1, 2, and 3, ISAC can push it to the left. However, when the box is placed at location 4, ISAC can only push it to the left in a small distance. The reason is that location 4 is close to the edge of the working space of ISAC.

Figure 59 displays the generated motion trajectories in the Cartesian space for pushing the box to the right at 4 different locations labeled by circles.

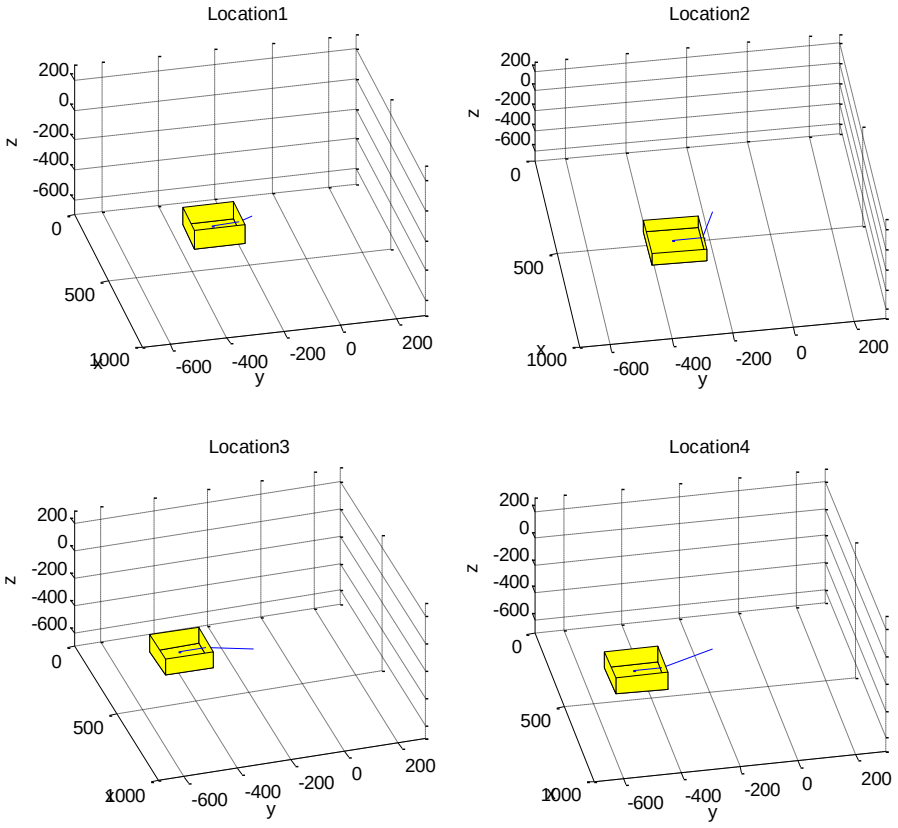


Figure 59 Generated Motion Trajectories for Experiment I (Pushing to the Right)

In Figure 60, ISAC pushes the object placed at different locations to the left.

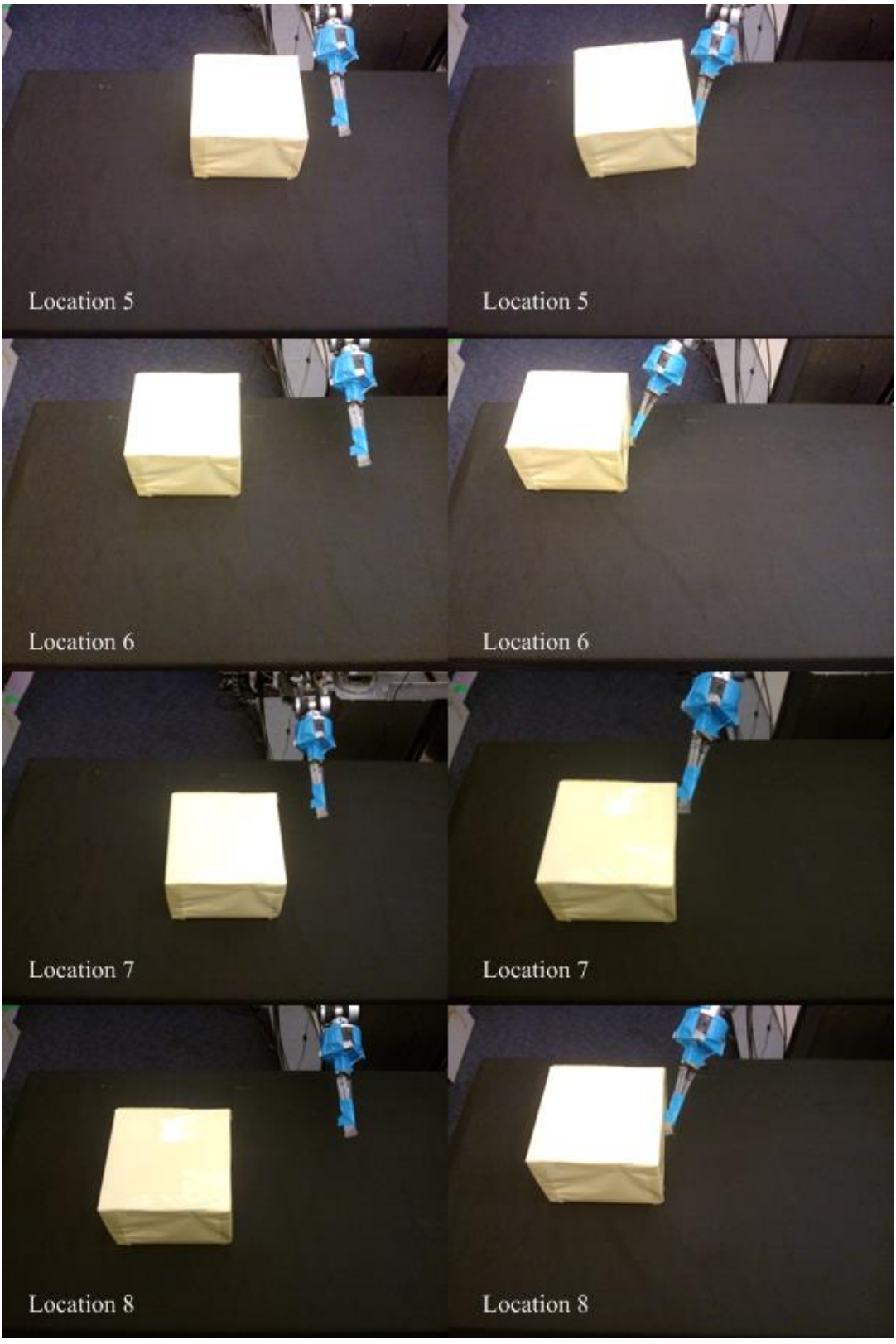


Figure 60 Experimental Results of Experiment I (Pushing to the Right)

From the experimental results, ISAC pushes the box to the right when it is placed at four different locations.

--*Discussion*

According the experiment results, ISAC pushes the object to left, which is placed at 4 different locations and pushes the object to the right when it is placed at 4 different locatioins. The experiment results demonstrate that ISAC can use this system to parse the speech command, check the LTM, observe demonstrations, generalize behaviors, generate behavior sequences, and generate motion trajectories to complete tasks.

Experiment 1B: Supervised Pushing by Physical Coaching

--*Objective*

The objective of this experiment is to investigate how the system learns behaviors from physical coaching.

--*Experimental Setup*

In this experiment, ISAC's right arm is physically moved to the pushing point on an object placed on a table. A human teacher demonstrates then how to push the object on the table by manually moving the right arm of ISAC. This is called physical coaching or physical human-robot interaction [Lee et al., 2011], which is different from the demonstration method in Experiment 1A. During coaching, each successful and failed pushing is recorded. This is similar to reinforcement learning [Sutton and Barto, 1998]. Then ISAC generalizes the demonstrations and uses the most successful point to push the object.

This experiment is to validate the following specifications:

1. Record success or failure of the locations of the pushing point which is demonstrated by physical coaching by a human teacher.
2. Record the most successful point of pushing and store the corresponding behavior in the LTM
3. Generate the desired motion trajectories for ISAC

Figure 61 displays the experimental setup similar to Experiment 1A. However, in this experiment, ISAC learns the best pushing point on the object.

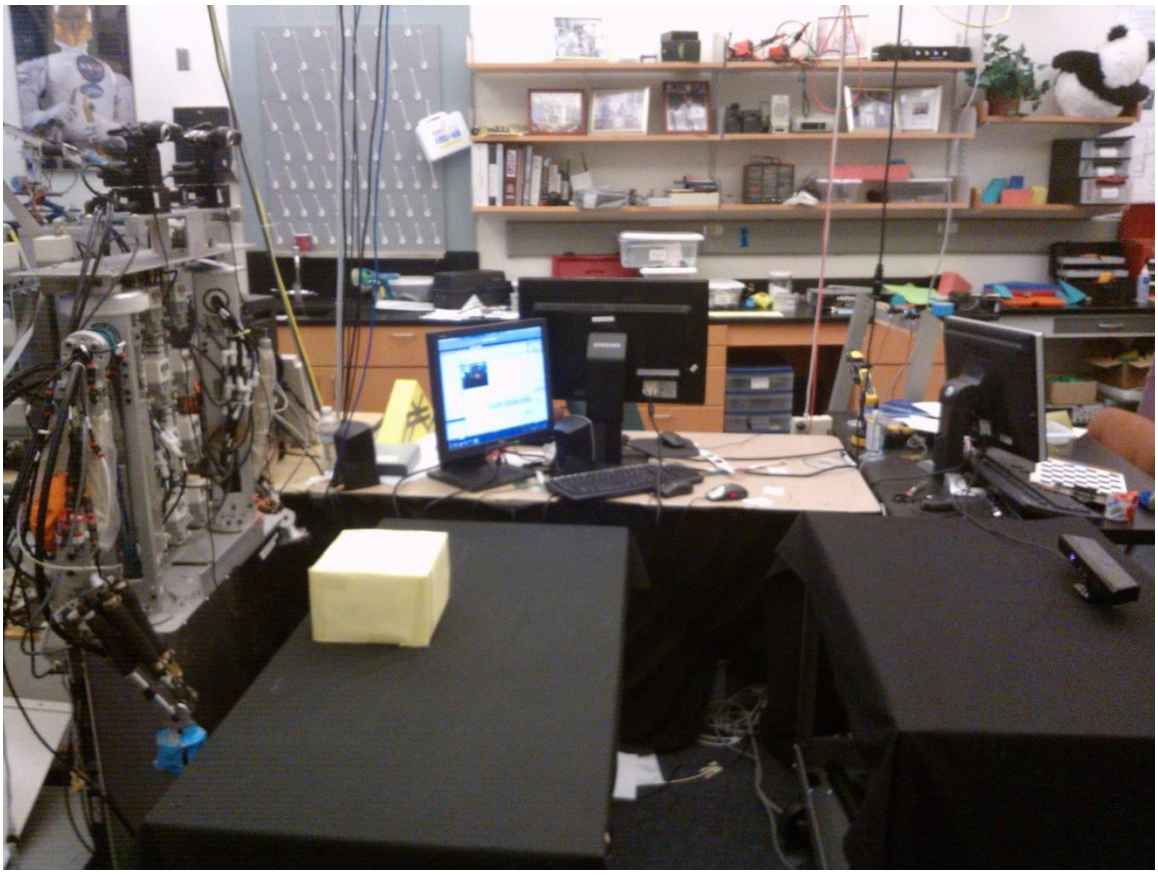


Figure 61 Experimental Setup of the Generation Stage in Experiment 1B

Figure 62 displays the experimental setup of the learning stage. A human teacher physically moves the arm to push the object at different points on the side of the object

and tells ISAC whether each push was successful or not. In this experiment, the height of the box is larger than the length and the width.

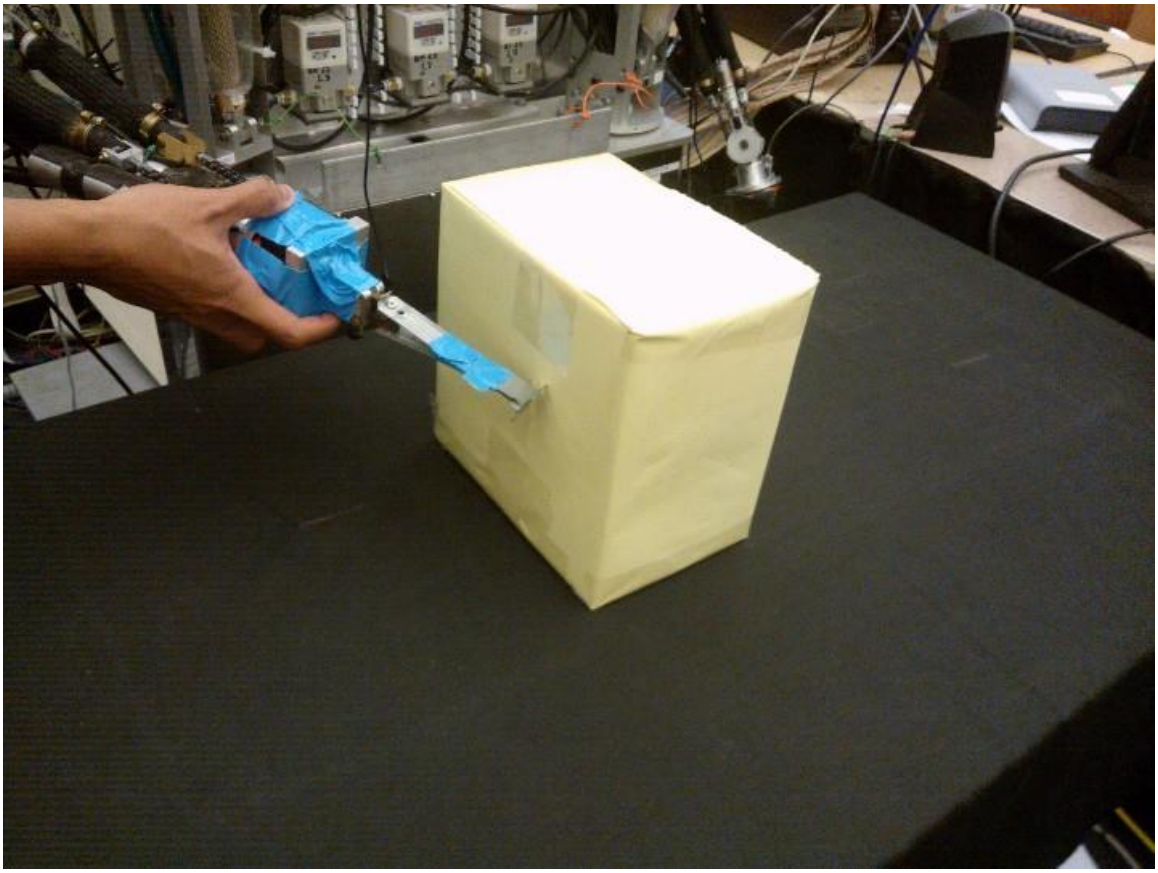


Figure 62 Experimental Setup of the Learning Stage in Experiment 1B

Learning

In Experiment 1B, the center of the object is placed at $\{560, 100, -460\}$. Figure 63 displays the pushing points on the side of the object during the demonstrations. The points labeled with circles are those ISAC can use to push the object, and the points labeled with stars are those ISAC cannot use to push the object.

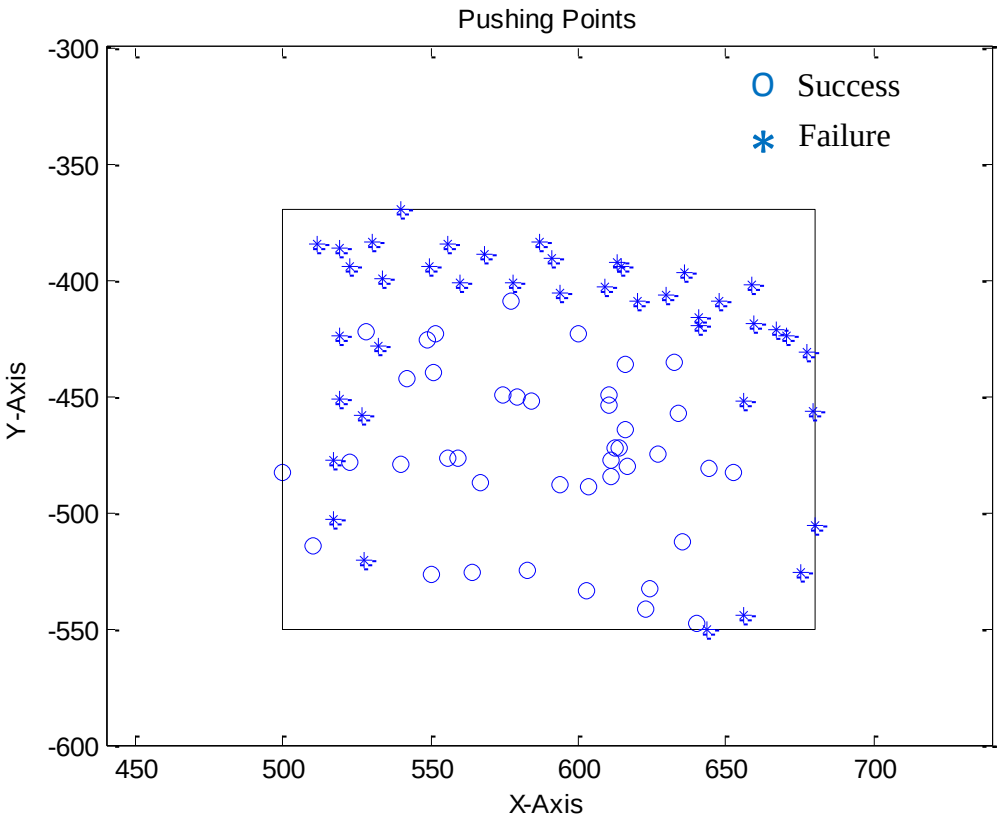


Figure 63 Pushing Points of Experiment 1B

In Figure 63, the points (o), which ISAC should use to push the object, are in the middle and at the bottom, and the points (*), which ISAC should not use to push the object, are at the top and on the sides. We used a Gaussian model to describe the group of the points that ISAC can use to push the object as shown in Figure 64.

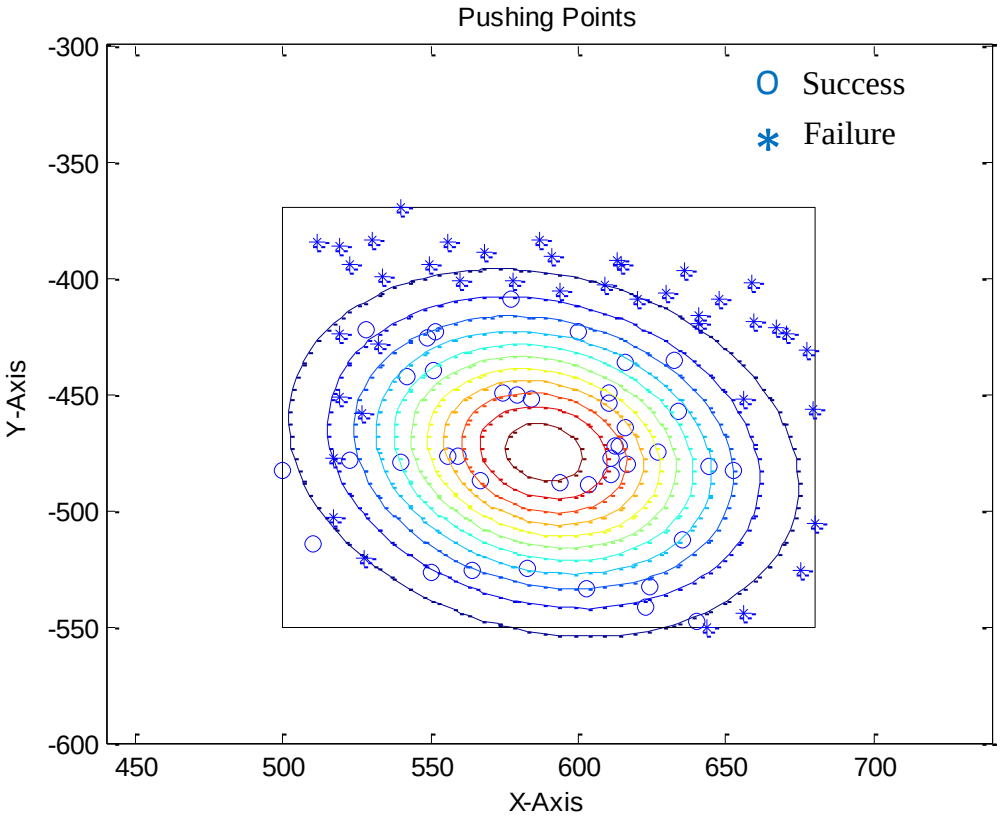


Figure 64 The Gaussian Model of the pushing points in Experiment 1B

The parameters of the Gaussian model shown in Figure 64 are:

$$\mu = \{588.27, -475.20\}$$

$$\Sigma = \begin{bmatrix} 1510.8 & -243.3232 \\ -243.3232 & 1264.8 \end{bmatrix}$$

where μ is the mean and Σ is the covariance matrix of this Gaussian Model. The meaning of this Gaussian model is that ISAC needs to choose points around $\{588.27{,}40, -475.20\}$ as the pushing point to increase the probability of pushing the object to the required location. If the center of the box is $\{x_c, y_c, z_c\}$, the chosen pushing point is $\{x_c - 1.73, y_c \pm 60, z_c - 15.20\}$

Experimental Results

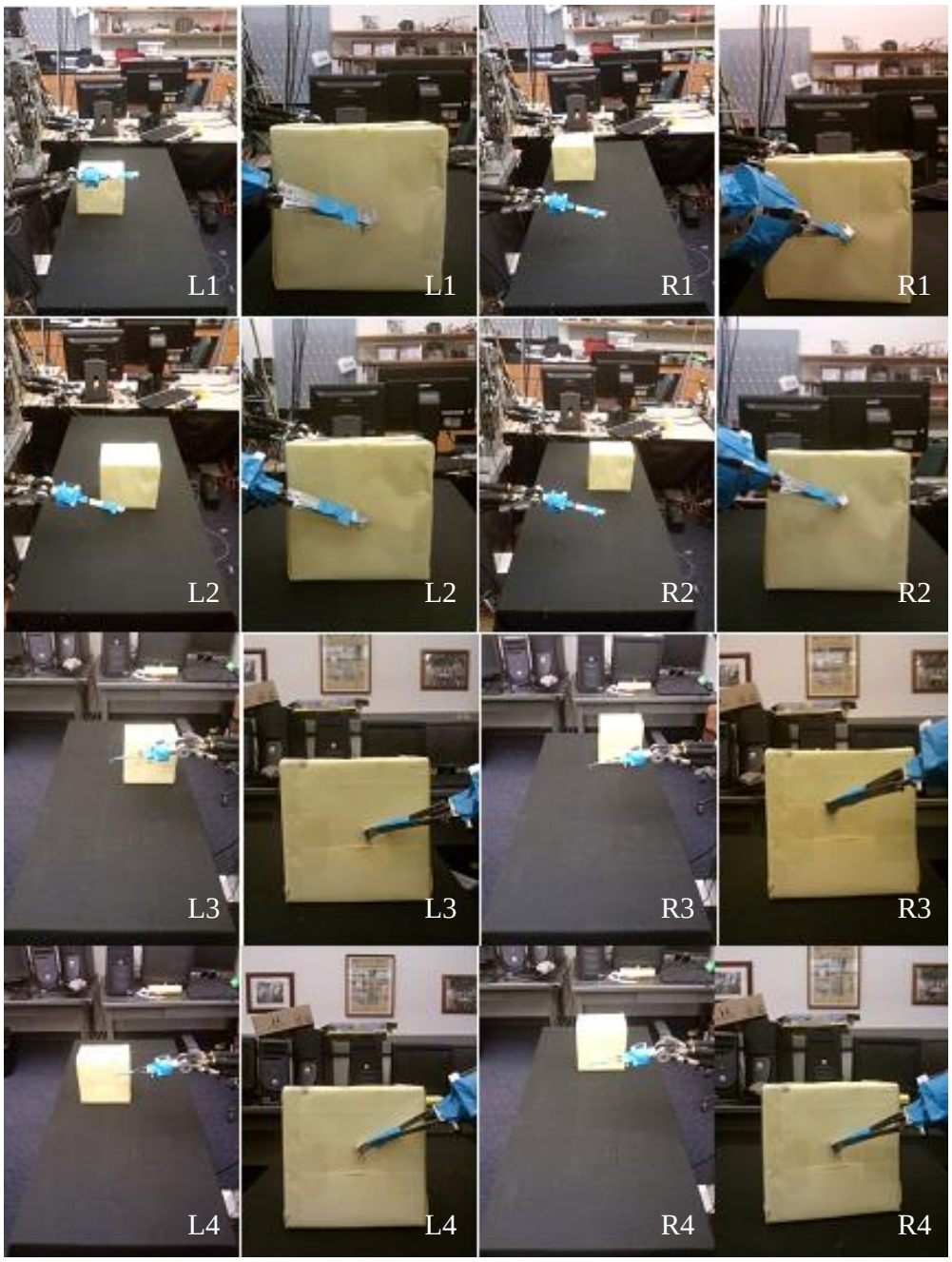


Figure 65 Experimental Results of Experiment 1B

In Figure 65, ISAC reaches the pushing points which are described using Gaussian model learned from the physical coaching of the human teacher.

--*Discussion*

In Experiment 1B, a human teacher teaches how to find the pushing points on the object using physical coaching method. A Gaussian model is used to describe the learned result and ISAC chose the mean of this model as the pushing point. The quantitative analysis of the generation results of Experiment 1B will be explained in Chapter V.

Experiment 1C: Compensator for ISAC Arm Control

--*Objective*

The objective of the experiment is to how the system compensates the error generated by the hardware.

--*Experimental Setup*

In this experiment, ISAC is asked to reach the pushing point on an object which is placed on a table in front of it. ISAC first tries to use the pushing point without using the compensator. The error is measure by computing the distance between the end-effector and the pushing point on the object, which is used as the input of the compensator. Then ISAC uses the stored information of the compensator to change the target position to overcome the error generated by the hardware.

This experiment is to validate the following specifications

1. Store the error in the compensator
2. Use the compensator to overcome the error generated by the hardware.

The target object used in the experiments carried out on ISAC is a yellow box with the size: 18 cm (length), 18 cm (width), and 12 cm (height).

The box is placed on a table in front of ISAC. ISAC is asked to reach the pushing point on the box. The box is placed at 4 different locations. Figure 66 displays the environmental setup. This experiment is to validate that our system can enable ISAC to overcome the error generated by the hardware.

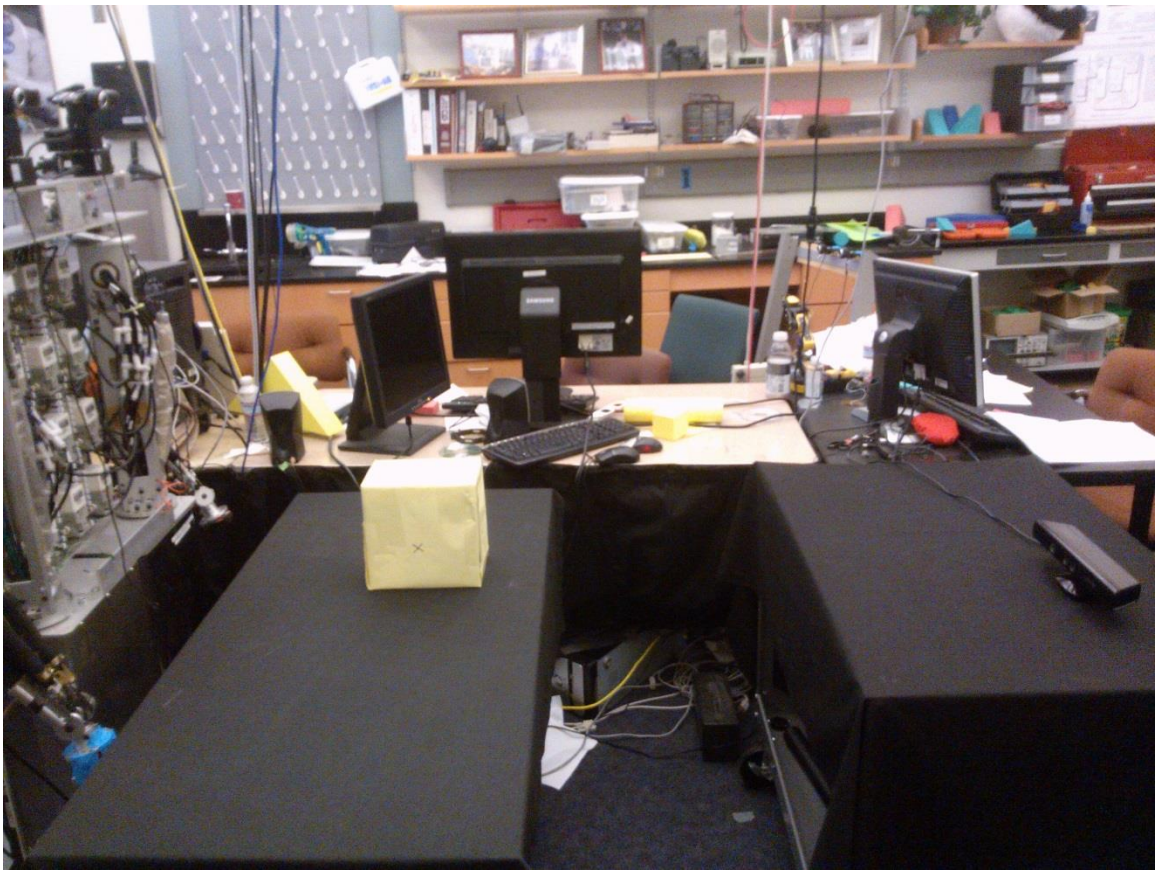


Figure 66 Experimental Setup of Experiment 1C

--*Experimental Results*

Figure 67 displays the experimental results in Experiment 1C.

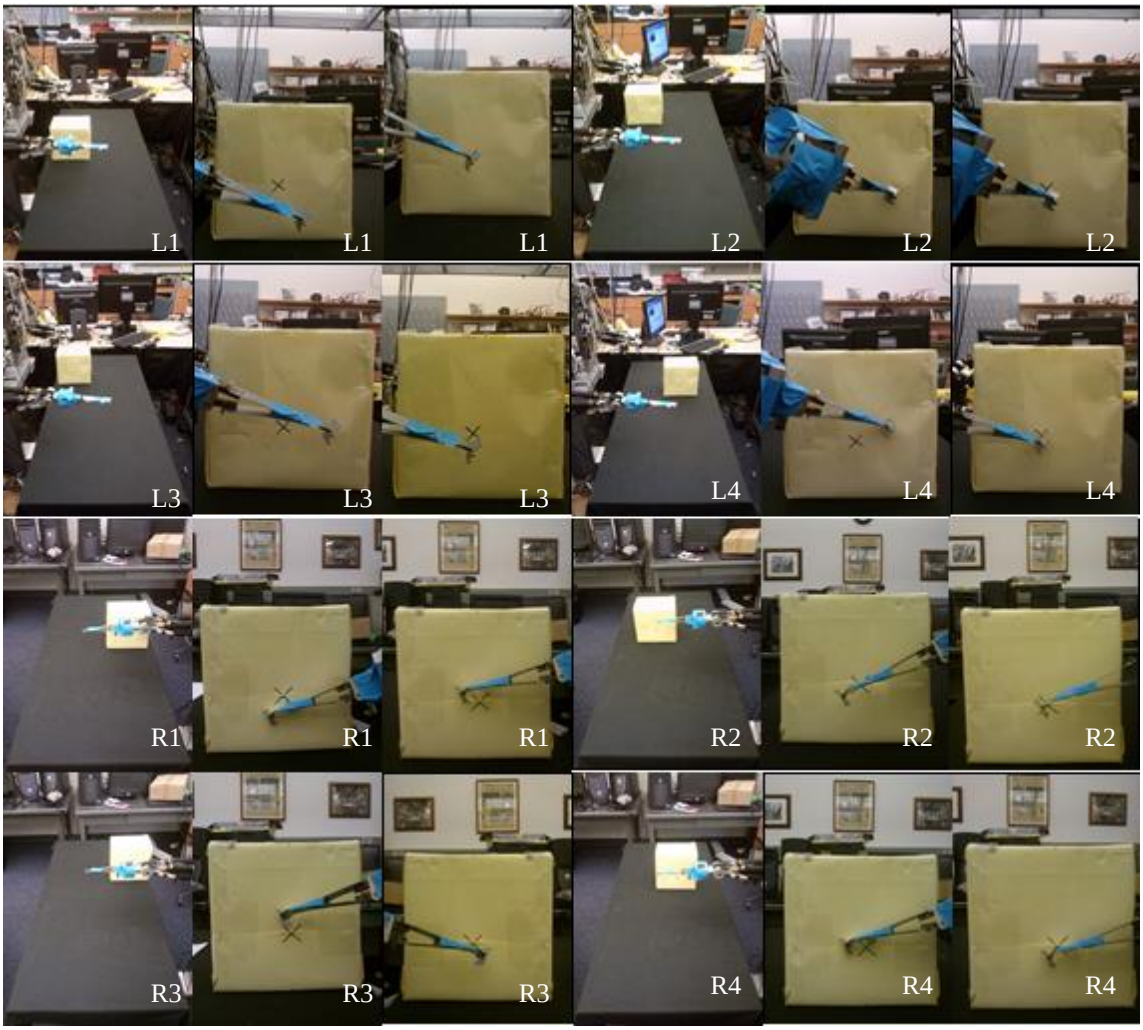


Figure 67 Experimental Results of Experiment 1C

L means that ISAC needs to reach the pushing point on the box to push to its left and R means ISAC needs to reach the pushing point on the box to push to its Right. The box is placed at 4 different locations for pushing left and right. Thus there are 8 experiments in Experiment 1C.

In the figures for each experiment, the left picture displays the location of the box, the middle picture displays the location of the end-effector without using the compensator,

and the right picture displays the location of the end-effector using the compensator. The required pushing point is labeled using "×" on the box.

In Figure 67, the compensator improves the performance of the arm control, and the error becomes smaller. The quantitative analysis of the experimental results of Experiment 1C is explained in Chapter V.

The next experiment is to investigate how ISAC can generate motion trajectories which are similar to demonstrated behaviors. This is popular in current imitation learning research. We want to investigate how this type of traditional imitation learning methods could be integrated within the system in this dissertation.

Experiment 2: Yo-Yo Playing

Objective

The objective of this experiment is to investigate how the system learns newly observed behaviors and generates similar motions.

Simulation Experiment Description

This experiment is to ask ISAC to play Yo-Yo, which is a very common game for children. The Yo-Yo is placed within the working space of the right arm of ISAC and ISAC needs to generate a Yo-Yo Playing behavior sequence, which is composed of the "Reaching", "Grasping", and "Yo-Yo Motion", to complete the task. In order to generate this behavior sequence, ISAC needs to observe and generalize the "Yo-Yo Motion" behavior and add it to the behavior graph. In this experiment, the generated "Yo-Yo Motion" behavior should be similar to the demonstrated "Yo-Yo Motion" behavior. This

is different from generating the “Reaching” and the “pushing” behaviors in Experiment 1, which does not require the similarity between the demonstrated behavior and the generated behavior.

The Yo-Yo playing requires ISAC to generate a behavior sequence which is composed of several behaviors in order to play Yo-Yo. In the learning stage, ISAC checks the LTM and founds it has not learned the “Yo-Yo Motion” behavior. So it asks a human teacher to demonstrate how to play Yo-Yo. The human teacher demonstrates a behavior sequence which is composed of “Reaching”, “Grasping”, and “Yo-Yo Motion”. ISAC checks the LTM and founds that it has already learned the “Reaching” behavior, so it does not need to generalize the “Reaching” behavior again. Because of the limitation of the hardware, ISAC cannot learn the “Grasping” through physical coaching and observe the grasping using Kinect, the “Grasping” behavior is added into the database by the human teacher. Then ISAC needs to learn and generalize “Yo-Yo Motion” behavior in this experiment, and to add them in the LTM.

In the generation stage, based on our designed method, “Yo-Yo Motion” behavior has a pre-condition which requires that a Yo-Yo already be in hand. So ISAC needs to find a behavior which satisfies the pre-condition requirements of the “Yo-Yo Motion” behavior. I.e., ISAC need to find a behavior in the behavior graph, which has a transition edge going to the “Yo-Yo Motion” behavior. By searching the constructed behavior graph, ISAC should find a path from the “Starting” to the “Yo-Yo Motion” behavior to generate a behavior sequence to play Yo-Yo. This behavior sequence is composed of “Reaching”, “Grasping”, and “Yo-Yo Motion”. The requirement of the “Reaching” behavior is to minimize the distance between the end-effector and the target object and

the requirement of the “Yo-Yo Motion” behavior is to generate motion trajectories which are similar to the demonstrations.

This experiment is to evaluate the following major requirements:

1. Learn new behavior: “Yo-Yo Motion” from a demonstrated behavior sequence.
2. Generalize demonstrated “Yo-Yo Motion” behavior, the internal constraint of which is to keep motion dynamics similar to a demonstrated motion;
3. Use behaviors learned in the first experiment (Reaching) and learn necessary additional behavior in this experiment (Yo-Yo Motion);
4. Construct a complex behavior graph from stored information in the LTM;
5. Find a path from the “Starting” behavior to the required “Yo-Yo Motion” behavior and generate behavior sequence;
6. Generate motion trajectories which are similar to the demonstrations of the “Yo-Yo Motion” behavior;
7. Run the simulator.

Learning

The learning stage was implemented in the real environment with ISAC using a Kinect sensor to observe the demonstrations; the generation stage was implemented in a simulation environment.

A Yo-Yo is placed at 5 different locations in front of ISAC, and a human teacher demonstrates how to play the Yo-Yo using the right hand as shown in Figure 68. The demonstration is composed of the “Reaching”, the “Grasping”, and the “Yo-Yo Motion” behaviors.

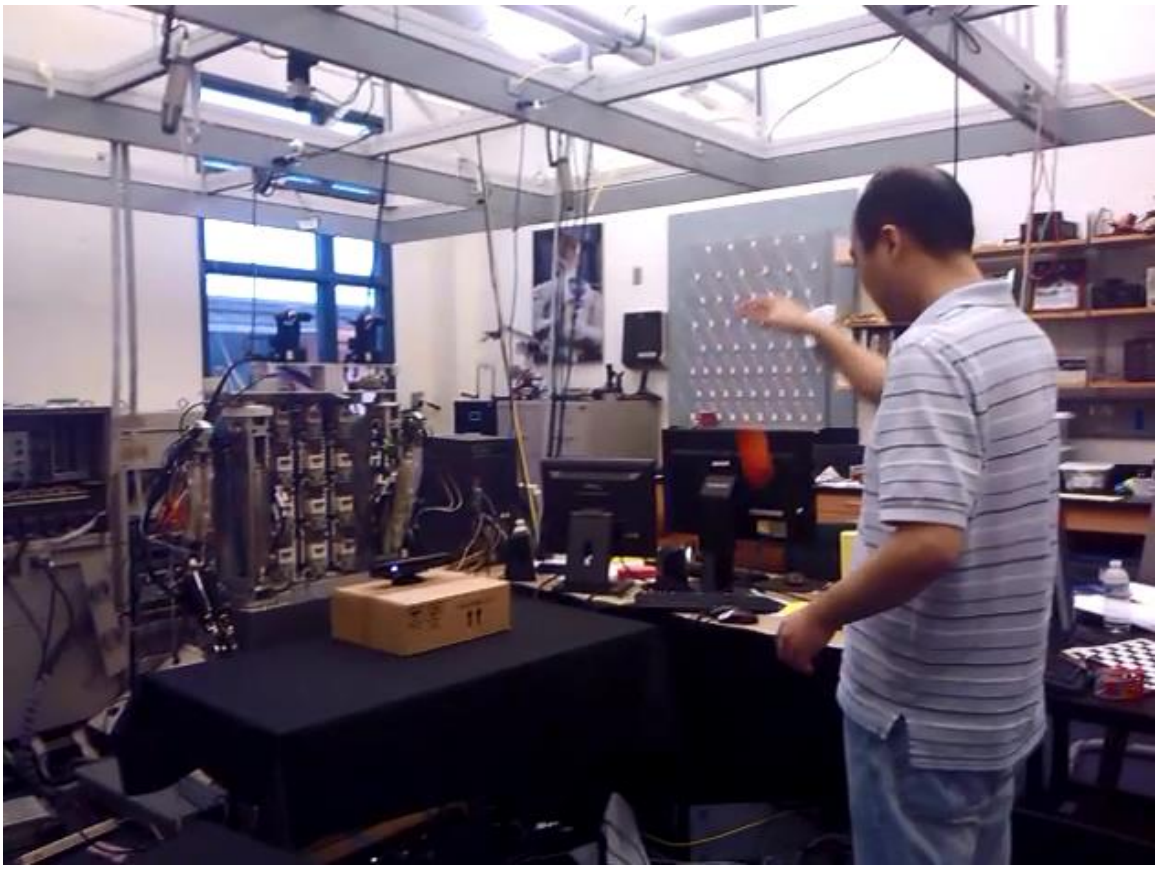


Figure 68 Experimental Setup of the Learning Stage of Experiment 2

Figure 69 displays the recorded motion trajectories of the demonstrations in the Cartesian space. The starting positions are labeled with stars, and the target positions are labeled with circles. These demonstrations are composed the “Reaching”, “Grasping”, and “Yo-Yo Motion” behaviors.

ISAC checks the LTM and finds that it has already learned the “Reaching” behavior. The “Grasping” behavior is manually added into the LTM for ISAC. Then ISAC needs to generalize the “Yo-Yo” motion behavior in this experiment.

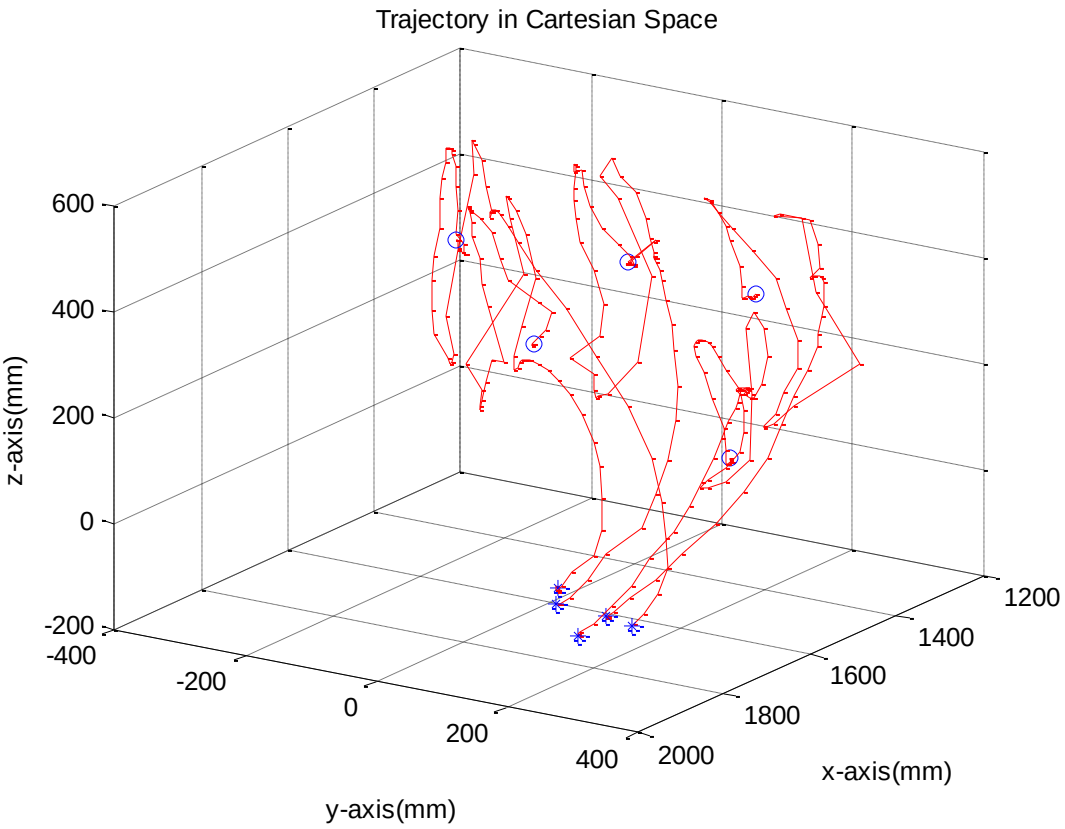


Figure 69 Recorded Motion Trajectory of the "Yo-Yo Playing" Behavior Sequence

Figure 70 displays the motion trajectories on X, Y, and Z axis.

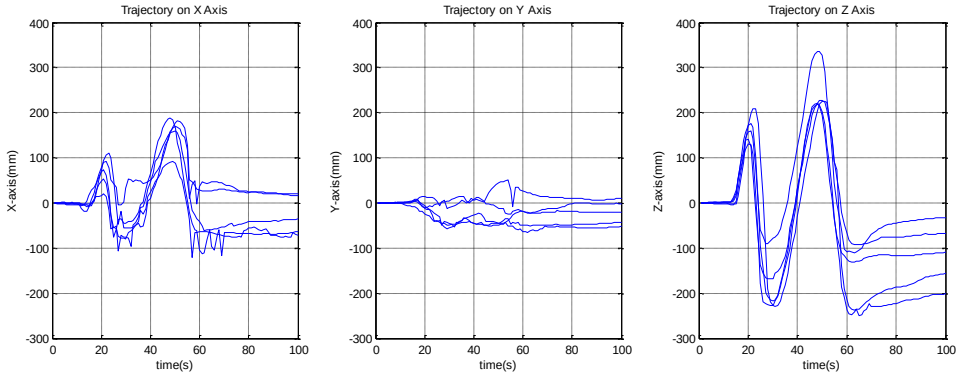


Figure 70 Recorded Motion Trajectories of "Yo-Yo Motion" Behavior

Figure 71 displays the generalization results of "Yo-Yo Motion" behavior.

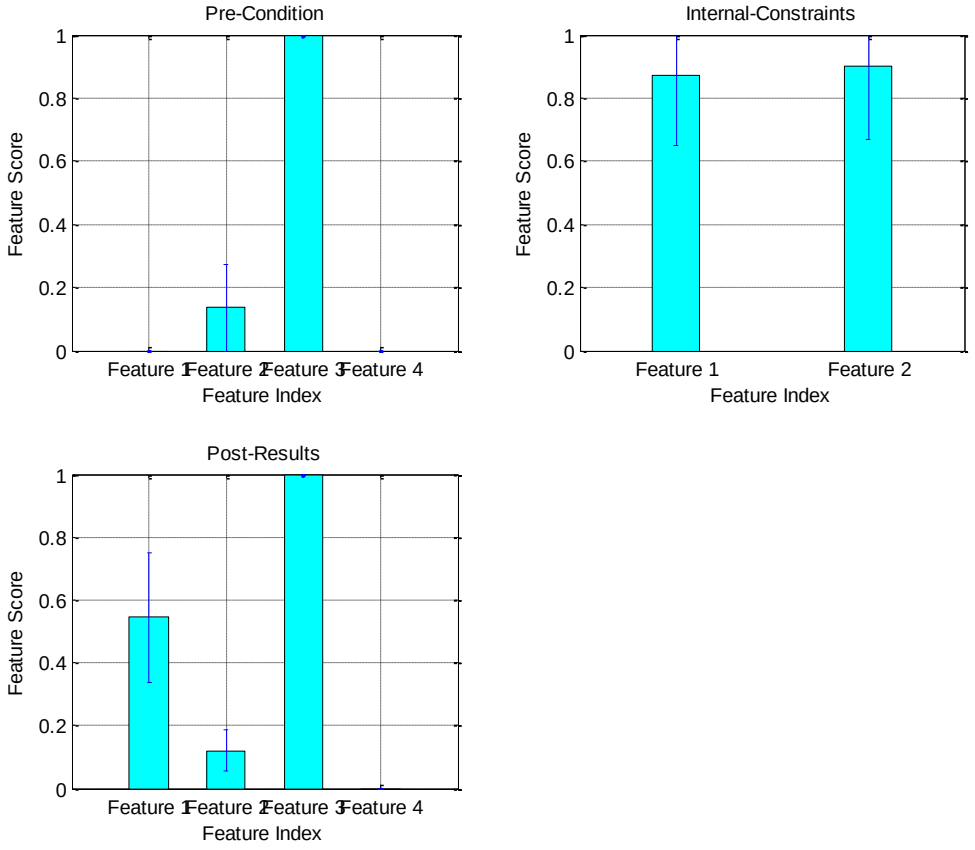


Figure 71 Generalization Results of the "Yo-Yo Motion" Behavior

Based on these results, the pre-condition of the "Yo-Yo Motion" behavior is to minimize the distance between the hand and the Yo-Yo and close the hand, the internal constraint is to keep similar dynamics, and the Post-Result is to close the hand.

Generation

Given a command: "Play Yo-Yo", ISAC finds that the required behavior is the "Yo-Yo Motion" behavior and the parameter is "Yo-Yo". ISAC first constructed a behavior graph as shown in Figure 72 and generated a behavior sequence by searching the behavior graph to find a shortest path from "Starting" to "Yo-Yo Motion".

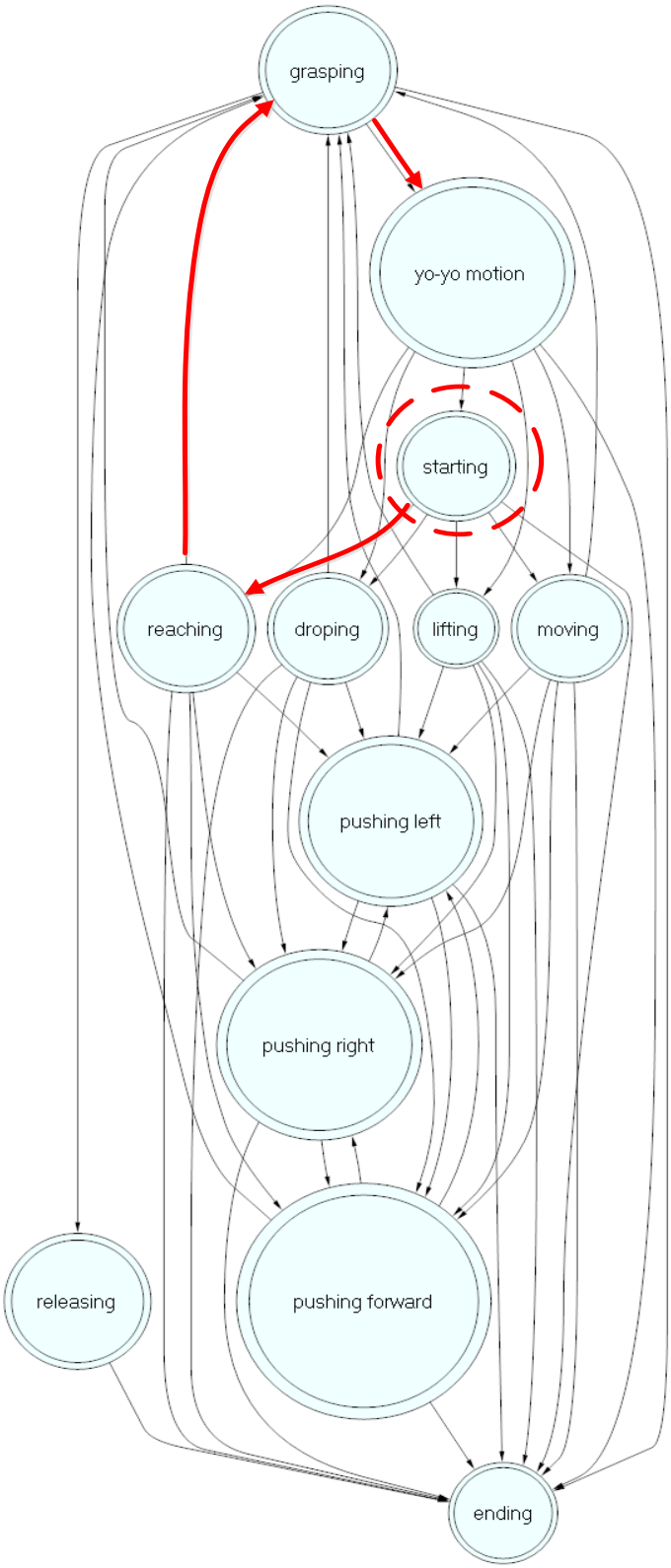


Figure 72 Behavior Sequence Generation for Yo-Yo Playing

This behavior is composed of “Starting”, “Reaching”, “Grasping”, and “Yo-Yo Motion”. 2nd-Order attractor is used to generate motion trajectories for “Reaching”, “Closing End-Effector” is used for “Grasping”, and DMP is used for “Yo-Yo Motion”. The selection of these methods is based on Table 5 in Chapter III.

Simulation Experimental Results

In Figure 73, the Yo-Yo is placed at 3 different locations labeled by circles. Fig.10 displays the generated motion trajectories for ISAC to play Yo-Yo.

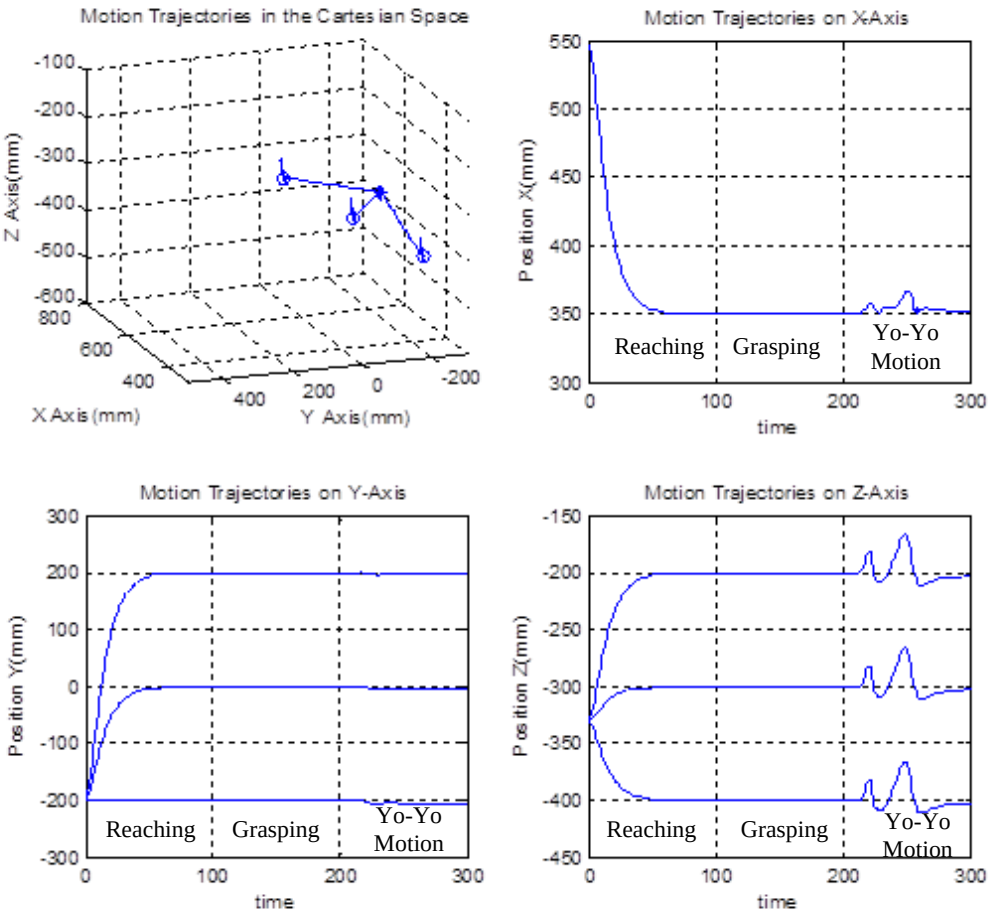


Figure 73 Generated Motion Trajectories for Experiment 2

The first behavior is “Reaching”, which is used by ISAC to minimize the distance between the end-effector and the target-object. It starts from timing step 0 to timing step 100. The second behavior is “Grasping” which is used by ISAC to grasp the Yo-Yo. It starts from timing step 101 to timing step 200. The third behavior in the behavior sequence for playing Yo-Yo is the “Yo-Yo Motion” behavior and the generalization results of the “Yo-Yo Motion” behavior is to keep similar dynamics. It starts from timing step 201 to timing step 300. The generated motion trajectories of the “Yo-Yo Motion” behaviorsare similar to the demonstrated motion trajectories on X, Y, Z-Axis when comparing the right upper picture, the left lower picture, and right lower picture of Figure 70 and Figure 73 (The third behavior starts from timing step 201).

The quantitative evaluation of the similarity of the generated “Yo-Yo Motion” behavior will be explained in Chapter V.

Discussion

In Experiment 2, the demonstrated Yo-Yo Motion has been learned. The results show that the generated motion trajectories are similar to the demonstrations, which satisfies the requirements of traditional imitation learning research.

From Experiment 1, ISAC already learned the “Reaching” behavior and stored it in the LTM. After ISAC checks the LTM, it puts it the learned “Reaching” behavior in the generated behavior sequence to satisfy the requirement of the pre-condition of the “Yo-Yo Motion” behavior.

Using this method, the human teacher does not need to demonstrate all behavior sequences for ISAC. He/she only needs to demonstrate new basic behaviors and ISAC

will add it into the behavior graph and add the transitions among the new basic behaviors and other behaviors by matching the pre-conditions and the post-results. Given a required behavior, ISAC can find a path from the "Starting" to the required behavior to complete the task.

Experiment 3: Cognitive Control

Objective

This objective of this experiment is to investigate how the system switches strategies to complete a task.

Simulation Experiment Description

In Experiments 1 and 2, ISAC generalized demonstrated behaviors, stored newly learned behaviors in the LTM, constructed behavior graph, generated behavior sequences, and generated motion trajectories to complete tasks. An important contribution of this dissertation is to integrate imitation learning with cognitive control for robot dynamically to perform tasks. Using the integrated system, ISAC should adaptively switch strategies to achieve a given task goal. In Experiment 3, the target object is placed in the environment at different locations, and ISAC is asked to push it using either of its arms. Using the decision making mechanism described in Chapter III, ISAC can adaptively switch strategies to push the target object to the right using one of its arms. If it considers it is impossible to complete this task, it will ask the human teacher to demonstrate it differently or tell why it cannot be done. In Experiment 3A, there is no obstacle in the environment; in Experiment 3B, an obstacle is placed on the table.

Experiment 3A: Integrated System without Obstacle

--*Objective*

This experiment is to evaluate the integrated system of behavior-based cognitive control when there is no obstacle. Since imitation learning has been described in detail earlier, we assume that ISAC already learned all needed behaviors.

--*Simulation Setup*

Figure 74 displays the simulation experimental setup of Experiment 3A. The target object, a yellow box, is placed in the environment at 10 locations, and ISAC is asked to push it to its right. The size of the box is: 18 cm (length), 18cm (width), and 12 cm (height). The coordinates of the 10 locations on the table are: Location 1: $\{40, -40, -46\}$, Location 2: $\{40,0, -46\}$, Location 3: $\{40,25, -46\}$, Location 4: $\{40,50, -46\}$, Location 5: $\{40,90, -46\}$, Location 6: $\{70, -40, -46\}$, Location 7: $\{70,0, -46\}$, Location 8: $\{70,25, -46\}$, Location 9: $\{70,50, -46\}$, Location 10: $\{70,90, -46\}$. (-46 is the Z coordinates of the surface of the table) The units of all the coordinates are centimeters.

The speech command is: "Push the box to the right". ISAC parses the speech command and find the required behavior is "Pushing right", the parameters are "the box" and "to the right". It searches the behavior graph and generates a behavior sequence: $\{Starting, Reaching, Pushing\ Right\}$.

Location 1, 6, 7, and 8 are within the working space of the right arm of ISAC. Then ISAC determines to push the box to its left using its right arm. Location 2, 3, 4 and 9 are within the working space of the left arm of ISAC. ISAC firstly tries to push the box to its left using its left arm. Using the evaluation results from the IRS, it determines that it

cannot push the box to its right using its right arm. Thus, it chooses to transfer the generated behavior sequences to its left arm and tries to push the cube to its right. The evaluation results from the IRS shows that it can push the cube to the left using the left arm. Location 5 and 10 are out of the working space of ISAC. After trying to use the left arm and the right arm, ISAC displays that the box is placed out of its working space and it cannot complete that.

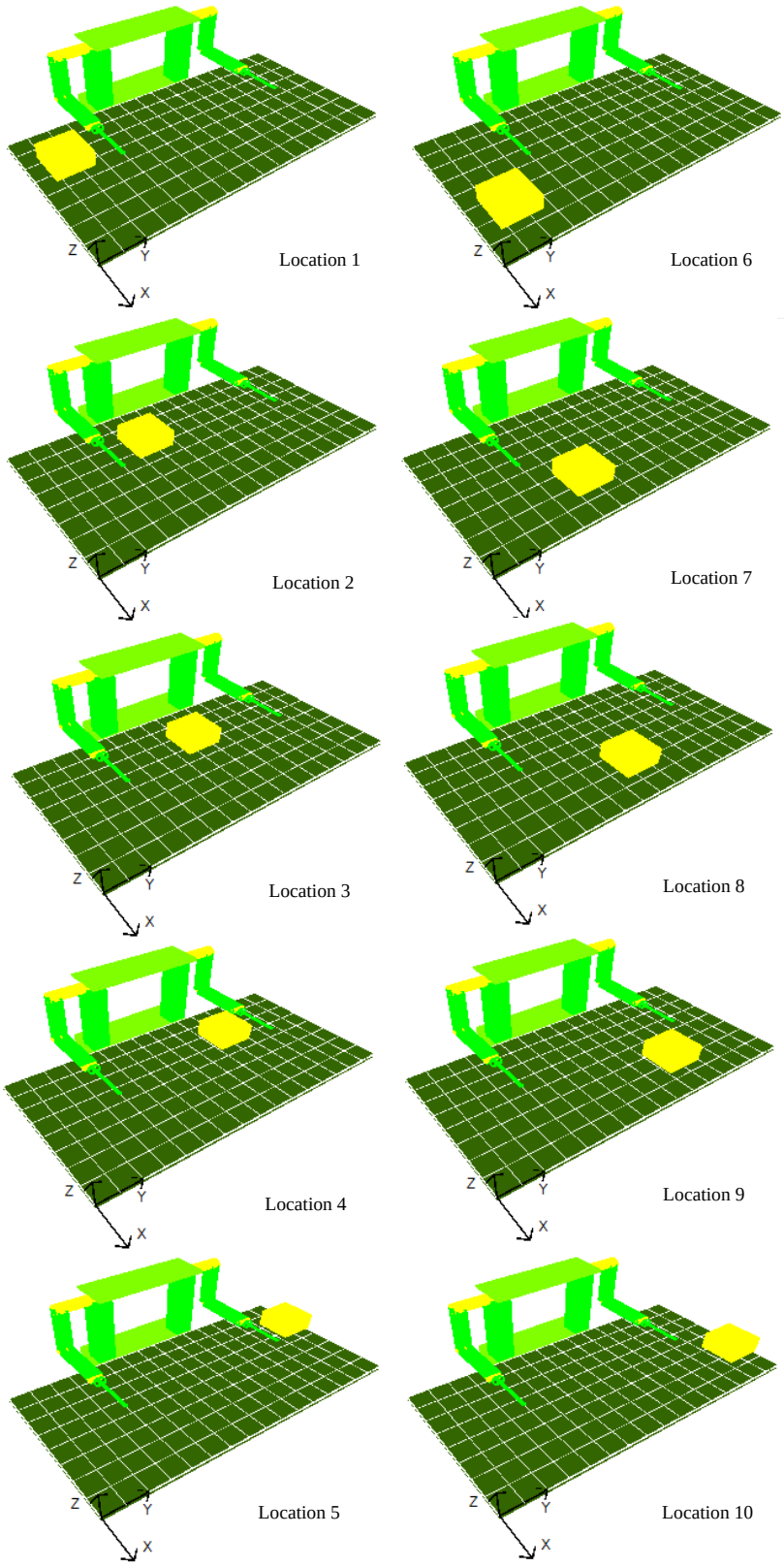


Figure 74 Simulation Setup for Experiment 3A

In Figure 75, the box is placed at Location 1. ISAC pushes the box to the right using its right arm.

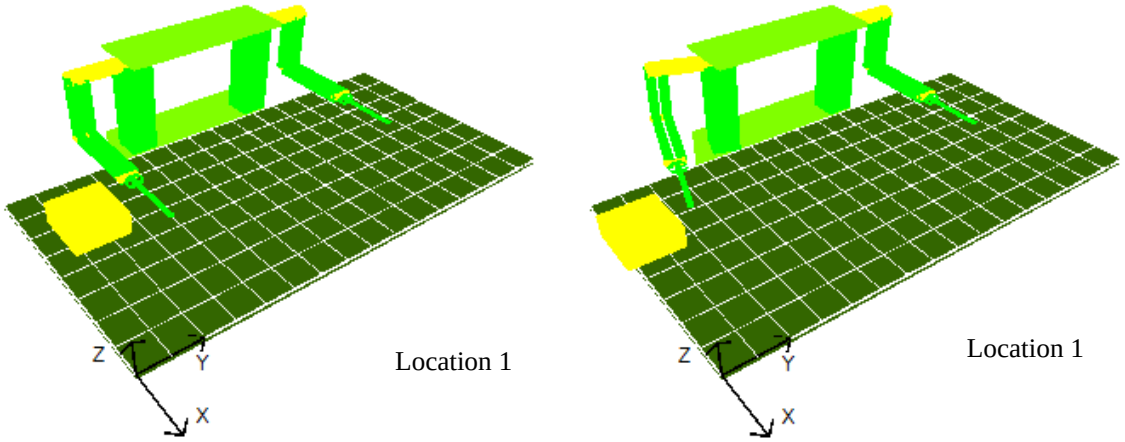


Figure 75 Simulation Results of Experiment 3A-1

Figure 76 displays the generated motion trajectory of pushing the box at location 1 to the right using the right arm.

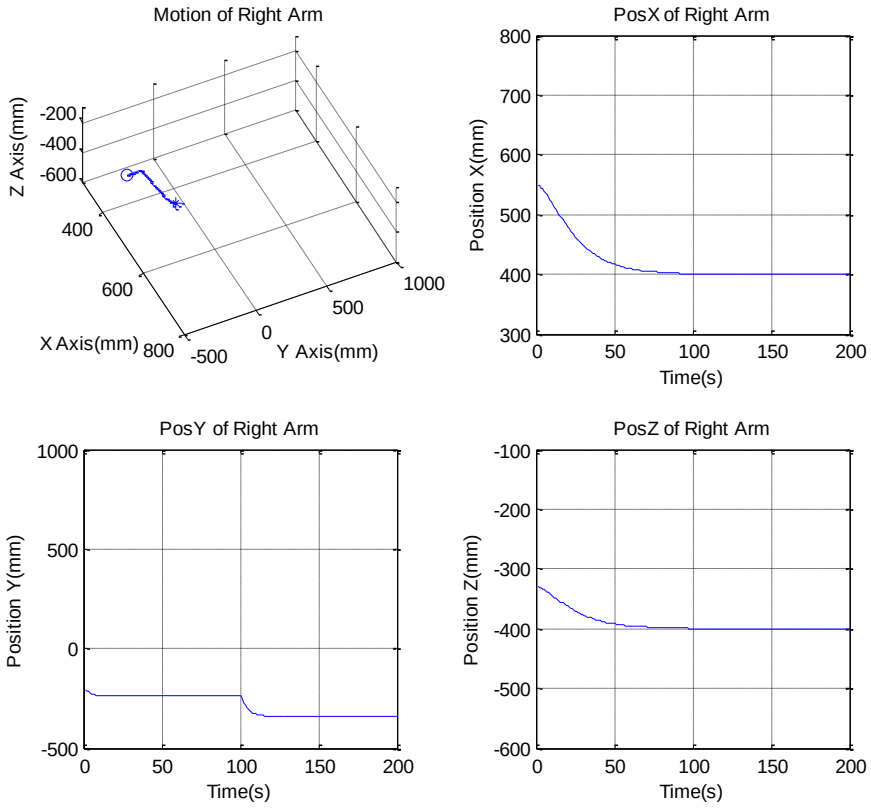


Figure 76 Generated Motion Trajectories for the Right Arm (Experiment 3A-1)

In Figure 77, the object is placed at Location 2. ISAC first tries to push the object using its right arm and detected the collision with the object which is labeled using a black circle. Then ISAC switches the generated sequence to the left arm and pushes the box to the right.

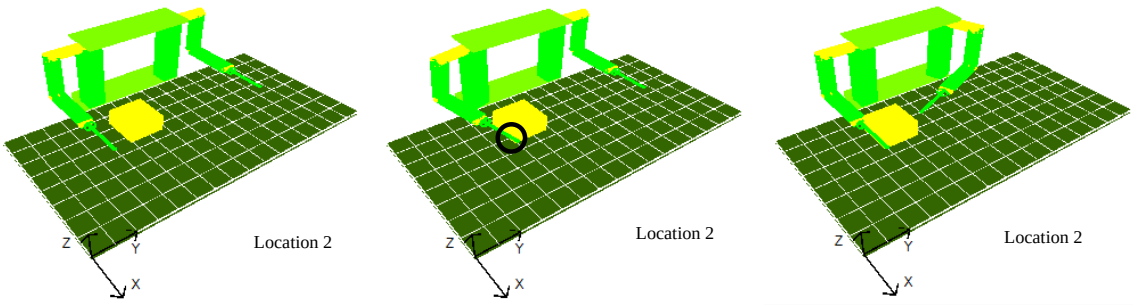


Figure 77 Simulation Results of Experiment 3A-2

Figure 78 displays the generated motion trajectory of pushing the box at location 2 to the right using the right arm of ISAC.

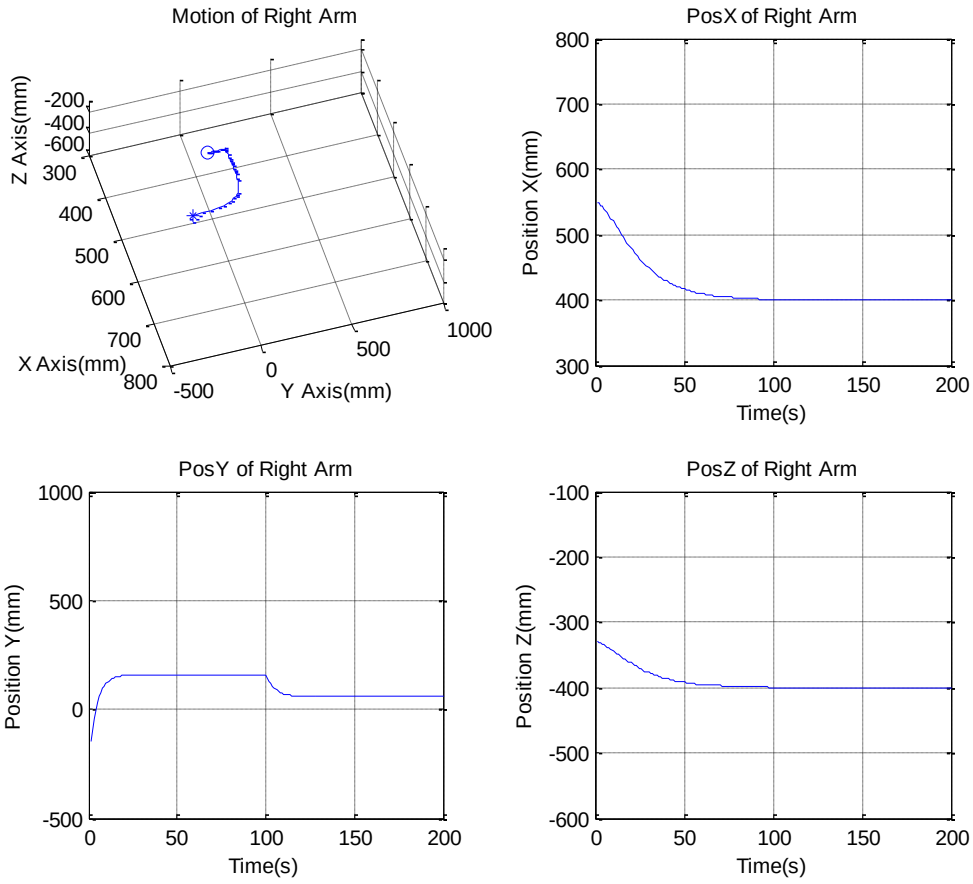


Figure 78 Generated Motion Trajectories for the Right Arm (Experiment 3A-2)

Figure 79 displays the generated motion trajectories for the left arm.

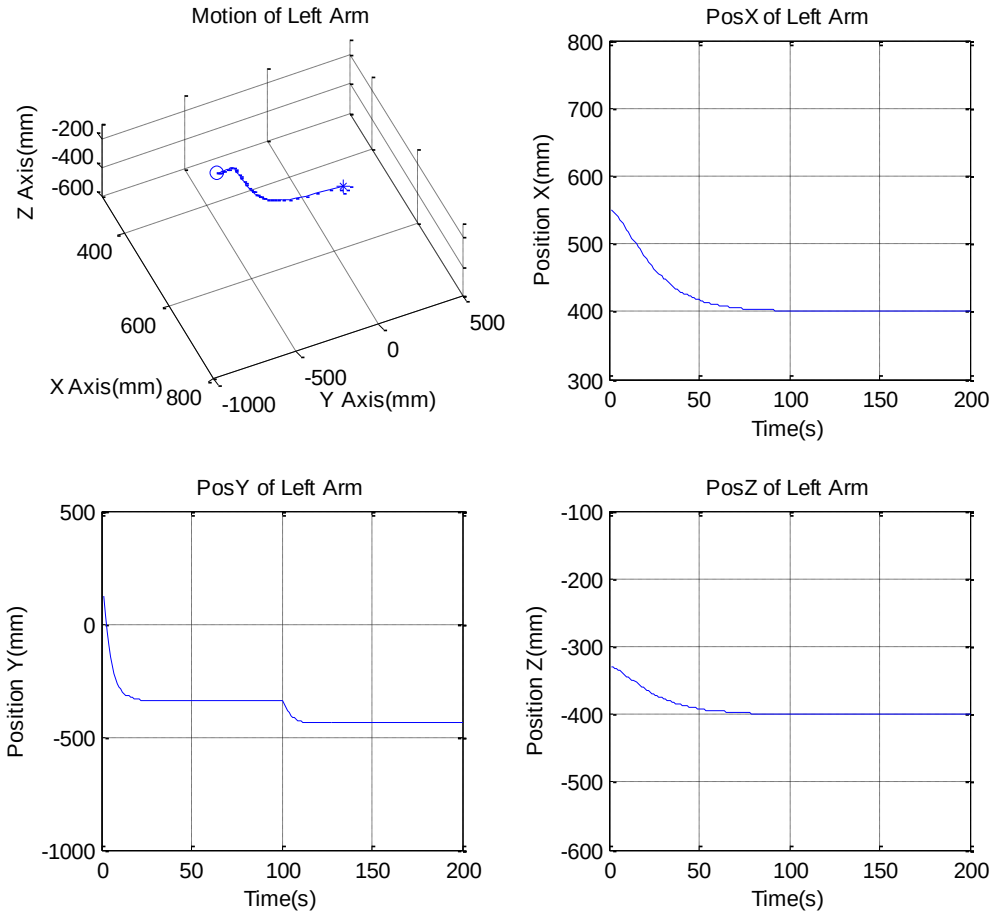


Figure 79 Generated Motion Trajectories for the Left Arm (Experiment 3A-2)

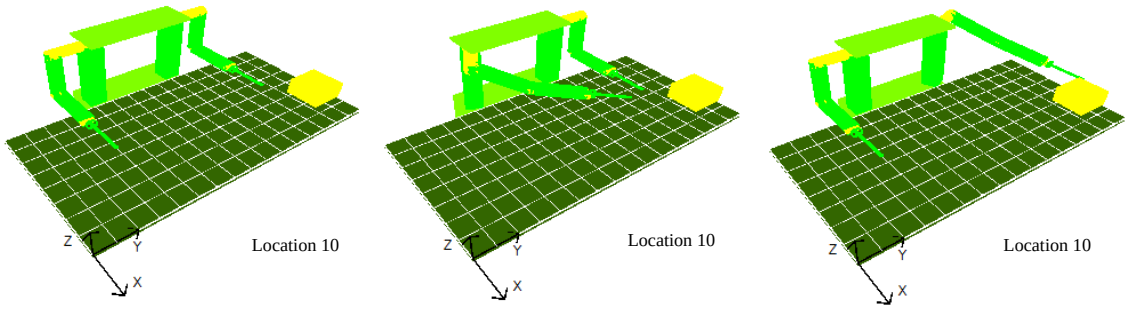


Figure 80 Simulation Results of Experiment 3A-10

In Figure 80, the object is placed at Location 10. ISAC first tries to push the object using its right arm and finds that the object is out of the working space of the right

arm. Then ISAC switches the generated sequence to the left arm and finds that it is out of the working space of the left arm. Then ISAC returns to the home position and displays that the object is out of its working space and it cannot complete the task.

--*Simulation Results of Experiment 3A*

The simulation results are summarized in Table 6.

Table 6 Simulation Results of Experiment 3A

	Feasible/ Infeasible	Left/Right Arm	Failure Reason
Location 1	Feasible	Right Arm	N/A
Location 2	Feasible	Left Arm	N/A
Location 3	Feasible	Left Arm	N/A
Location 4	Feasible	Left Arm	N/A
Location 5	Infeasible	N/A	Right Arm: Collision with the object Left Arm: Out of working Space
Location 6	Feasible	Right Arm	N/A
Location 7	Feasible	Right Arm	N/A
Location 8	Feasible	Left Arm	Right Arm: Collision with the object
Location 9	Feasible	Left Arm	N/A
Location 10	Infeasible	N/A	Right Arm: Out of working space Left Arm: Out of working space

The figures of simulation results and generated motion trajectories for both arms are included in Appendix C.

Experiment 3B: Integrated System with Obstacle

--*Objective*

This experiment is to evaluate the integrated system of behavior-based cognitive control when obstacle exists. Since imitation learning has been described in detail earlier, we assume that ISAC already learned all needed behaviors.

Figure 81 illustrates key system components used in this experiment.

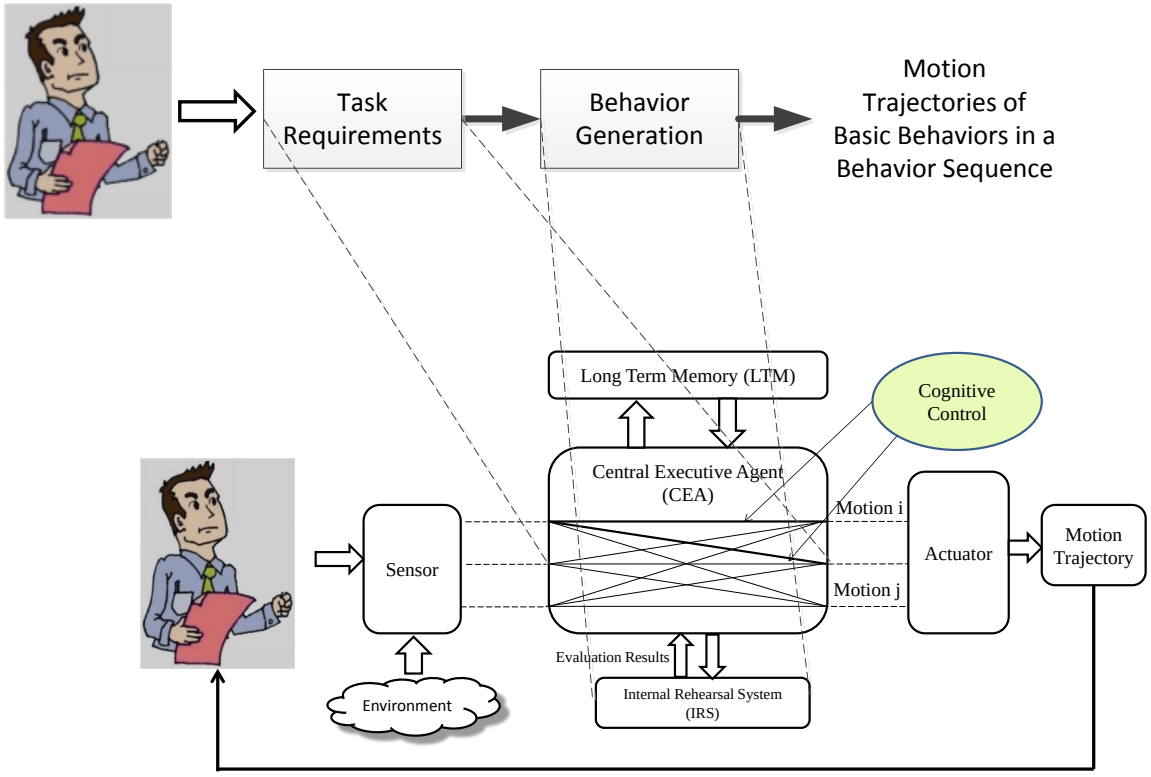


Figure 81 Key Software Components in Experiment 3B

As mentioned earlier, simulation was implemented using Microsoft Visual C# with OpenGL. Major software components are: Central Executive Agent (CEA), Internal Rehearsal System, Long-Term Memory, and Behavior Generation. How they were implemented for this integrated experiment is describe here.

--*Central Executive Agent (CEA)*

The CEA is a rule based module which makes decisions to switch cognitive control processes.

Given a task, the CEA parses the command and finds the required behavior and related parameters. Then the CEA searches the LTM to find whether the behavior has been learned. The input of the searching is the name of the required behavior, and the output is a returned Boolean value. Based on the Boolean value, the CEA uses the following rules:

if searching result is true, then switches to behavior generation;

if searching result is false, then switches to learning stage.

The CEA finds a path from the "Starting" behavior to the required behavior in the constructed graph and generates a behavior sequence. The input of the behavior sequence generation is the name of the required behavior and the output is a behavior sequence.

if behavior sequence generation is completed, then switches to behavior sequence generation

For each behavior in this behavior sequence, a motion trajectory is generated. The generation method is selected from Table 5 in Chapter III. The input of motion trajectory generation is behavior sequence with related parameters and the output is a motion trajectory.

if motion trajectory generation is completed, then switches to motion trajectory generation

The generated motion trajectory is sent to IRS for evaluation. The input of the IRS is the generated motion trajectory and the output is a returned Boolean value. Based on the returned Boolean value, the CEA switches the strategies using the following rules:

if the evaluation of using the right arm is true, then switches to execution

if the evaluation of using the right arm is false, then switches to evaluating the behavior sequence using the left arm

if the evaluation of using the left arm is true, then switches to execution

if the evaluation of using the left arm is false, then switches to displaying a message on the screen and waiting for commands

--Internal Rehearsal System (IRS)

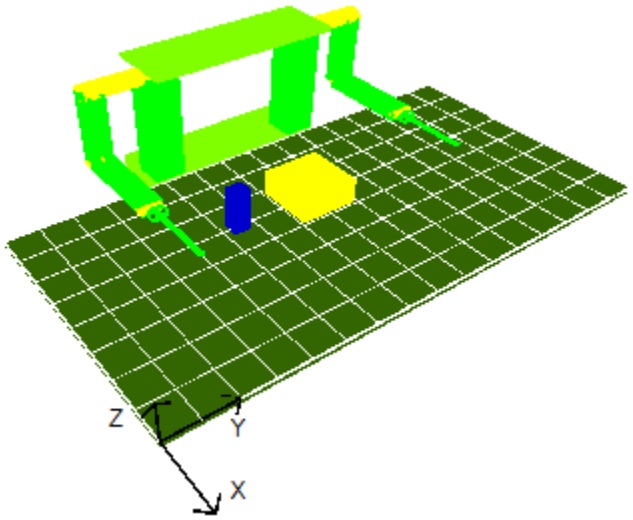


Figure 82 Example of the IRS Environment

Figure 82 displays an example of the environment in IRS, which is developed using Microsoft Visual C# with OpenGL. Two key modules are implemented: display module and evaluation module. In display module, the joint angles of ISAC, the position and sizes of the target object and the obstacle are updated continuously in order to display

the current situation of the evaluation process. In evaluation module, IRS detects the collision of the arms of ISAC with the target object and the obstacle, and tests whether the required via points on the motion trajectory are out of the working space of ISAC. The detailed description of the evaluation module is included in Chapter III.

The input of the IRS is a motion trajectory and the environmental information including the positions and the sizes of the target object and the obstacle. The output of the IRS is a Boolean value describing whether the arms of ISAC collides with the target object and the obstacle and whether the via points on the motion trajectory is within the working space of ISAC, which is sent to the CEA.

--*Long Term Memory (LTM)*

The LTM stores the learned basic behaviors and pre-defined behavior generation methods. Microsoft Visual C# and Microsoft Access 2010 is used for the implementation of the database. The structure of the learned basic behavior is:

Behavior Name
{
Behavior ID
Pre-Condition
Post-Results
Internal Condition
Original Regression Model
Latent Regression Model
Projection Matrix
}

A sample of stored basic behaviors is:

Reaching
{

Behavior ID:	*1*
Pre-Condition:	*0*
Post-Results:	*1*
Internal Condition:	*1*
Original Regression Model:	*null*
Latent Regression Model:	*null*

Projection Matrix: *null*

}

--*Behavior Generation*

In Behavior Generation, a behavior sequence and motion trajectories for all the behaviors in the behavior sequence are generated. The behavior sequence is generated by finding a shortest path in the constructed behavior graph using Dijkastra's algorithm. The motion trajectories are generated using the behavior generation methods described in Table 5 of Chapter III. The generated motion trajectories are sent to the IRS for evaluation. The CEA decides whether the motion trajectories for the left arm or the right arm are sent to the IRS.

--*Simulation Setup*

The target object, a yellow box, is placed in the environment at 8 locations, and ISAC is asked to push it to its right. The size of the box is: 18 cm (length), 18cm (width), and 12 cm (height).

The coordinates of the 8 locations on the table are: Location 1: $\{40, -40, -46\}$, Location 2: $\{40,0, -46\}$, Location 3: $\{40,25, -46\}$, Location 4: $\{40,50, -46\}$, Location 5: $\{70, -40, -46\}$, Location 6: $\{70,0, -46\}$, Location 7: $\{70,25, -46\}$, Location 8: $\{470,50, -46\}$ (-46 is the Z coordinates of the surface of the table) The units of all the coordinates are centimeters.

Figure 83 displays the locations of the object.

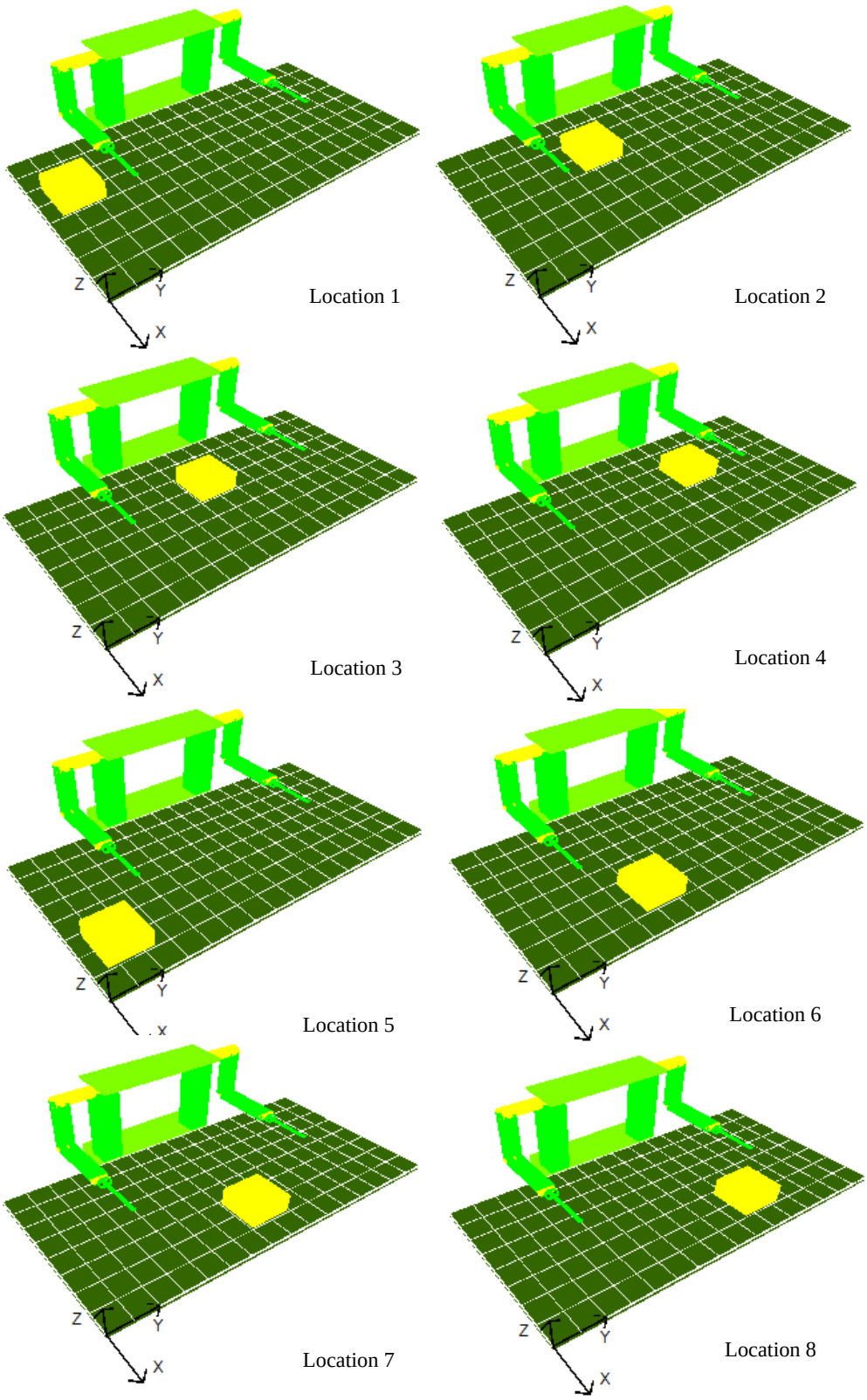


Figure 83 Simulation Setup of Experiment 3B

In each experiment, an obstacle is placed on the table as the obstacle. When the object is placed at each location, the obstacle is placed at four different locations around the object. The positions of the obstacle are:

(a) $\{X\ of\ the\ object, Y\ of\ the\ object + 25, Z\ of\ the\ object\}$,

(b) $\{X\ of\ the\ object + 20, Y\ of\ the\ object, Z\ of\ the\ object\}$,

(c) $\{X\ of\ the\ object, Y\ of\ the\ object - 25, Z\ of\ the\ object\}$,

(d) $\{X\ of\ the\ object - 30, Y\ of\ the\ object, Zof\ the\ object\}$

All the units of the coordinates are centimeters. Figure 84 displays the simulation setup when the target object is placed at location 3.

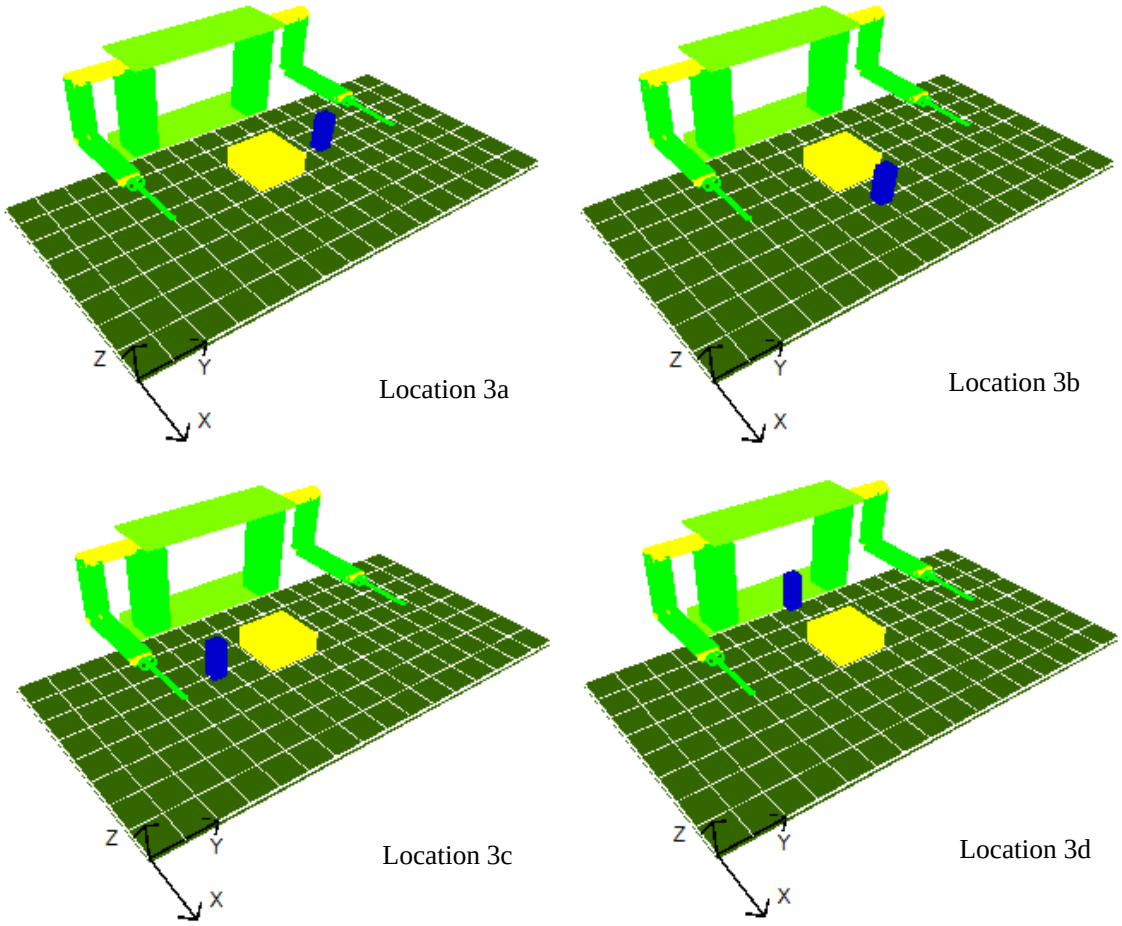


Figure 84 Locations of the Obstacles in Experiment 3B-3

The obstacles used in the simulation are blue boxes with the sizes: (1) 1 cm (length), 1cm (width), and 20 cm (height); (2) 5 cm (length), 5cm (width), and 20 cm (height); (3) 10 cm (length), 10 cm (width), and 20 cm (height); (4) 20 cm (length), 20 cm (width), and 20 cm (height). Figure 85 displays that when the target object is placed at location 3, and the obstacle is also placed at location 3c, the obstacles with different sizes are placed on the table.

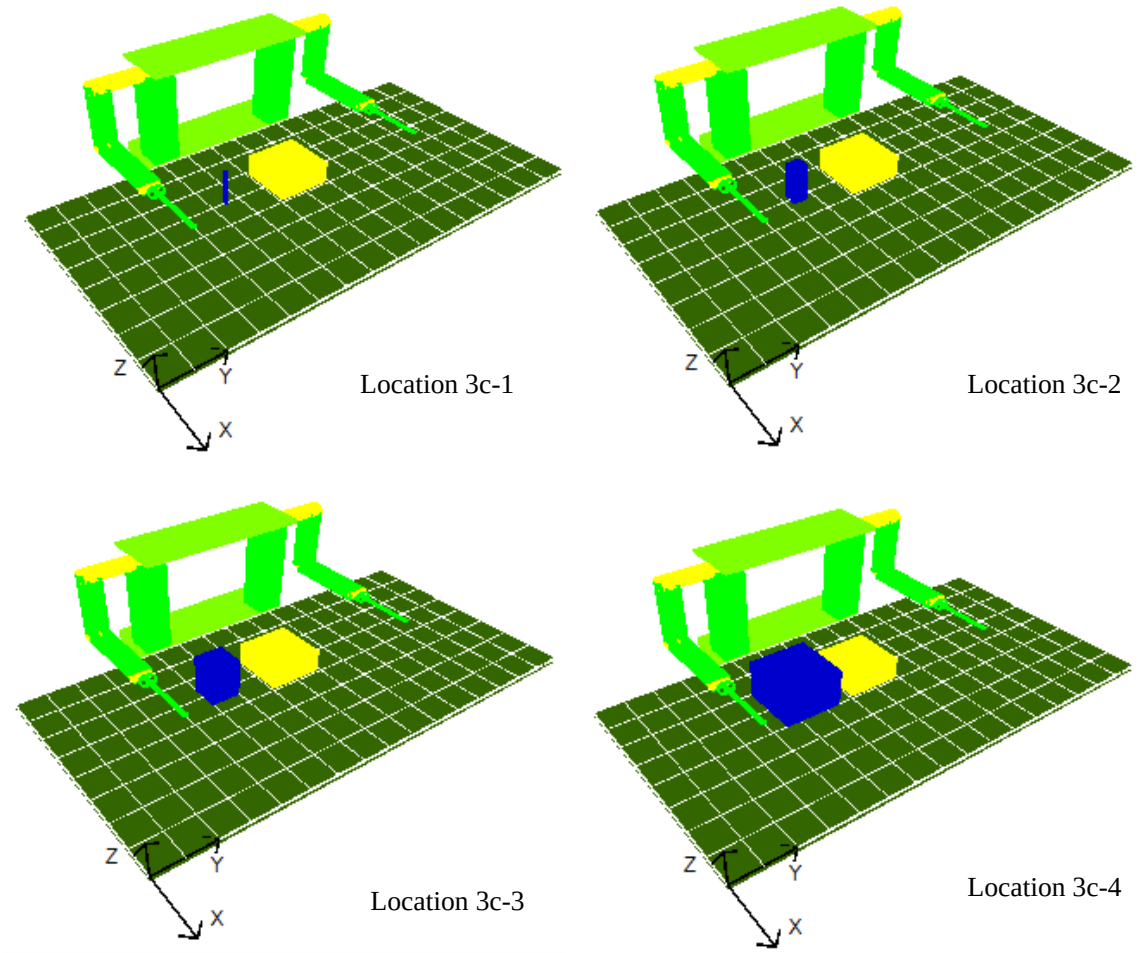


Figure 85 Different Sizes of Obstacles in Experiment 3B-3c

Thus, there are 128 simulation experiments. When the object is placed at each location, there are 16 simulation experiments.

The indices of the experiments Experiment 3B are designed as: 3B-XY-Z. X is related to the index number of location of the target object, Y is related to the position of obstacle, Z is related to the size of the obstacle used.

ISAC is asked to push the box to the right while avoiding the obstacle and avoiding hitting the box before the "Pushing Right" behavior.

The given command is: "Push the box to the right". ISAC parses the speech command and find the required behavior is "Pushing Right", the parameters are "the box" and "to the right". It searches the behavior graph and generates a behavior sequence: $\{Starting, Reaching, Pushing\ Right\}$. Because the obstacle is detected in the environment, ISAC will generate motion trajectories to avoid this obstacle and switch strategies by using the right arm or the left arm to complete the required task. If ISAC finds that the target is out of its working space or it cannot push the object while avoiding the obstacle, it displays a message on the screen.

Example: Experiment 3B-6

The target object is placed at location 7: $\{70, 0, -46\}$, the obstacle is placed around the yellow box at four locations: $\{70, 25, -46\}$, $\{90, 0, -46\}$, and $\{70, -25, -46\}$, $\{40, 0, -46\}$. ISAC is asked to push the object to the right using one of its arms.

In Figure 86, the obstacle is placed at $\{70, 25, -46\}$ and the size of the obstacle is 1 cm (length), 1cm (width), and 20 cm (height). ISAC pushes the object to the right using its right arm. In Figure 87, the size of the obstacle is increased to 5 cm (length), 5cm (width), and 20 cm (height), and ISAC pushes the object to the right using its right arm. Figure 88 displays the generated motion trajectory of the right arm.

Given a command, ISAC first checks the LTM to generate a behavior sequence which is composed of $\{Reaching, Pushing\}$. Then it generates motion trajectories for all the behaviors in this behavior sequence. 2nd-Order attractor is used according to the selection methods in Table 5 of Chapter III. The generated motion trajectories are sent to the IRS for evaluation. When size of the obstacles are 1 cm (length), 1cm (width), and 20 cm (height) and 5 cm (length), 5cm (width), and 5 cm (height), the returned evaluation results demonstrate that ISAC can push the box using its right arm. Then the CEA switches the process to execution.

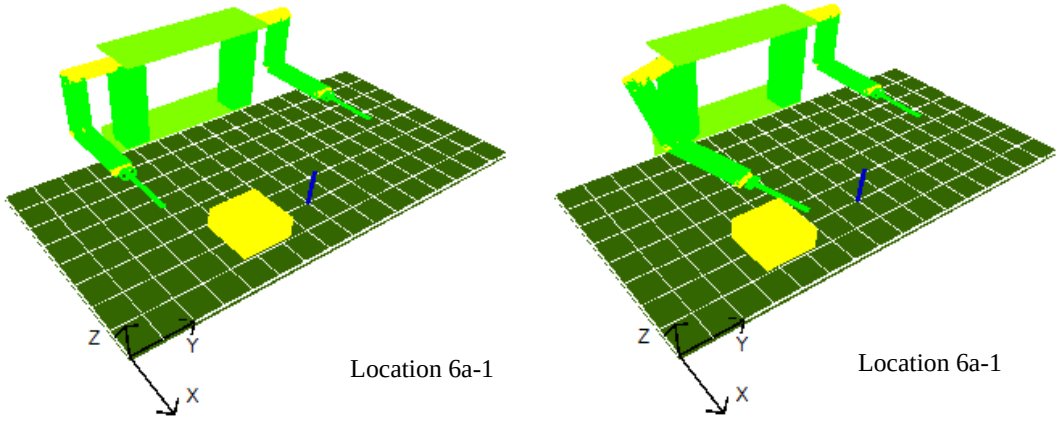


Figure 86 Simulation Results of Experiment 3B-6a-1

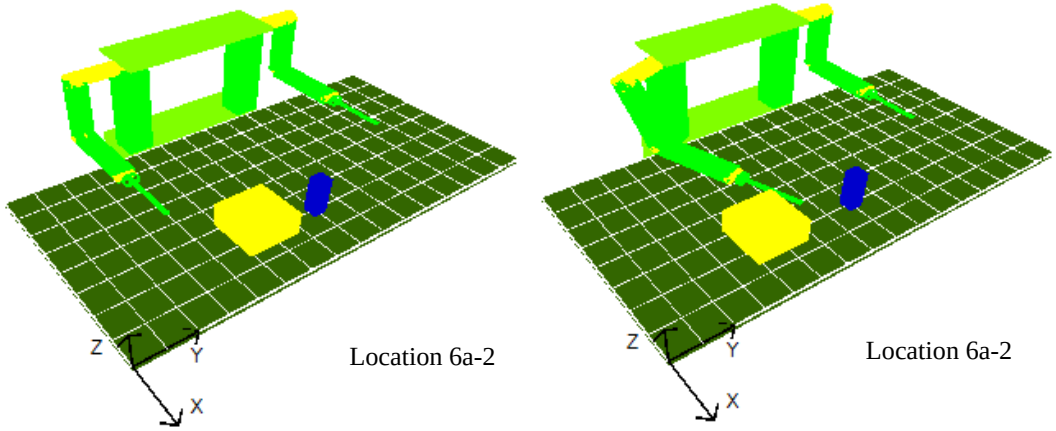


Figure 87 Simulation Results of Experiment 3B-6a-2

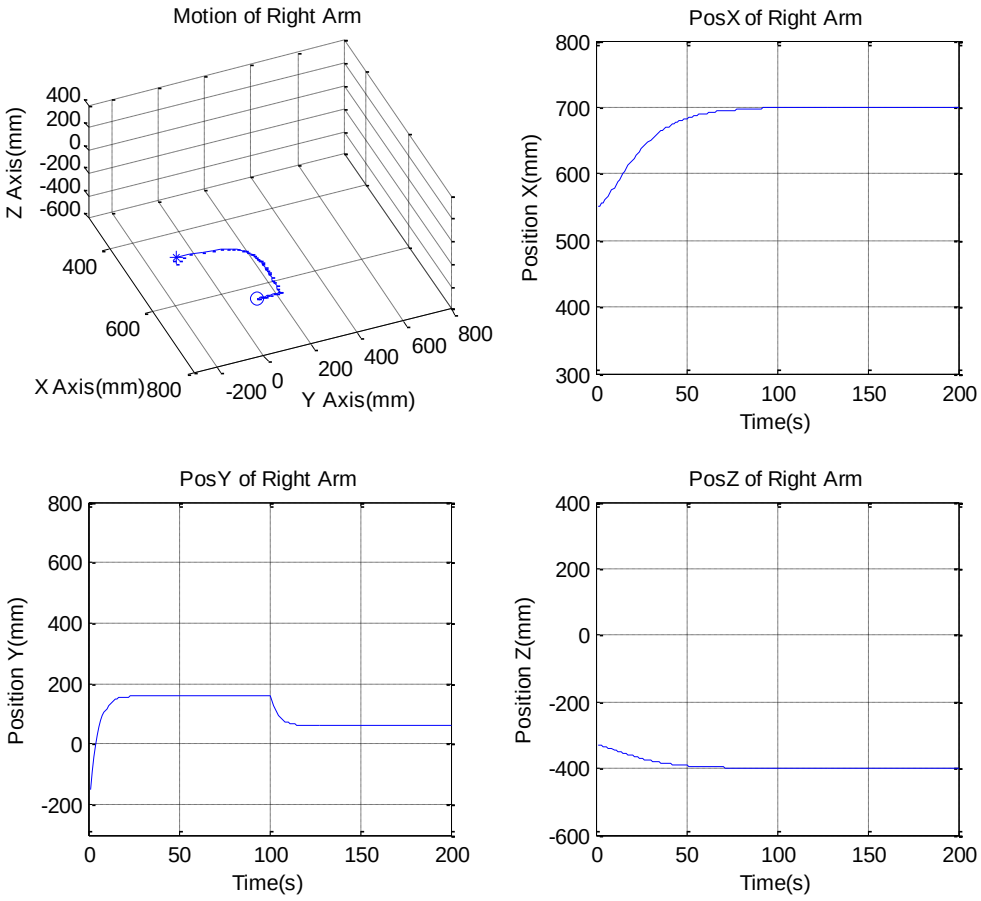


Figure 88 Generated Motion Trajectories for the Right Arm (Experiment 3B-6a-1/2)

The size of the obstacle is increased to 10 cm (length), 10 cm (width), and 20 cm (height) in Figure 89. Given a command, ISAC first checks the LTM to generate a behavior sequence which is composed of $\{Reaching, Pushing\}$. Then it generates motion trajectories for all the behaviors in this behavior sequence. 2nd-Order attractor is used based according to the selection method in Table 5 of Chapter III. The generated motion trajectory for the right arm is sent to the IRS for evaluation. The returned evaluation results demonstrate that ISAC cannot push the box using its right arm because it has to avoid the obstacle. Then the CEA switches the strategy and transfer the

generated motion trajectories to the left arm. The evaluation result demonstrates that ISAC can push the box using its left arm. Then the CEA switches the process to execution.

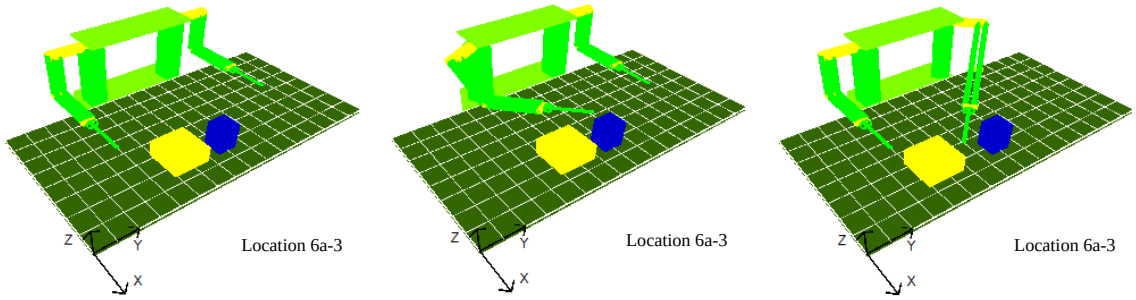


Figure 89 Simulation Results of Experiment 3B-6a-3

Figure 90 and Figure 91 displays the generated motion trajectories for the left arm and the right arm in Experiment 3B-6a-3 respectively.

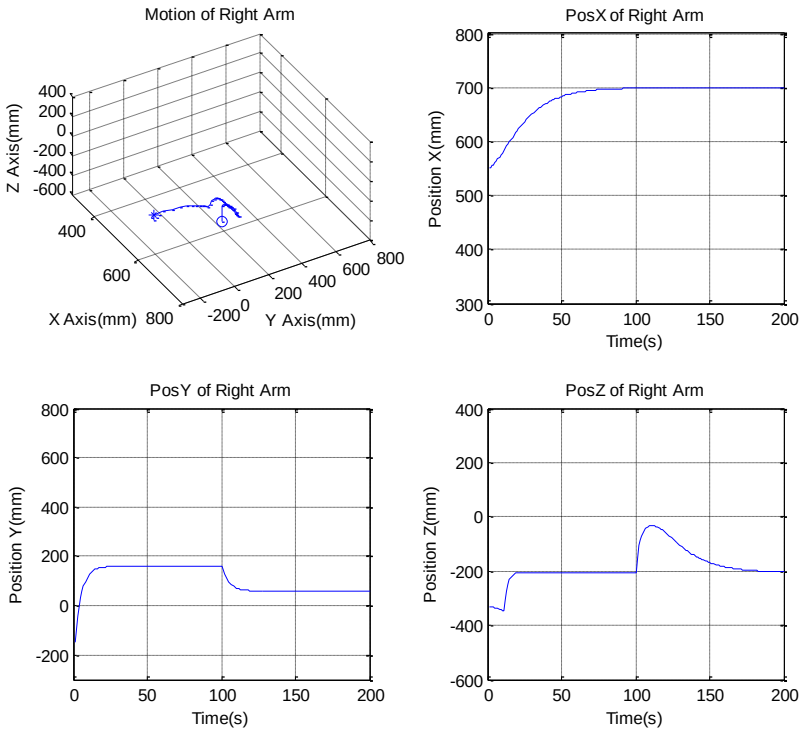


Figure 90 Generated Motion Trajectories for the Right Arm (Experiment 3B-6a-3)

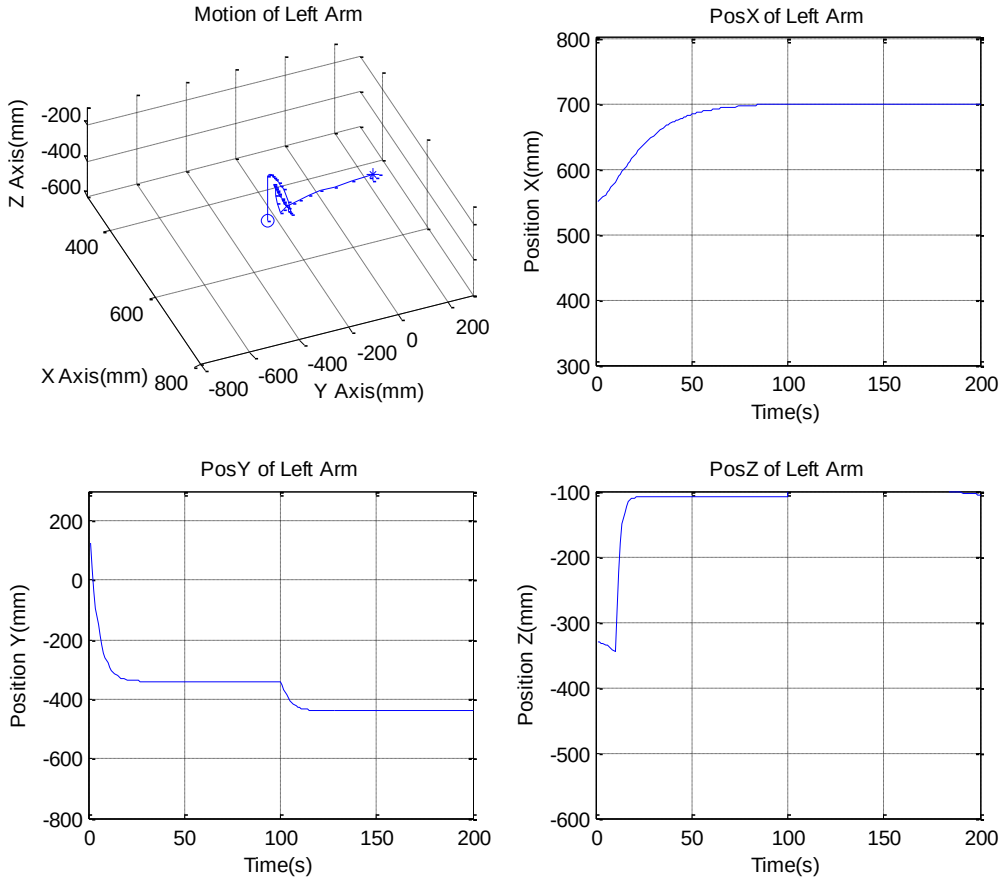


Figure 91 Generated Motion Trajectories for the Left Arm (Experiment 3B-6a-3)

The size of the obstacle is increased to 20 cm (length), 20 cm (width), and 10 cm (height) in Figure 92. Given a command, ISAC first checks the LTM to generate a behavior sequence which is composed of $\{Reaching, Pushing\}$. Then it generates motion trajectories for all the behaviors in this behavior sequence. 2nd-Order attractor is used based according to the selection method in Table 5 of Chapter III. The generated motion trajectory for the right arm is sent to the IRS for evaluation. The returned evaluation results demonstrate that ISAC cannot push the box using its right arm because it has to avoid the obstacle. Then the CEA switches the strategy and transfer the generated motion trajectories to the left arm. The evaluation result demonstrates that

ISAC cannot push the box using its left arm because it has to avoid the obstacle. Then the CEA decide to return to the home position and display a message on the screen.

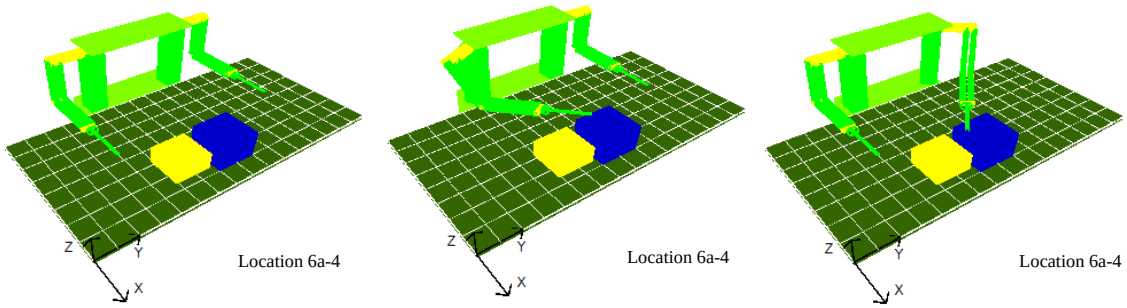


Figure 92 Simulation Results of Experiment 3B-6a-4

Figure 93 and Figure 94 display the generated motion trajectories for the right and left arm in Experiment 3B-6a-4 respectively.

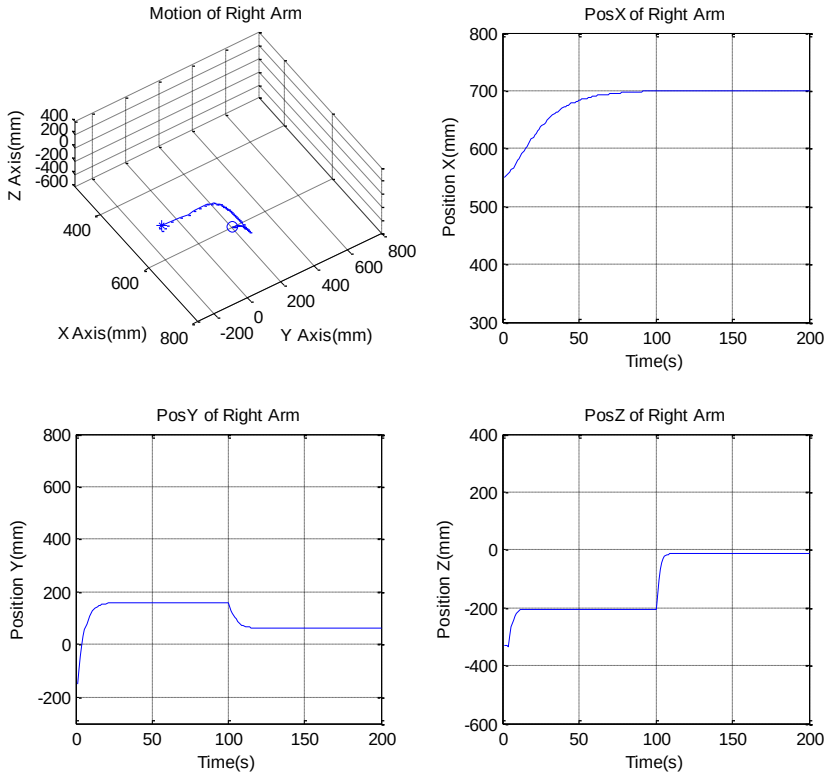


Figure 93 Generated Motion Trajectories for the Right Arm (Experiment 3B-6a-4)

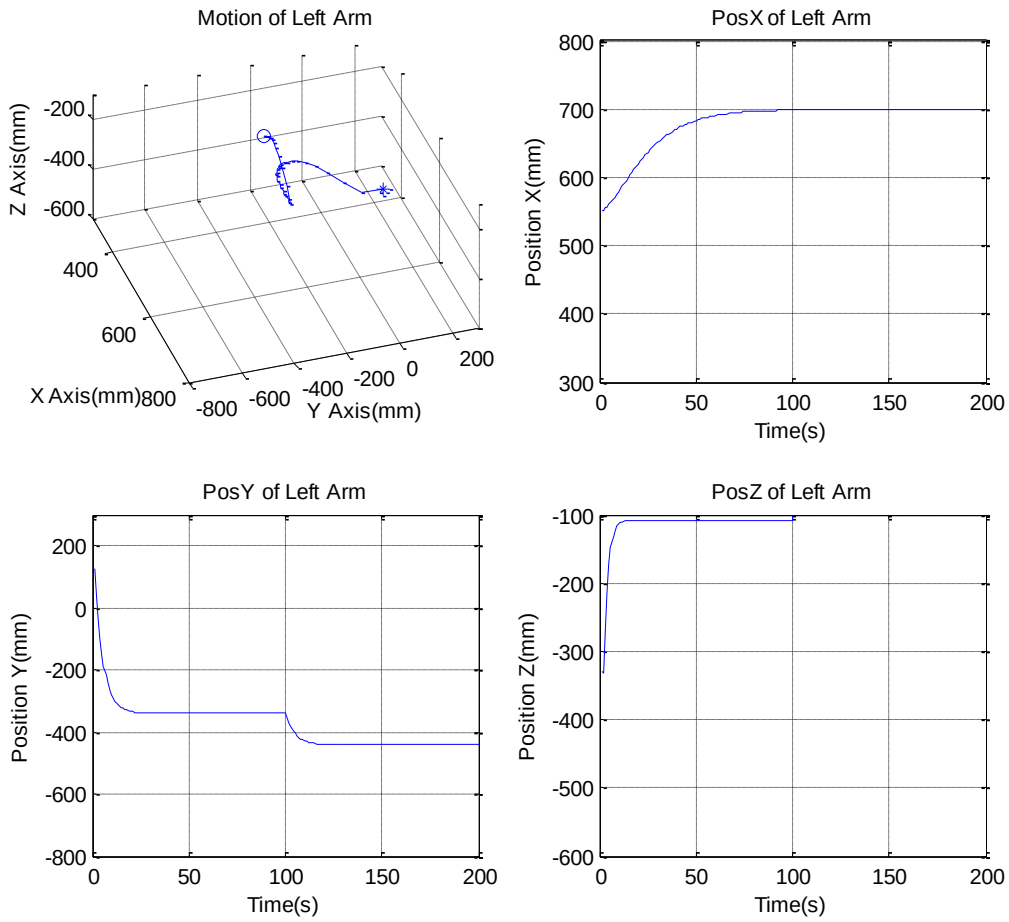


Figure 94 Generated Motion Trajectories for the Left Arm (Experiment 3B-6a-4)

In this example, ISAC switches strategies to complete the task when the obstacle with different sizes is placed at the same location. When the size increases, ISAC finds that it cannot push the box using its right arm and transfer the generated behavior sequence to the left arm. Based on the evaluation results from the IRS, ISAC makes decisions to choose switching strategies or displaying a message on the screen to tell human why it cannot complete the task.

The pictures of simulation results and generated motion trajectories for Experiment 3B are included in Appendix D.

Table 7 Simulation Results of Experiment 3B-1

	Feasible/ Infeasible	Left/Right	Failure Reason
Experiment 3B-1a-1	Feasible	Right Arm	N/A
Experiment 3B-1a-2	Feasible	Right Arm	N/A
Experiment 3B-1a-3	Infeasible	N/A	Right Arm: Cannot Reach the Box Left Arm: Cannot Reach the Box
Experiment 3B-1a-4	Infeasible	N/A	Right Arm: Cannot Reach the Box Left Arm: Cannot Reach the Box
Experiment 3B-1b-1	Feasible	Right Arm	N/A
Experiment 3B-1b-2	Feasible	Right Arm	N/A
Experiment 3B-1b-3	Feasible	Right Arm	N/A
Experiment 3B-1b-4	Infeasible	Left Arm	Right Arm: Cannot Reach the Box Left Arm: Cannot Reach the Box
Experiment 3B-1c-1	Feasible	Right Arm	N/A
Experiment 3B-1c-2	Feasible	Right Arm	N/A
Experiment 3B-1c-3	Feasible	Right Arm	N/A
Experiment 3B-1c-4	Feasible	Right Arm	N/A
Experiment 3B-1d-1	Feasible	Right Arm	N/A
Experiment 3B-1d-2	Feasible	Right Arm	N/A
Experiment 3B-1d-3	Feasible	Right Arm	N/A
Experiment 3B-1d-4	Infeasible	N/A	Right Arm: Collision with the obstacle Left Arm: Cannot Reach the Box

Table 8 Simulation Results of Experiment 3B-2

	Feasible/ Infeasible	Left/Right	Failure Reason
Experiment 3B-2a-1	Infeasible	N/A	Right Arm: Cannot Reach the Box Left Arm: Collision with the obstacle
Experiment 3B-2a-2	Infeasible	N/A	Right Arm: Cannot Reach the Box Left Arm: Collision with the obstacle
Experiment 3B-2a-3	Infeasible	N/A	Right Arm: Cannot Reach the Box Left Arm: Collision with the obstacle
Experiment 3B-2a-4	Infeasible	N/A	Right Arm: Cannot Reach the Box Left Arm: Collision with the obstacle
Experiment 3B-2b-1	Feasible	Left Arm	Right Arm: Cannot Reach the Box
Experiment 3B-2b-2	Feasible	Left Arm	Right Arm: Cannot Reach the Box
Experiment 3B-2b-3	Feasible	Left Arm	Right Arm: Cannot Reach the Box
Experiment 3B-2b-4	Feasible	Left Arm	Right Arm: Cannot Reach the Box
Experiment 3B-2c-1	Feasible	Left Arm	Right Arm: Cannot Reach the Box
Experiment 3B-2c-2	Feasible	Left Arm	Right Arm: Cannot Reach the Box
Experiment 3B-2c-3	Feasible	Left Arm	Right Arm: Cannot Reach the Box
Experiment 3B-2c-4	Feasible	Left Arm	Right Arm: Cannot Reach the Box
Experiment 3B-2d-1	Feasible	Left Arm	Right Arm: Cannot Reach the Box
Experiment 3B-2d-2	Feasible	Left Arm	Right Arm: Cannot Reach the Box
Experiment 3B-2d-3	Feasible	Left Arm	Right Arm: Cannot Reach the Box
Experiment 3B-2d-4	Feasible	Left Arm	Right Arm: Cannot Reach the Box

Table 9 Simulation Results of Experiment 3B-3

	Feasible/ Infeasible	Left/Right	Failure Reason for the Left/Right Arm
Experiment 3B-3a-1	Infeasible	N/A	Right Arm: Collision with the obstacle Left Arm: Collision with the obstacle
Experiment 3B-3a-2	Infeasible	N/A	Right Arm: Collision with the obstacle Left Arm: Collision with the obstacle
Experiment 3B-3a-3	Infeasible	N/A	Right Arm: Collision with the obstacle Left Arm: Collision with the obstacle
Experiment 3B-3a-4	Infeasible	N/A	Right Arm: Collision with the obstacle Left Arm: Collision with the obstacle
Experiment 3B-3b-1	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-3b-2	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-3b-3	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-3b-4	Infeasible	N/A	Right Arm: Collision with the object Left Arm: Cannot reach the box
Experiment 3B-3c-1	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-3c-2	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-3c-3	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-3c-4	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-3d-1	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-3d-2	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-3d-3	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-3d-4	Feasible	Left Arm	Right Arm: Collision with the object

Table 10 Simulation Results of Experiment 3B-4

	Feasible/ Infeasible	Left/Right	Failure Reason for the Left/Right Arm
Experiment 3B-4a-1	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-4a-2	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-4a-3	Infeasible	N/A	Right Arm: Collision with the object Left Arm: Cannot reach the box
Experiment 3B-4a-4	Infeasible	N/A	Right Arm: Collision with the object Left Arm: Collision with the obstacle
Experiment 3B-4b-1	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-4b-2	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-4b-3	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-4b-4	Infeasible	N/A	Right Arm: Collision with the object Left Arm: Cannot reach the box
Experiment 3B-4c-1	Feasible	Left Arm	Right Arm: Collision with the obstacle
Experiment 3B-4c-2	Feasible	Left Arm	Right Arm: Collision with the obstacle
Experiment 3B-4c-3	Feasible	Left Arm	Right Arm: Collision with the obstacle
Experiment 3B-4c-4	Feasible	Left Arm	Right Arm: Collision with the obstacle
Experiment 3B-4d-1	Feasible	Left Arm	Right Arm: Collision with the obstacle
Experiment 3B-4d-2	Feasible	Left Arm	Right Arm: Collision with the obstacle
Experiment 3B-4d-3	Feasible	Left Arm	Right Arm: Collision with the obstacle
Experiment 3B-4d-4	Infeasible	N/A	Right Arm: Collision with the obstacle Left Arm: Collision with the obstacle

Table 11 Simulation Results of Experiment 3B-5

	Feasible/ Infeasible	Left/Right	Failure Reason for the Left/Right Arm
Experiment 3B-5a-1	Feasible	Right Arm	N/A
Experiment 3B-5a-2	Feasible	Right Arm	N/A
Experiment 3B-5a-3	Infeasible	N/A	Right Arm: Cannot Reach the Box Left Arm: Cannot Reach the Box
Experiment 3B-5a-4	Infeasible	N/A	Right Arm: Cannot Reach the Box Left Arm: Cannot Reach the Box
Experiment 3B-5b-1	Feasible	Right Arm	N/A
Experiment 3B-5b-2	Feasible	Right Arm	N/A
Experiment 3B-5b-3	Feasible	Right Arm	N/A
Experiment 3B-5b-4	Infeasible	Left Arm	Right Arm: Cannot Reach the Box Left Arm: Cannot Reach the Box
Experiment 3B-5c-1	Feasible	Right Arm	N/A
Experiment 3B-5c-2	Feasible	Right Arm	N/A
Experiment 3B-5c-3	Feasible	Right Arm	N/A
Experiment 3B-5c-4	Feasible	Right Arm	N/A
Experiment 3B-5d-1	Feasible	Right Arm	N/A
Experiment 3B-5d-2	Feasible	Right Arm	N/A
Experiment 3B-5d-3	Feasible	Right Arm	N/A
Experiment 3B-5d-4	Infeasible	Right Arm	Right Arm: Collision with the obstacle Left Arm: Cannot Reach the Box

Table 12 Simulation Results of Experiment 3B-6

	Feasible/ Infeasible	Left/Right	Failure Reason for the Left/Right Arm
Experiment 3B-6a-1	Feasible	Right Arm	N/A
Experiment 3B-6a-2	Feasible	Right Arm	N/A
Experiment 3B-6a-3	Feasible	Left Arm	N/A
Experiment 3B-6a-4	Infeasible	N/A	Right Arm: Cannot Reach the Box Left Arm: Cannot Reach the Box
Experiment 3B-6b-1	Feasible	Right Arm	N/A
Experiment 3B-6b-2	Feasible	Right Arm	N/A
Experiment 3B-6b-3	Feasible	Right Arm	N/A
Experiment 3B-6b-4	Feasible	Left Arm	Right Arm: Cannot Reach the Box
Experiment 3B-6c-1	Feasible	Right Arm	N/A
Experiment 3B-6c-2	Feasible	Right Arm	N/A
Experiment 3B-6c-3	Feasible	Right Arm	N/A
Experiment 3B-6c-4	Feasible	Right Arm	N/A
Experiment 3B-6d-1	Feasible	Right Arm	N/A
Experiment 3B-6d-2	Feasible	Right Arm	N/A
Experiment 3B-6d-3	Feasible	Left Arm	Right Arm: Cannot reach the Box
Experiment 3B-6d-4	Feasible	Left Arm	Right Arm: Cannot reach the Box

Table 13 Simulation Results of Experiment 3B-7

	Feasible/ Infeasible	Left/Right	Failure Reason for the Left/Right Arm
Experiment 3B-7a-1	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-7a-2	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-7a-3	Infeasible	N/A	Right Arm: Cannot Reach the Box Left Arm: Cannot Reach the Box
Experiment 3B-7a-4	Infeasible	N/A	Right Arm: Cannot Reach the Box Left Arm: Cannot Reach the Box
Experiment 3B-7b-1	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-7b-2	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-7b-3	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-7b-4	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-7d-1	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-7d-2	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-7d-3	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-7d-4	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-7d-1	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-7d-2	Feasible	Left Arm	Right Arm: Collision with the obstacle
Experiment 3B-7d-3	Feasible	Left Arm	Right Arm: Collision with the obstacle
Experiment 3B-7d-4	Infeasible	N/A	Right Arm: Collision with the obstacle Left Arm: Collision with the object

Table 14 Simulation Results of Experiment 3B-8

	Feasible/ Infeasible	Left/Right	Failure Reason for the Left/Right Arm
Experiment 3B-8a-1	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-8a-2	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-8a-3	Infeasible	N/A	Right Arm: Collision with the object Left Arm: Cannot reach the box
Experiment 3B-8a-4	Infeasible	N/A	Right Arm: Collision with the object Left Arm: Cannot reach the box
Experiment 3B-8b-1	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-8b-2	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-8b-3	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-8b-4	Infeasible	N/A	Right Arm: Collision with the object Left Arm: Cannot reach the box
Experiment 3B-8c-1	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-8c-2	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-8c-3	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-8c-4	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-8d-1	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-8d-2	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-8d-3	Feasible	Left Arm	Right Arm: Collision with the object
Experiment 3B-8d-4	Infeasible	N/A	Right Arm: Collision with the object Left Arm: Collision with the obstacle

These tables display that ISAC is able to switch strategies according to the current situation. When the object is placed close to the right arm, the probability of using the right arm to complete the task is high; when the object is placed far away from the right arm, the probability of using the left arm to complete the task becomes high. The positions and the sizes of the obstacle also affect the selection of the arms. Quantitative evaluation will be explained in Chapter V.

Summary

Three experiments are designed to validate the designed system. One experiment is carried out on ISAC robot, and the other two are implemented in simulation. According to the simulation and experimental results, ISAC can complete the requirements of our designed system as described in the beginning of this chapter.

In the next chapter, the system performance will be evaluated by analyzing the similarity of the generated motion trajectory and the demonstrations, the running time of key modules in this system, and the success rate of the cognitive control mechanism.

CHAPTER V

EVALUATION OF SYSTEM PERFORMANCE

The aim of this dissertation was to integrate imitation learning with cognitive control for a humanoid robot's skill learning. In this chapter, we will evaluate the major results produced by the three experiments.

Behavior Generation

Experiment 1A and 1B

In Experiment 1, ISAC was asked to push a box placed on a table in front of it without grasping. In this experiment, ISAC needed to learn the "Reaching", "Pushing Left" and "Pushing Right" behavior, and generated new behavior sequences to complete the task.

The most common feature of the "Reaching" behavior is to minimize the distance of the end-effector and the target position at the end of its motion trajectory. There is no requirement of internal constraint and pre-condition. The most common features of the "Pushing Left" and "Pushing Right" behaviors are to minimize the distance of the end-effector and the target object at the beginning of its motion trajectory and to minimize the distance of the end-effector and the target position at the end of its motion trajectory.

There are no requirements for the internal constraint and the pre-condition for the "Pushing Left" and "Pushing Right" behaviors.

For the "Reaching" behavior, the distances between the end-effector and the target position were measured to evaluate how accurate the generated results are. For the "Pushing" behavior, the DTW distances between the generated motion trajectories and the demonstrated motion trajectory was measured. There is no failure because human teacher shows the pushing points that ISAC can use to push the object. The Index of the Experiment 1 is represented as Experiment 1-XY. X includes L and R, which are related to the Pushing Left or Pushing Right experiment respectively. Y includes 1, 2, 3, and 4, which are related to the number of the location of the object.

In Table 15, the average value of the distance between the end-effector and the target position of the "Reaching" behavior is 3.33cm and the average value of the DTW distance between the demonstrated motion trajectory and the generated motion trajectories for the "Pushing" behavior is 4.41 cm. The overall error, the average of which is 1.71cm, is measured by reading the encoder values and computing the distance between the end-effector and the target position at the end of "Reaching" and "Pushing" behaviors. The overall error, which is measured by reading the position value from the Kinect sensor and computing the distance between the end-effector and the target position at the end of "Reaching" and "Pushing" behaviors, is 3.09. The error from the Kinect sensor is larger than the error from the encoder value. The reason is that the box is not pushed straightly to its left or right. So the error comes from both the position of the end-effector and the orientation of the box.

Table 15 Evaluation Results of Behavior Errors of Experiment 1A and 1B

	Reaching	Pushing	Overall (Encoder)	Overall (Kinect Camera)
Experiment 1-L1	4.91	4.00	1.66	3.04
Experiment 1-L2	4.21	5.99	1.06	4.53
Experiment 1-L3	1.57	1.13	0.91	5.17
Experiment 1-L4	2.20	0.99	0.22	6.02
Experiment 1-R1	1.73	11.05	1.05	3.62
Experiment 1-R2	10.76	6.48	4.62	0.20
Experiment 1-R3	0.82	0.47	2.73	1.19
Experiment 1-R4	0.88	4.79	1.41	0.97
Average	3.38	4.36	1.71	3.09
STD	3.33	4.41	1.38	2.13

Unit: Centimeter (cm)

Comparing the size of the object used in Experiment, which is 18cm × 18cm × 12 cm, and the error values after the reaching behavior are smaller than the threshold value: 12 cm, and within the manipulation area of the task. So according to the rules of the DMM described in Chapter III, the DMM considers that the error is acceptable. (If

the errors become larger than the threshold value: 12cm, ISAC cannot push the box. Then ISAC needs to use the compensator described in Chapter IV to overcome the error.)

Experiment 1C

Table 16 Experimental Results of Experiment 1C

	Error without Compensator	Error with Compensator	Ratio
Experiment 1C-L1	3.33	0.09	2.70%
Experiment 1C-L2	3.94	0.60	15.23%
Experiment 1C-L3	4.91	0.53	10.79%
Experiment 1C-L4	4.64	0.48	10.34%
Experiment 1C-R1	3.00	1.72	57.33%
Experiment 1C-R2	2.33	0.69	29.61%
Experiment 1C-R3	3.97	0.81	20.40%
Experiment 1C-R4	1.57	0.60	38.22%
Average	3.46	0.69	19.93%
STD	1.14	0.47	40.97%

Unit: Centimeter (cm)

In this Experiment, ISAC uses the compensator to overcome the errors generated by the hardware. In Experiment 1A and 1B, the errors are smaller than the threshold value. According to the rules of the DMM, these errors are acceptable. In Experiment 1C, ISAC is asked to use the compensator to overcome the errors even the errors are acceptable. The target is to test how ISAC can overcome the errors using the compensator.

Table 16 displays the experimental results of the Experiment 1C. The ratios are computed by dividing the "Error with Compensator" with "Error without Compensator".

In Table 16, the average error without using compensator is 3.46cm and the average error using compensator is 0.47cm. After using the compensator, the average error becomes 19.93% of the average error before using the compensator. From Table 16, we can conclude that the compensator improves the performance of the ISAC arm control.

Experiment 2

In Experiment 2, ISAC needs to learn the new behavior in a demonstrated behavior sequence. The behavior sequence is composed of the "Reaching", "Grasping", and the "Yo-Yo Motion" behaviors. The stored "Reaching" behavior is described with $\{0,0,1\}$ in the LTM. The demonstrated behavior sequence is segmented into "Reaching", "Grasping", and the "Yo-Yo Motion" behaviors. ISAC searches the LTM to find whether it has learned these behaviors. It successfully finds that it has learned the "Reaching" behavior. The success rate of the searching is 100%. The "Grasping" behavior is added into the LTM by the human teacher.

In Experiment 2, motion trajectories which are similar to the demonstrations are generated for the "Yo-Yo Motion" behavior. A typical evaluation method is to evaluate

the similarity between the generated motions and the demonstrations. Dynamic Time Warping (DTW) [Berndt and Clifford, 1994] and the Cosine Similarity [Liu and Schneider, 2012] methods are used to quantitatively evaluate the similarity between the generated "Yo-Yo Motion" behavior and the generalized "Yo-Yo Motion" behavior.

Dynamic time warping (DTW) is an algorithm for measuring similarity between two sequences (e.g., speech signals, motion trajectories, etc.) which may vary in time or speed. It finds an optimal match between two sequences of feature vectors which allows for stretched and compressed sections of the sequence.

Using DTW method, the normalized distances between the generated "Yo-Yo Motion" behavior and the demonstrated "Yo-Yo Motion" behavior are computed and divided by the overall length of the motion trajectory. The distances computed by DTW are shown in the first column of Table 17. These distances are very small numbers compared to the overall length of the motion trajectory, which means the generated motion trajectories are similar to the demonstrations.

Table 17 Evaluation of Similarity in Experiment 2

	DTW Distance	Ratio
Generated Trajectory 1	1.98 cm	0.2%
Generated Trajectory 2	1.70 cm	0.17%
Generated Trajectory 3	2.16 cm	0.22%

The normalized overall length of the generalized motion trajectory is 990cm. We set the criterion of similarity is 2%. From the evaluation results using DTW in Table 17, we can conclude that the generated motion trajectories are similar to the demonstrations, which satisfies the requirement of imitation learning.

The analysis above demonstrates that ISAC can learn the behaviors when the human teacher demonstrates the behaviors in several demonstrations. Another target is to teach ISAC to learn cyclic motions. The human teacher demonstrates how to play the Yo-Yo five times consecutively in one demonstration. Figure 95 displays the recorded motion trajectories in one demonstration.

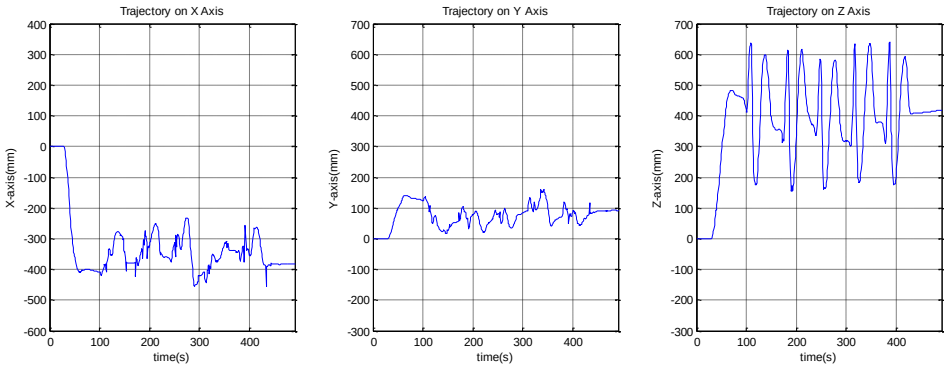


Figure 95 Recorded Motion Trajectories of Cyclic "Yo-Yo Motion"

The left picture and the right picture of Figure 95 are considered as oscillations. The right picture of Figure 95 displays five "Yo-Yo Motions". These five "Yo-Yo Motions" are segmented manually by the human teacher and normalized. The normalized results are displayed in Figure 96.

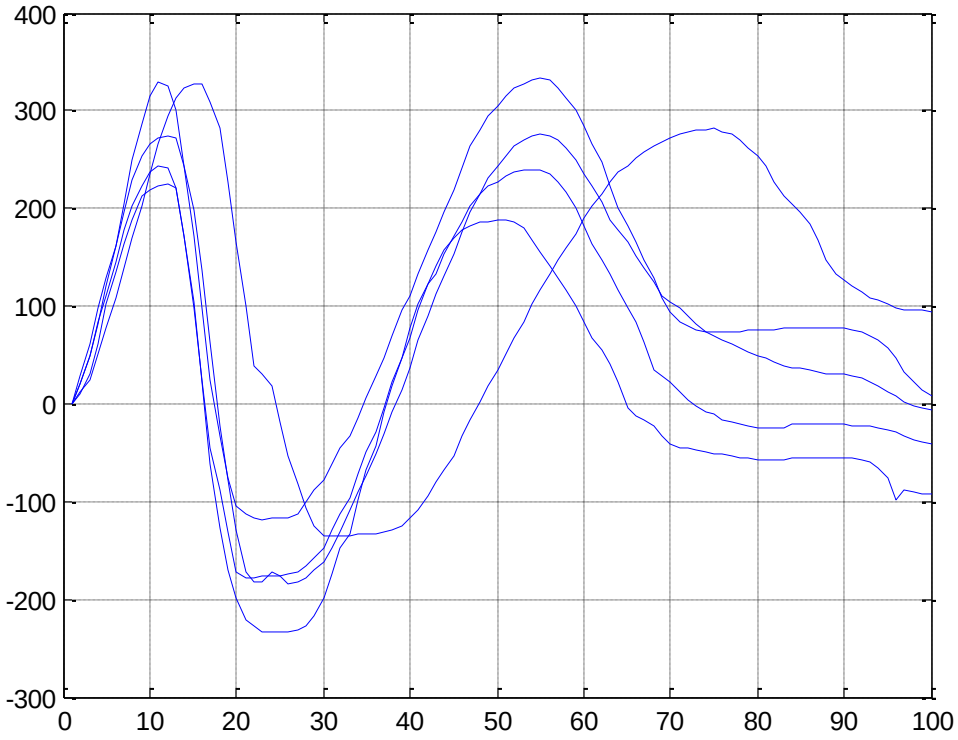


Figure 96 Normalized “Yo-Yo Motion” Behaviors

Table 18 Similarity between the Cyclic “Yo-Yo Motion” Behaviors and the Generalized “Yo-Yo” Motion Behavior

	DTW Distance	Ratio
“Yo-Yo Motion” Trajectory 1	1.74 cm	0.18%
“Yo-Yo Motion” Trajectory 2	0.26 cm	0.03%
“Yo-Yo Motion” Trajectory 3	0.21 cm	0.02%
“Yo-Yo Motion” Trajectory 4	1.34 cm	0.14%
“Yo-Yo Motion” Trajectory 5	1.60 cm	0.16%

The DTW distances between these demonstrated “Yo-Yo Motion” behaviors and the generalized “Yo-Yo Motion” trajectory are displayed in Table 18.

From Table 18, all the rations are smaller than 2%. Then we can conclude that these cyclic “Yo-Yo Motion” behaviors are similar to the generalized “Yo-Yo Motion” behavior, which means that these behaviors belong to the same type of behaviors and can be generalized.

Figure 97 displays the generalized results from these cyclic “Yo-Yo Motion” behaviors.

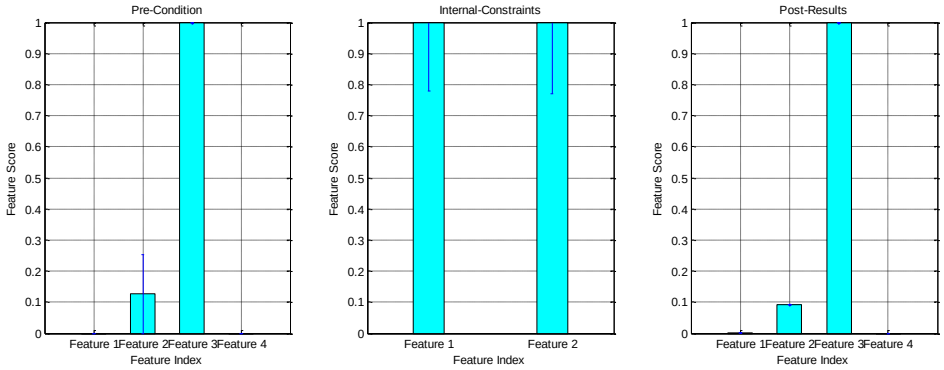


Figure 97 Generalization Results of Cyclic “Yo-Yo Motion” Behaviors

From Figure 97, the generalized results of the cyclic “Yo-Yo Motion” Behaviors are to generated similar dynamics of the motion trajectories while keeping the Yo-Yo in hand. So we can conclude that ISAC can learn cyclic behaviors using the methods described in this dissertation.

Experiment 3A

In Experiment 3A, ISAC was asked to push the box to its right. The locations of the box lie within or out of the working space of the arms. ISAC first tried to use the right arm to push the box. And at some locations, before the end-effector on the right arm of ISAC reaches the pushing point, the arm or the end-effector may collide with the box. So ISAC uses the cognitive control mechanism to switch strategy to use the left arm to push the box. If ISAC found that it cannot complete the task using the left arm, it refuses to complete the task and display a message on the screen.

Table 19 Simulation Results of Experiment 3A

Location Number	Selected Strategy	Feasible/Infeasible	IRS Correctness
1	Right Arm	Feasible	Correct
2	Left Arm	Feasible	Correct
3	Left Arm	Feasible	Correct
4	Left Arm	Feasible	Correct
5	Refusal	Infeasible	Correct
6	Right Arm	Feasible	Correct
7	Right Arm	Feasible	Correct
8	Left Arm	Feasible	Correct
9	Left Arm	Feasible	Correct
10	Refusal	Infeasible	Correct

In Experiment 3A, the object was placed at 10 different locations. ISAC first tried to push the object using its right arm. If the evaluation failed, ISAC switched to the left arm. If the evaluation still failed, ISAC refused to complete the task.

Table 19 summarizes the evaluation results of Experiment 3A.

At location 5 and 10, the object is out of the working space of ISAC. ISAC found that it cannot push the box using either of its arms. So it displayed a message on the screen and refused to do. The rate of feasibility in Experiment 3A is 80%.

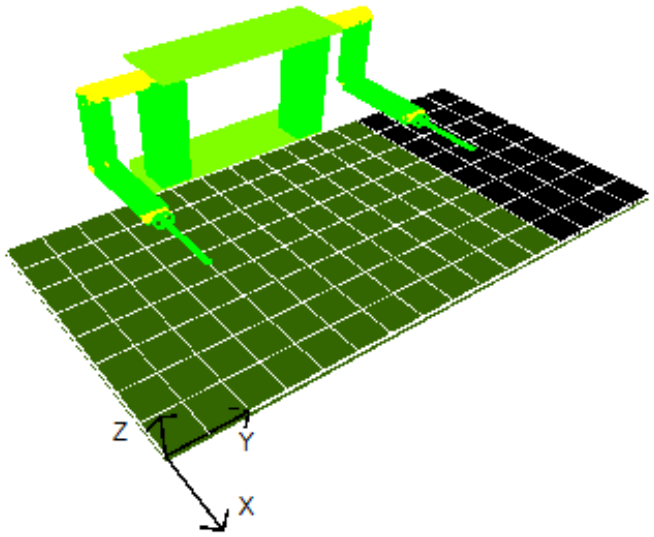


Figure 98 Successful Pushing Area in Experiment 3A

When the box is placed in the black area, the size of which is approximately 20% of the table, ISAC cannot push it to the right

At all locations, ISAC made a correct decision, and chose a suitable arm to push the cylinder. The success rate of the evaluation is 100%. When the obstacle is used in Experiment 3B, the success rate of the evaluation decreases.

The actual time to run the key modules in our system was measured and displayed in Table 20. The abbreviations in the table are designed as: Behavior Sequence

Generation (BSG), Behavior Generation for the Right Arm (BGRA), Behavior Generation for the Right Arm (BGLA), IRS for the Right Arm (IRSRA), and IRS for the Left Arm (IRSLA), Standard Deviation (STD).

Table 20 Running Time of Key Components in Experiment 3A

	BSG	BGRA	BGLA	IRSRA	IRSLA
Experiment 3A-1	0.0045	11.1654	11.2266	0.8325	0.0004
Experiment 3A-2	0.0049	11.7184	11.4417	0.0849	1.2241
Experiment 3A-3	0.0057	10.9466	11.5291	0.0903	1.2020
Experiment 3A-4	0.0053	11.0135	11.1674	0.1289	0.1149
Experiment 3A-5	0.0057	11.0057	10.9236	0.1691	0.1531
Experiment 3A-6	0.0135	10.9445	11.0336	0.7955	0.0004
Experiment 3A-7	0.0049	12.1034	10.8054	0.8189	0.0004
Experiment 3A-8	0.0049	10.9573	10.9293	0.4039	1.1572
Experiment 3A-9	0.0049	11.1231	10.7832	0.2339	1.0952
Experiment 3A-10	0.0045	10.8374	11.0020	0.1777	0.3587
Average	0.0059	11.1815	11.0842	0.3736	0.7579
STD	0.0027	0.4057	0.2539	0.3182	0.5206
Percentage	0.03%	48.25%	47.83%	1.61%	2.29%

Unit: millisecond (ms)

As shown in Table 20, the average running time for generating the behavior sequence is 0.0059ms. There is a large value in Experiment 3A-6 comparing to the values in other experiments. This is because that Windows is a time-sharing operating system. The operating system kernel may interrupt the current running threads to perform some kernel tasks, which increases the time needed to complete the running operation. The

average running time for generating behaviors for the right arm and the left arm is 11.1815ms and 11.0842ms respectively.

The average running time of evaluating the generated behaviors for the right arm and for the left arm is 0.3736ms and 0.7579ms respectively. In Experiment 3A-1, 3A-6, and 3A-7, ISAC only needed to evaluate the behaviors for the right arm and did not evaluate the behaviors for the left arm, so the running time for evaluating the left arm is very small: 0.0004 ms. In Experiment 3A-2, 3A-3, 3A-4, 3A-5, 3A-8, 3A-9, and 3A-10, ISAC found that it cannot complete the task using the right arm, so it switched the strategy to use the left arm. In these experiments, the running time of evaluating the behaviors for the left arm increases.

In some experiments, ISAC did not need to evaluate the overall motion trajectory. It stops evaluation when it found that collision happens, the via point on the trajectory is out of the working space, or the behavior is not completed. In Experiment 3A-1, 3A-6, and 3A-7, ISAC used the right arm to complete the task that means ISAC evaluated the overall motion trajectory of the right arm, and the average time is 0.8156ms. In Experiment 3A-2, 3A-3, 3A-4, 3A-8, and 3A-9, ISAC used the left arm to complete the task that means ISAC evaluated the overall motion trajectory of the left arm, and the average time is 0.9587ms.

The running time of generating behavior sequence is 0.03% of the overall running time of the key components. 48.25% and 47.83% of the overall running time is used to generate motion trajectories for the right arm and the left arm respectively. The evaluation time is 1.61% and 2.29% of the overall running time.

From the above quantitative evaluation results in Experiment 3A, the running time of each key components in our system is very small and can satisfy the requirements of generating behaviors and evaluating behaviors when there is no obstacle placed in the environment.

Experiment 3B

In Experiment 3B, ISAC was asked to push the box to the right and avoid the obstacle on the table. The locations of the box lie within or out of the working space of the arms. And at some locations, before the end-effector on the right arm of ISAC reaches the pushing point, the arm or the end-effector may collide with the box. So ISAC used the cognitive control mechanism to switch strategies to use the left arm to push the box. The evaluation results for all the experiments in Experiment 3B are included in Appendix C. Table 21 summarizes the evaluation results.

The average rate of feasibility is 77%. The highest rate of the feasibility is 93.75% for Experiment 3B-6. The reason is that the object is placed in front of the right arm of ISAC, which is close to the home position of the right arm. Additionally, the manipulation point is also within the working space of the left arm. So ISAC has more choices to complete the task. The lowest rate of feasibility is 68.75% in Experiment 3B-3. The reason is that the object is out of the working space of the right arm, and if the obstacle is placed at Location 3B-3a, ISAC cannot push the box using its left arm.

The highest success rate of the IRS evaluation is 100% for Experiment 3B-6. In Experiment 3B-6, as stated in the last paragraph, ISAC has more choices to complete the task. Also, the obstacle avoidance module also did not generate wrong collision information for IRS to evaluate. The lowest success rate of IRS Evaluation is 81.25% in

Experiment 3B-8. The reason is that the obstacle avoidance module generates wrong information to prevent the end-effector moving to the manipulation point.

Table 21 Evaluation Results of Experiment 3B

	Feasibility	Success Rate of the IRS
Experiment 3B-1	75%	87.5%
Experiment 3B-2	75%	87.5%
Experiment 3B-3	68.75%	87.5%
Experiment 3B-4	75%	87.5%
Experiment 3B-5	75%	93.75%
Experiment 3B-6	93.75%	100%
Experiment 3B-7	81.25%	93.75%
Experiment 3B-8	75%	81.25%
Average	77%	90%

Table 22 Success Rates of Experiment 3B with Different Obstacles

	Rate of Feasibility	Success Rate of the IRS
Obstacle 1 (1cm × 1cm × 20cm)	93.75%	93.75%
Obstacle 2 (5cm × 5cm × 20cm)	93.75%	93.75%
Obstacle 3 (10cm × 10cm × 20cm)	78.125%	90.625%
Obstacle 4 (20cm × 20cm × 20cm)	43.75%	87.5%

Table 22 displays the rate of feasibility and the evaluation when different sizes of obstacles are placed on the table.

In Table 22, the rate of feasibility decreases when the size of the obstacle increases. When the obstacle size increases to 20cm × 20cm × 20cm, the success rate is the lowest. This is reasonable because large obstacle can greatly prevent ISAC from pushing the box. The success rate of the IRS evaluation decreases when the size of the obstacle increases. The reason is that when the size of the obstacle becomes larger, the obstacle avoidance requires the end-effector moves in a very small area. This makes the evaluation more difficult.

Overall Analysis of Cognitive Control

Table 23 displays the rate of feasibility and the success rate of the IRS evaluation.

Table 23 Success Rates of Experiment 3A and 3B

	Rate of Feasibility	Success Rate of the IRS
Experiment 3A	80%	100%
Experiment 3B	77%	90%
Average	77.64%	90.97%

The average rate of feasibility is 77.64%. The average success rate of the IRS evaluation is 90.97%.

Table 24 displays the average running time for Experiment 3A and 3B

Table 24 Running Time of Key Components in Experiment 3A and 3B

	BSG	BGRA	BGLA	IRSRA	IRSLA
Experiment 3A	0.0059	11.1815	11.0842	0.3736	0.5306
Experiment 3B-1	0.0050	11.3215	10.9597	0.7758	0.4358
Experiment 3B-2	0.0050	10.7149	13.6738	0.1618	0.9363
Experiment 3B-3	0.0055	11.6473	11.0467	0.2115	0.8738
Experiment 3B-4	0.0052	10.8341	11.4375	0.2401	0.9526
Experiment 3B-5	0.0049	11.6536	10.7647	0.8069	0.2456
Experiment 3B-6	0.0054	15.9670	11.9227	0.8134	1.0514
Experiment 3B-7	0.0053	11.0320	11.9550	0.4772	1.1315
Experiment 3B-8	0.0052	10.8341	11.4375	0.2401	0.9526
Average	0.0053	11.6873	11.5869	0.4556	0.7900
Standard Deviation	0.0003	1.6409	0.8847	0.2738	0.3073
Percentage	0.02%	47.65%	47.24%	1.86%	3.22%

In Table 24, the overall average running time for all the experiments is:

Behavior Sequence Generation: 0.0053ms

Behavior Generation for the Right Arm: 11.6873ms

Behavior Generation for the Left Arm: 11.5869ms

IRS evaluation for the Right Arm: 0.4556ms

IRS evaluation for the Left Arm: 0.7900ms

From the above quantitative evaluation results, the running time of each key components in our system is very small and can satisfy the requirements of generating behaviors and evaluating behaviors in dynamic environment.

The running time of generating behavior sequence is 0.02% of the overall running time of the key components. 47.65% and 47.24% of the overall running time is used to generate motion trajectories for the right arm and the left arm respectively. The evaluation time is 1.86% and 3.22% of the overall running time.

Several conclusions can be obtained from the above analysis

1. The success rate of the IRS evaluation is higher than 90%.
2. If we want to increase the success rate of the task, we should limit the size of the obstacle. In order to keep the success rate of the task, the size of the obstacle should be smaller than 10cm × 10cm × 20cm
3. If the size of the obstacle is fixed and larger than 10cm × 10cm × 20cm, we need to change the motion trajectory generation methods in this system, e.g., improve the potential field design of the obstacles.

System Performance

The input of the system is the behavior demonstrations. Given a task, the output is the generated motion trajectories. .

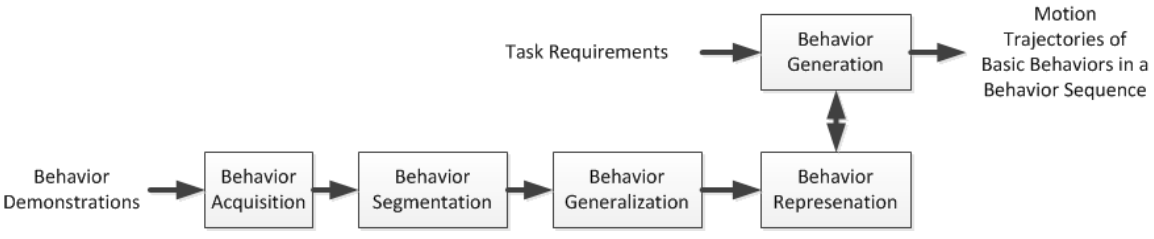


Figure 99 Imitation Learning Framework

Performances of individual components can affect the overall system performance. In this section, unsupervised teaching and physical coaching are used to as the demonstration methods for ISAC to learn demonstrated behaviors. We will analyze how different demonstration methods affect

(1) the output of the behavior generalization and

(2) the overall output of the system.

The output of the behavior generalization is:

- Score of Feature 1 of Pre-Condition
- Score of Feature 2 of Pre-Condition
- Score of Feature 3 of Pre-Condition
- Score of Feature 4 of Pre-Condition
- Score of Feature 1 of Internal Constraint
- Score of Feature 2 of Internal Constraint
- Score of Feature 1 of Post-Result
- Score of Feature 2 of Post-Result
- Score of Feature 3 of Post-Result
- Score of Feature 4 of Post-Result

The output of the Generated Motion Trajectories (determined stored common features generalized behaviors). Because the generated motion trajectories are related to the behavior generation methods which are determined by the stored common features generalized behaviors in the LTM, we can analyze the stored common features of the learned behavior instead of analyzing the motion trajectories.

Two behaviors are demonstrated for ISAC to learn: the “Reaching” and “Pushing”.

System Performance between the Input and the Output of the Behavior Generalization Module

--*“Reaching”*

Figure 100 displays the experimental setup of the learning stage. In the left figure, a human teacher demonstrates how to reach an object on the table and ISAC uses Kinect to observe the demonstration. In the right figure, the human teacher demonstrates how to reach an object by manually moving the arm of ISAC.

Figure 100 Learning from Observation VS Learning from Physical Coaching for “Reaching”

Figure 101 displays the generalization results of the “Reaching” behavior learned from the physical coaching.

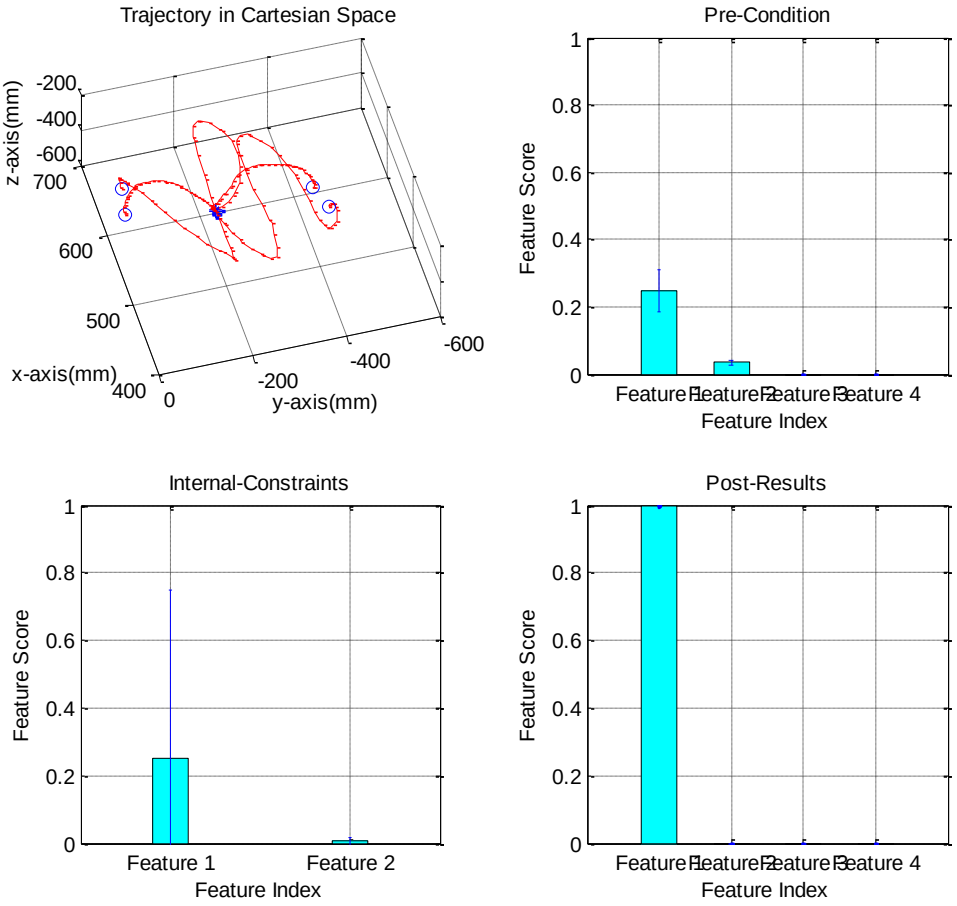


Figure 101 Generalization Results of the "Reaching" Behavior from Physical Coaching

In Figure 101, the most common feature is to minimize the distance between the end-effector and the target object at the end of the motion trajectory. Table 25 displays the comparison of the generalization results.

In Table 25, the scores of Feature 1 of the Post-Results are 0.9559, 0.9476, and 0.9989. The highest score comes from the generalization results of the learning from Physical coaching. This reflects that physical coaching has better performance to find the most common feature of the "Reaching" behavior.

Table 25 Comparison of Generalization Results of the "Reaching" Behavior

	Unsupervised Left Arm	Unsupervised Right Arm	Physical Coaching
Feature 1 of Pre-Condition	0.0442	0.0475	0.2477
Feature 2 of Pre-Condition	0.0717	0.0807	0.0337
Feature 3 of Pre-Condition	0	0	0
Feature 4 of Pre-Condition	0	0	0
Feature 1 of Internal Constraints	0.1250	0.1250	0.2498
Feature 2 of Internal Constraints	0.0748	0.0520	0.0058
Feature 1 of Post-Results	0.9559	0.9476	0.9989
Feature 2 of Post-Results	0.0406	0.0360	0
Feature 3 of Post-Results	0	0	0
Feature 4 of Post-Results	0	0	0

--*"Pushing Left" and "Pushing Right"*

Figure 102 Learning from Observation VS Learning from Physical Reaching for "Pushing"

Figure 103 displays the generalization results of the "Pushing Left" behavior learned from the physical coaching.

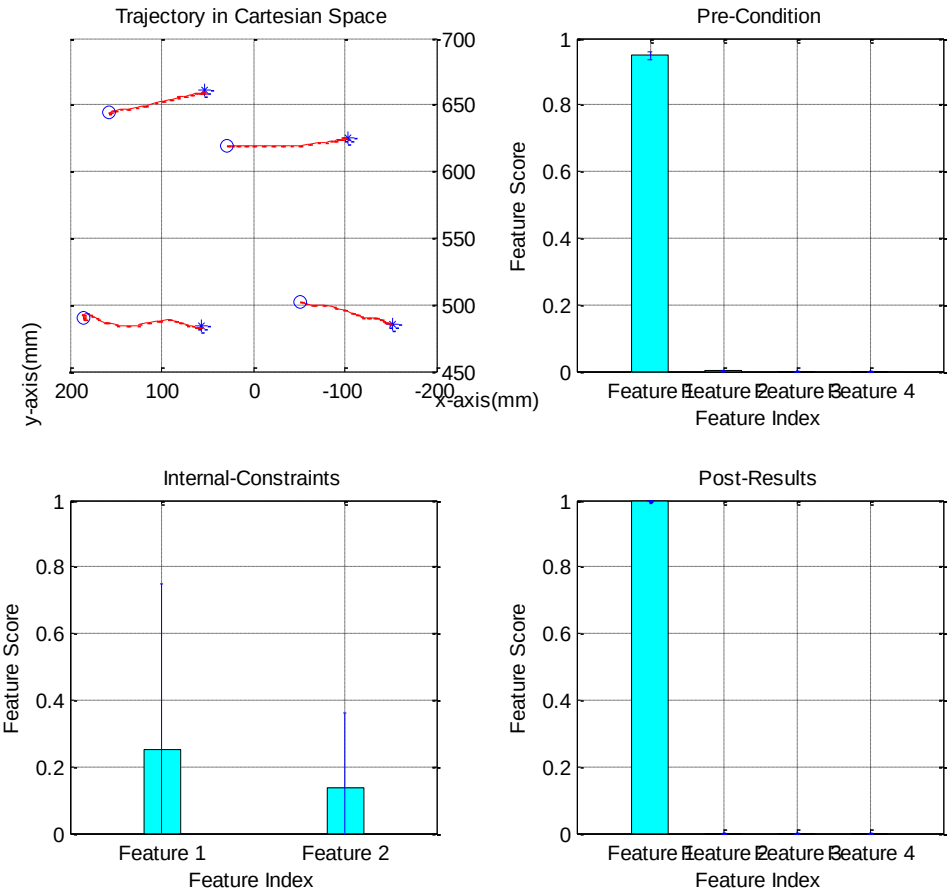


Figure 103 Generalization Results of the "Pushing Left" Behavior from Physical Coaching

Figure 104 displays the generalization results of the "Pushing Right" behavior learned from the physical coaching.

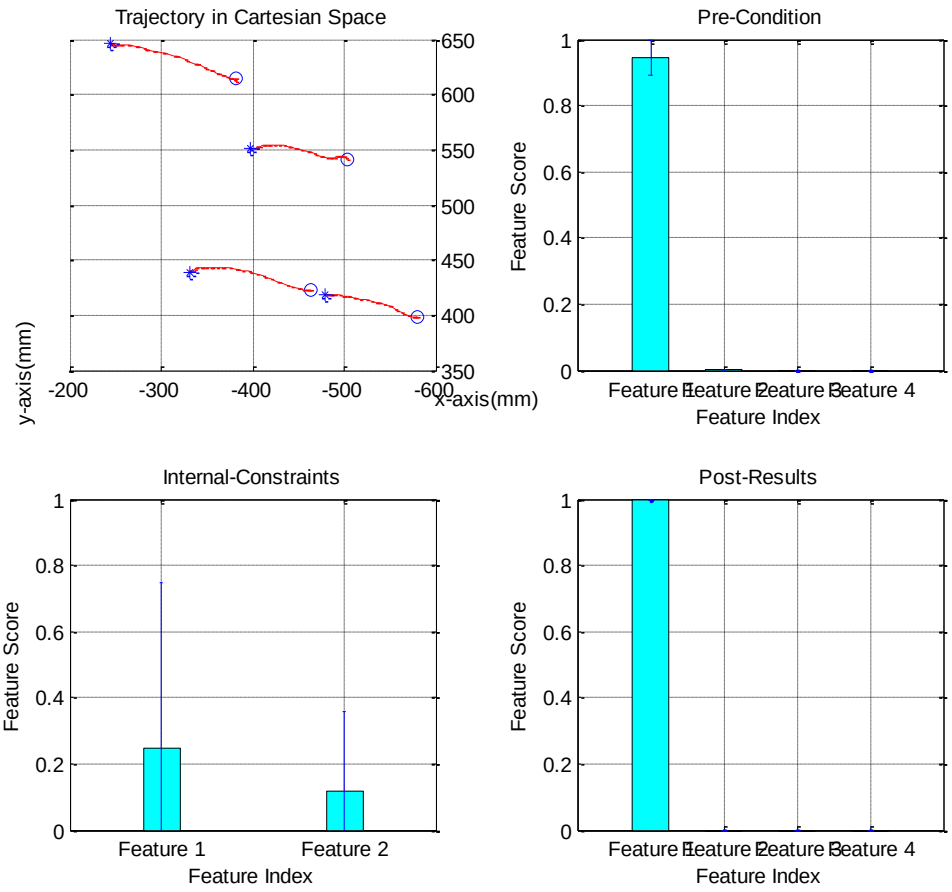


Figure 104 Generalization Results of the "Reaching Right" Behavior from Physical Coaching

In Figure 103 and Figure 104, the most common feature is to minimize the distance between the end-effector and the target object at the beginning of the motion trajectory and to minimize the distance between the end-effector and the target position at the end of the motion trajectory. Table 25 displays the comparison of the generalization results.

Table 26 Comparison of Generalization Results of the "Pushing" Behavior

	Unsupervised Teaching		Physical Coaching	
	PushingLeft	PushingRight	PushingLeft	PushingRight
Feature 1 of Pre-Conditions	0.8514	0.7450	0.9475	0.9444
Feature 2 of Pre-Conditions	0	0	0	0
Feature 3 of Pre-Conditions	0	0	0	0
Feature 4 of Pre-Conditions	0	0	0	0
Feature 1 of Internal Constraints	0.1111	0.1111	0.2496	0.2498
Feature 2 of Internal Constraints	0.0238	0.1897	0.1372	0.1158
Feature 1 of Post-Results	0.8965	0.9909	0.9984	0.9989
Feature 2 of Post-Results	0	0	0	0
Feature 3 of Post-Results	0	0	0	0
Feature 4 of Post-Results	0	0	0	0

Feature 1 of Pre-Conditions and Feature 1 of Post-Results are the common features of the "Pushing Left" and "Pushing Right" behaviors. In Table 26, the scores for these features from the physical coaching are higher than the scores from the observations. This means the physical coaching can improve the performance of finding the most common feature of the "Pushing Left" and "Pushing Right" Behaviors.

Based on the above analysis, different behavior acquisition methods can affect the performance of the learning results of the system. In our testing, it is displayed that the methods of Physical Coaching can improve the behavior generalization results of the system.

System Performance between the Input and the Overall Output of the System

Because the generated motion trajectory is related to the behavior generation methods which are determined by the stored common features generalized behaviors in the LTM, we can analyze the stored common features of the learned behavior instead of analyzing the motion trajectories.

Table 27 Comparison of Stored Features of the "Reaching" and "Pushing" Behaviors

	Reaching		Pushing	
	Learning from Observation	Physical Coaching	Learning from Observation	Physical Coaching
Stored Common Feature	{0,0,1}	{0,0,1}	{1,0,1}	{1,0,1}

The generated motion trajectory is related to the generalized behaviors stored in the LTM. In the above analysis, using different demonstration methods, the generalization results are the same. So the demonstration methods do not affect the overall output of the system.

Summary

The simulation and experimental results included in Chapter IV demonstrate that ISAC can use this integrated system to parse the speech command, check the LTM, switch tasks between learning and generation, observe demonstrations, generalize behaviors, generate behavior sequences, generate similar motion trajectories, and switch strategies to complete tasks.

In summary, we can conclude that the designed system satisfy the requirements of imitation learning and the cognitive control. It enables ISAC to generate similar motion trajectories and switch strategies to complete tasks.

CHAPTER VI

CONCLUSION AND FUTURE WORK

Conclusion

This dissertation investigated how imitation learning and cognitive control can be integrated into a humanoid robot. Through simple behavior learning, the integrated system was shown to learn new behaviors and perform a new task using behavior graph and strategy switching. The imitation learning framework was divided into: (i) Behavior Acquisition, (ii) Behavior Segmentation, (iii) Behavior Generalization, (iv) Behavior Representation, and (v) Behavior Generation. The cognitive control framework was designed based on the robotic cognitive architecture developed in our lab, which (i) uses sensors to collect environmental information, demonstrated information, and speech command, (ii) generalizes behaviors using the CEA, (iii) stores basic behaviors in the LTM, (iv) generates behavior sequences to complete tasks, (v) evaluates generated behavior sequences using IRS, and (vi) sends the generated motion trajectories to the actuators. The execution results will be monitored by human in case robots need more assistance or instructions. In our system, human can give instructions such as learning new behaviors, re-evaluating behavior sequences, etc., after ISAC searches the behavior library to find whether it has required behavior, executes the generated behavior sequence, and evaluates the behavior sequence in the IRS. After observing the response of ISAC and execution of evaluation results, human teachers can give ISAC a command to learn the required behavior, confirm the evaluation results, etc. For example, in Experiment 1,

ISAC was asked to push the object without grasping and found that it has not learned the required “Reaching” and “Pushing” behaviors. So the human teacher will tell ISAC that he/she will demonstrate ISAC these required behaviors as shown in Figure 105.

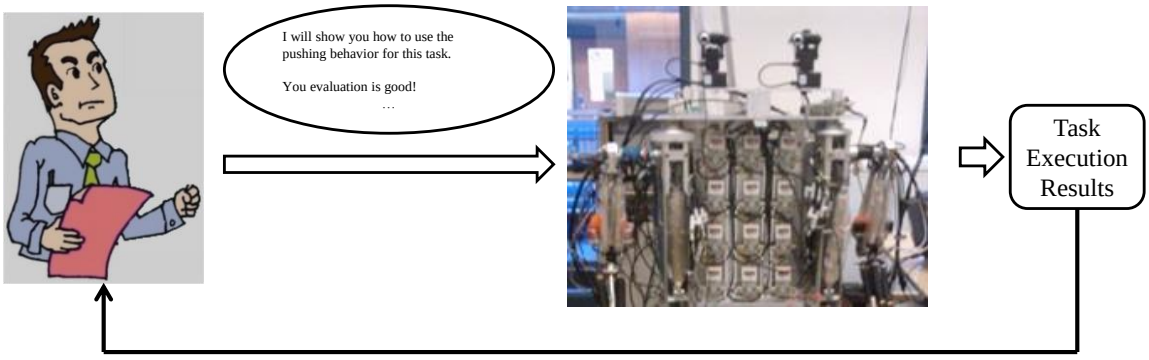


Figure 105 Guidance from Human

Three types of experiments were designed for ISAC to validate this system. One experiment was carried out on ISAC, and two other experiments were carried out in a simulation environment. The experimental results demonstrated that ISAC can use this system to complete simple graspless manipulation tasks according to the speech commands. Simulation-based cognitive control experiments demonstrate how the Central Executive Agent (CEA) switches strategies to adaptively complete tasks. The quantitative evaluation results in Chapter V demonstrated that the system satisfy the requirements for both imitation learning and cognitive control.

Future Work

The current designed system enables the robot to parse the speech command, check the LTM, switch tasks between learning and generation, observe demonstrations,

generalize behaviors, generate behavior sequences, generate similar motion trajectories, and switch strategies to complete tasks using one arm.

There are several areas which will enhance the performance of this system. They are:

1. Dual Arm Control and Behavior Generation

   In the experiments of this dissertation, ISAC only needs to use one arm to complete the required tasks. However, many tasks require ISAC to use both arms. For example, for graspless manipulation, a pivoting task [Aiyama et al., 1993], in which two robot grippers maneuver an object as if making the object "walk" as shown in Figure 106. In an assembly task, a robot needs to hold one piece using one hand and insert/put the other piece to the correction position. If a robot can use two hands to complete tasks, that will make the robot more useful in working environments, living environments, etc. In Chapter IV, we have proposed a prototype method of evaluate the generated behavior sequence using two arms. This method must be expanded to more real-life situations.

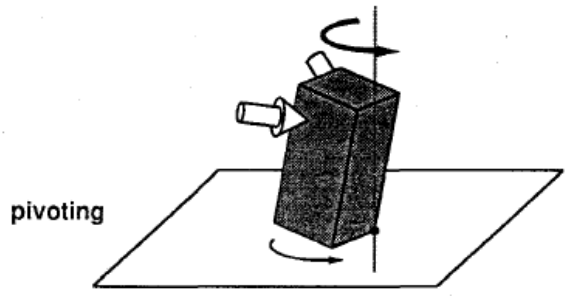


Figure 106 Pivoting [Aiyama et al., 1993]

However, in order to use two arms, the kinodynamics features should be considered in behavior generation of robots, e.g., force, torques, when generating behaviors to complete tasks. For example, in Experiment 1, ISAC is asked to push a box. The weight of the box, the required force on the end-effector, and the required torques on the joints should be taken into consideration in order to increase the probability of pushing the box successfully to designated areas.

2. Hierarchy Behavior Generation

A major limitation of this behavior graph-based method is: the binary-based preconditions, internal constraints, and post results-based behavior graph is too simplistic to be applied to some real-world applications. As robots keep learning, the number of the behaviors becomes larger and larger. It becomes more and more difficult for robots to handle the behavior graph when the behavior graph becomes more and more complex. In order to reduce the complexity of the behavior graph, we need to improve the simple binary-based behavior graph-based methods in this dissertation.

In our system, behaviors are treated equally in the behavior graph. All the behaviors are considered has same weights and the transitions among them are non-weighted. However, in some real-world applications, behaviors are often organized in a hierarchy way. This type of hierarchy methods simplifies the overall generation process and provides a robust method for solving the requirements of generating complex behavior sequences in complex task-

relevant situations. A possible solution could be to divide behavior graphs into several levels. In the generation stage, robots generated behavior sequences by assembling behavior sequences from different levels of behavior graphs.

Additionally, using this method, errors, which are generated by the hardware of ISAC, could be confined in the lower level behavior graphs. Probabilistic methods can be applied to lower level behavior graphs to overcome the errors. In higher level behavior graphs, error could not be taken into consideration.

APPENDIX

A. Hand-Tracking and Object-Tracking Using Kinect

Hand-Tracking

We define the observed positions of a wrist, a related hand, and a related shoulder of a human are:

$$p_{wrist\_image} = (p_{wirstX}, p_{wirstY}, p_{wirstZ})^T \quad (1)$$

$$p_{hand\_image} = (p_{handX}, p_{handY}, p_{handZ})^T \quad (2)$$

$$p_{shoulder\_image} = (p_{shoulderX}, p_{shoulderY}, p_{shoulderZ})^T \quad (3)$$

The coordinates of a camera image is designed as displayed in Figure 107.

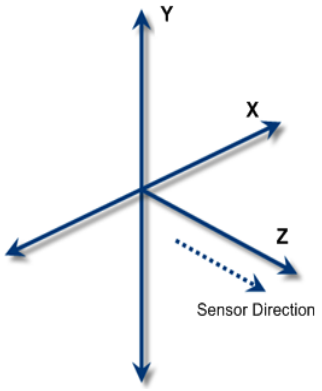


Figure 107 Coordinates of the Image Pixels

The relative positions of the wrist and the hand in the coordinates of a camera image are:

$$p_{wrist-shoulder\_image} = p_{wrist} - p_{shoulder} \quad (4)$$

$$p_{hand-shoulder\_image} = p_{hand} - p_{shoulder} \quad (5)$$

After this transformation, the origin of the coordinates has been moved to the shoulder.

Then we can convert the positions from the coordinates of a camera image to a human body.

The transformation matrices are:

$$T_{image}^{body} = \begin{vmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \tag{6}$$

Therefore, the positions of the wrist and the hand in the coordinates of a human body are:

$$p_{wrist} = T_{image}^{body} p_{wrist-shoulder\_image} \tag{7}$$

$$p_{hand} = T_{image}^{body} p_{hand-shoulder\_image} \tag{8}$$

The observed motion information in a task-space can be represented as

$$task\_position = p_{wrist} \tag{9}$$

$$task\_orientation = p_{hand-shoulder\_image} - p_{wrist-shoulder\_image} \tag{10}$$

$P_s = \{P_v, t\}$, which is an $N \times 4$ matrix, records the 3-dimensional position values, P_v, of the hand of a human teacher and the temporal information t on the sampling points.

Object-Tracking

The positions and sizes of a target object and obstacles are extracted from camera images in the color space. Obtained raw data of camera images are RGB-based and need to be converted to HSV-based to avoid the noises generated by light. In order to simplify our work, we used a yellow cube as a target object and blue cubes as obstacles. This is enough for our designed experiments. By extracting all yellow or blue pixels from the HSV images, the center of the target object and each obstacle can be computed as:

$$Center(i) = \frac{\sum_{j=0}^{n} p_j}{n} \tag{11}$$

where p_j represents the position value of each pixel in the Cartesian space:

$$p_j = \left(p_{j-x}, p_{j-y}, p_{j-z}\right)^T \tag{12}$$

The edges of target object and cubes are recorded by finding the maximum values and minimum values on X, Y, and Z direction in the Cartesian space.

$$XMin(j) = min\left(p_{j-x}\right) \tag{13}$$

$$XMax(j) = max\left(p_{j-x}\right) \tag{14}$$

$$YMin(j) = min\left(p_{j-y}\right) \tag{15}$$

$$YMax(j) = max\left(p_{j-y}\right) \tag{16}$$

$$ZMin(j) = min\left(p_{j-z}\right) \tag{17}$$

$$ZMax(j) = max\left(p_{j-z}\right) \tag{18}$$

Then the recorded states for d^{th} demonstration are:

$S^d = \{Center(1), Center(2), \dots, Center(M), XMin(j), XMax(j), YMin(j), YMax(j), ZMin(j), ZMax(j)\}$

and each element is a vector with length of the sampling time.

B. A Potential Field Method-Based Extension of the Dynamic Movement Primitive Algorithm for Imitation Learning with Obstacle Avoidance

In Figure 108, Figure 109, and Figure 110, the black circle is the obstacle placed in the environment.

In Figure 108, when $\vec{y}(k)$ is generated by the DMP method, it is in the impedance area (orange area outside black obstacle) around the obstacle and the obstacle is between $\vec{y}(k)$ and the goal state $\vec{g}$. Then, the goal state g is moved to a virtual goal state $\vec{g}_{temp}(k+1)$.

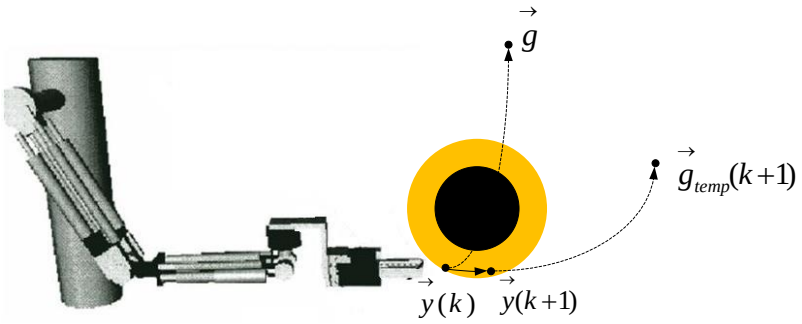


Figure 108 Calculation of the point $\vec{y}(k)$ in the impedance area around the obstacle using the improved DMP algorithm

In Figure 109, $\vec{y}(k+1)$ can be generated by using the DMP method with the new virtual goal state $\vec{g}_{temp}(k+1)$. $\vec{y}(k+1)$ is still in the impedance area around the obstacle and the obstacle still locates between $\vec{y}(k+1)$and the goal state $\vec{g}$, then the goal state $\vec{g}$ is modified into $\vec{g}_{temp}(k+2)$.

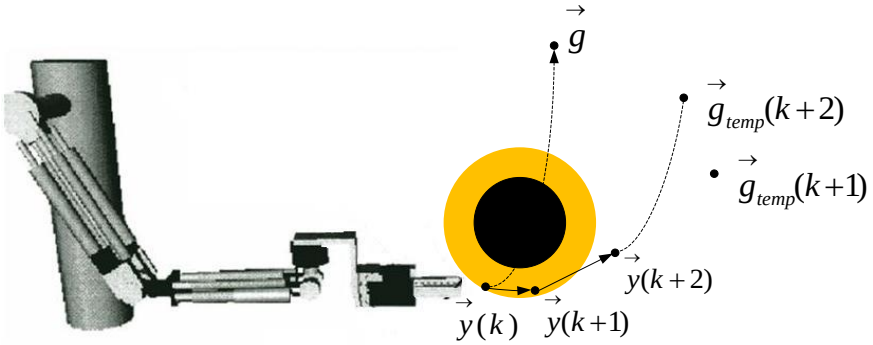


Figure 109 Calculation of the point $\vec{y}(k+1)$ in the impedance area around the obstacle using the improved DMP algorithm

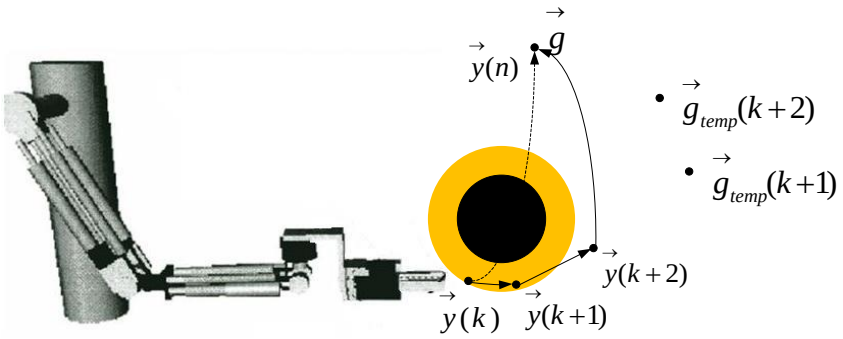


Figure 110 Calculation of the points not in the impedance area around the obstacle using the improved DMP algorithm

In Figure 110, $\vec{y}(k+2)$ can be generated using the DMP method with the new virtual state $\vec{g}_{temp}(k+2)$. Now, $\vec{y}(k+2)$ is not in the impedance area around the obstacle and the obstacle is not between $\vec{y}(k+2)$ and the goal state $\vec{g}$, then the goal state $\vec{g}$ will not be moved. Other points $\{\vec{y}(k+2), \vec{y}(k+3), \vec{y}(k+4), \dots, \vec{y}(n)\}$ will be generated using the original DMP method until $\vec{y}(n) = \vec{g}$ to achieve the initial goal state.

The impedance factor $\vec{g}_{impedance}(k+1)$ is generated by the virtual impedance force in the impedance field as shown in Figure 109. The center of the obstacle is $\vec{o}$, and the radius of the obstacle is r. d is the distance between $\vec{y}(k)$ and $\vec{o}$,

$$d = \|\vec{y}(k) - \vec{o}\| \tag{19}$$

When $\vec{y}(k)$ is in the obstacle area and the obstacle is between $\vec{y}(k)$ and $\vec{g}$,

$$|d - r| < ImpedanceArea \tag{20}$$

$$distance\left(\vec{o}, \left(\vec{g} - \vec{y}(k)\right)\right) < ImpedanceArea \tag{21}$$

The virtual impedance force is calculated by the following equations. α and β are constants.

$$\vec{F}_{impedance}(k) = \vec{F}_{impedance}(k)' \pm \vec{F}_{impedance}(k) \perp \tag{22}$$

$$\vec{F}_{impedance}(k)' = \alpha \frac{1}{(d-r)^2} \tag{23}$$

$$\vec{F}_{impedance}(k)' = \beta \frac{1}{(d-r)^2} \tag{24}$$

The impedance factor $\vec{g}_{impedance}(k+1)$ is proportional to $\vec{F}_{impedance}(k)$ obtained from equation (25).

$$\vec{g}_{impedance}(k+1) = \gamma \vec{F}_{impedance}(k) \tag{25}$$

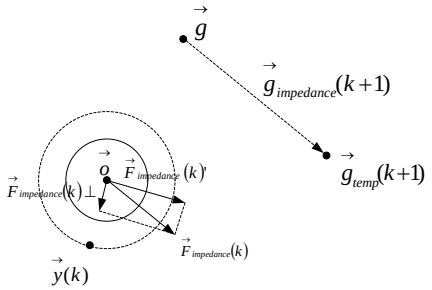


Figure 111 Calculation of $\vec{g}_{temp}$

Intuitively, the smaller the distance between $\vec{y}(k)$ and the surface of the obstacle is, the larger the $\vec{F}_{impedance}(k)$ is. When $\vec{y}(k)$ is around the obstacle, the virtual goal state $\vec{g}_{temp}(k+1)$ is moved proportionally to $\vec{F}_{impedance}(k)$. This virtual goal state changes the trajectory around the obstacle, but still keeps the dynamics of the generated trajectory similar to the demonstration because the generated trajectory is calculated by the DMP method with the original demonstration.

$$\tau \dot{\vec{z}}(k) = \alpha_z \left(\beta_z \left(\vec{g}_{impedance}(k+1) - \vec{y}(k) \right) - \vec{z}(k) \right) \tag{26}$$

$$\tau \dot{\vec{y}}(k) = \vec{z}(k) + f(k) \tag{27}$$

$$\vec{y}(k+1) = \vec{y}(k) + \dot{\vec{y}}(k) \tag{28}$$

$$\vec{z}(k+1) = \vec{z}(k) + \dot{\vec{z}}(k) \tag{29}$$

C. Simulation Results and Generated Motion Trajectories of Experiment 3

Experiment 3A

ISAC pushes the box to the right using its right arm in Experiment 3A-1.

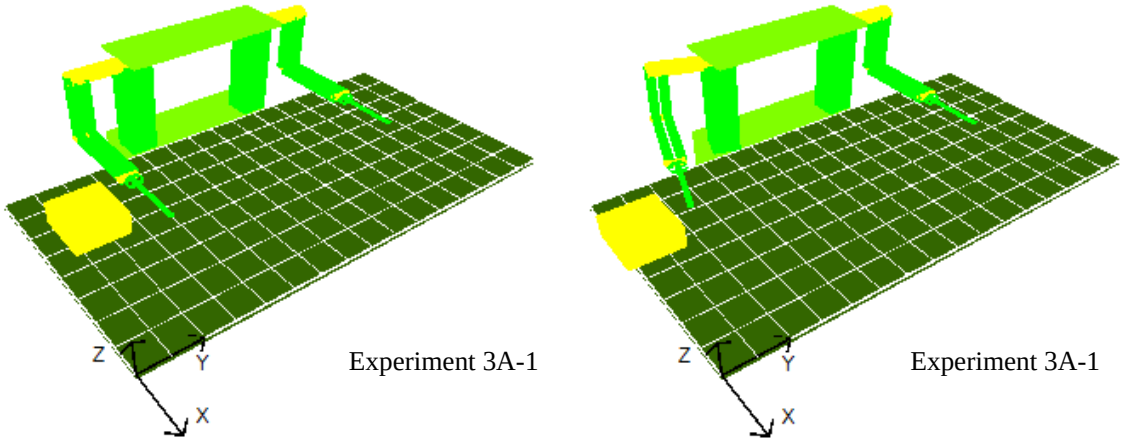


Figure 112 Simulation Results of Experiment 3A-1

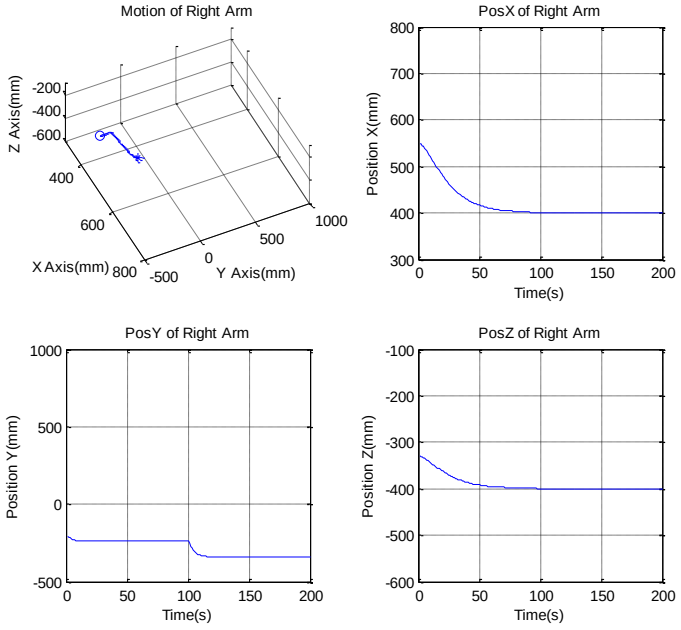


Figure 113 Generated Motion Trajectories of the Right Arm in Experiment 3A-1

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3A-2. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

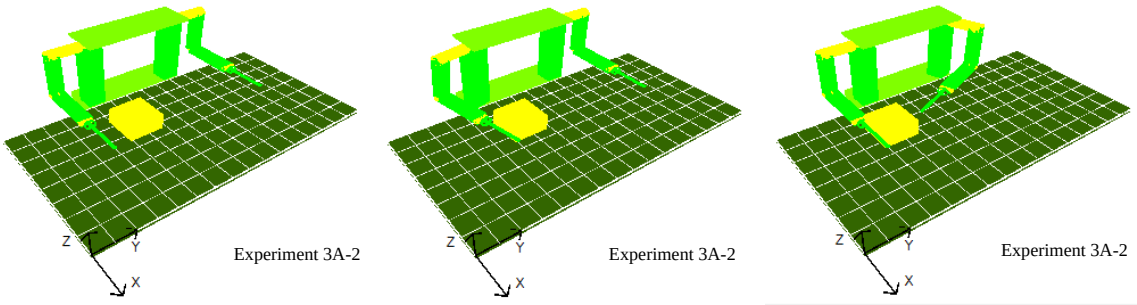


Figure 114 Simulation Results of Experiment 3A-2

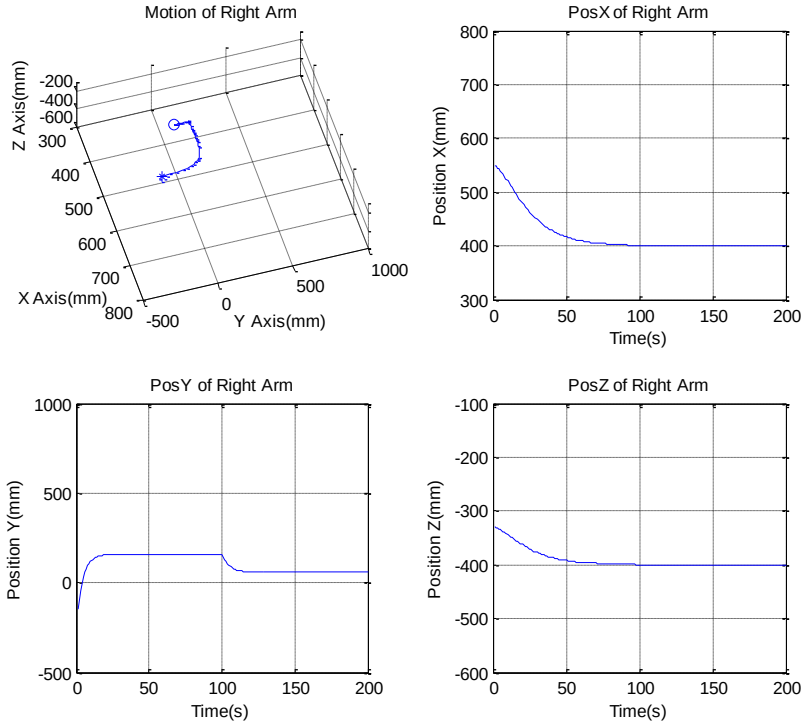


Figure 115 Generated Motion Trajectories of the Right Arm in Experiment 3A-2

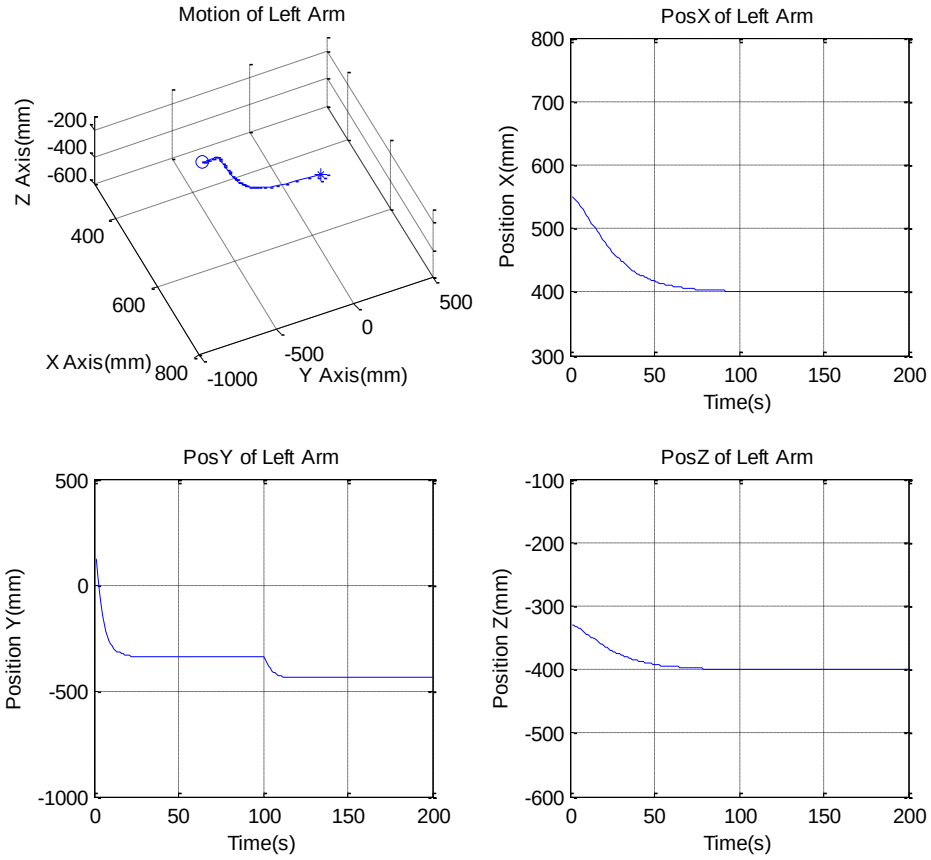


Figure 116 Generated Motion Trajectories of the Left Arm in Experiment 3A-2

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3A-3. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

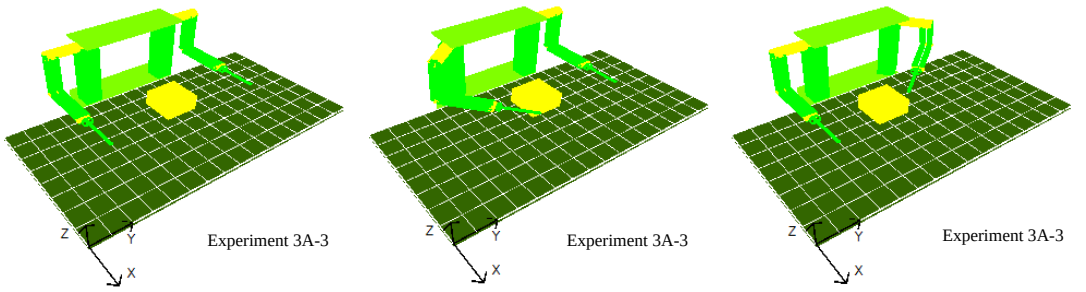


Figure 117 Simulation Results of Experiment 3A-3

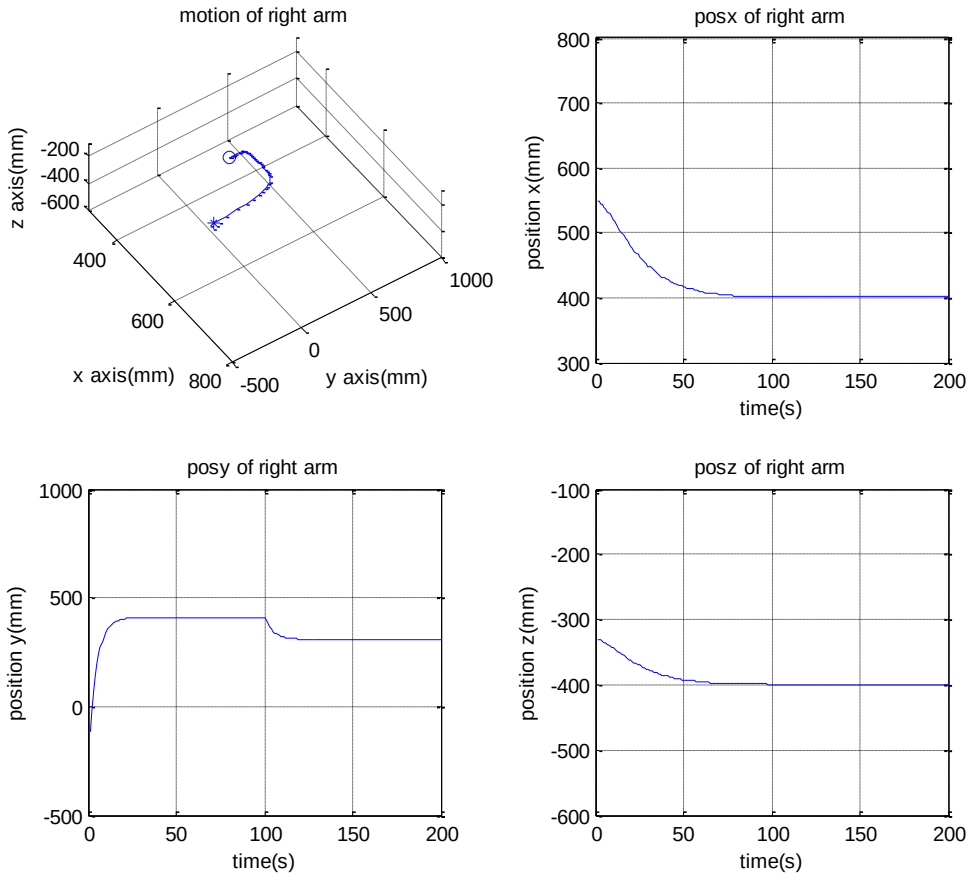


Figure 118 Generated Motion Trajectories of the Right Arm in Experiment 3A-3

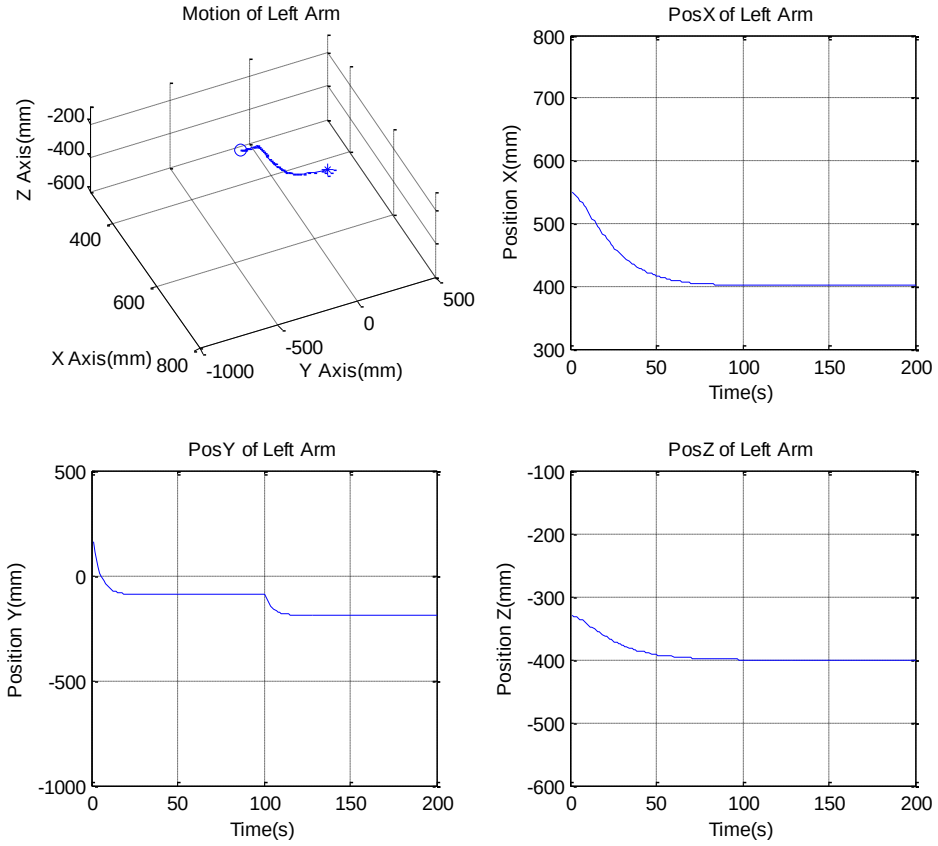


Figure 119 Generated Motion Trajectories of the Left Arm in Experiment 3A-3

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3A-4. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

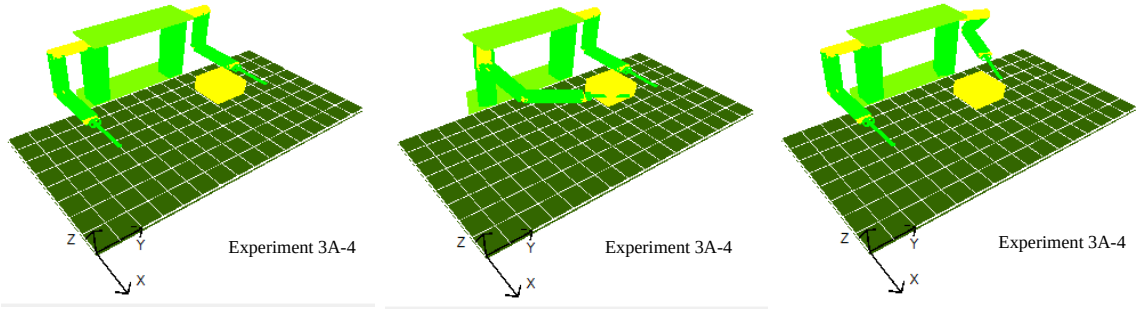


Figure 120 Simulation Results of Experiment 3A-4

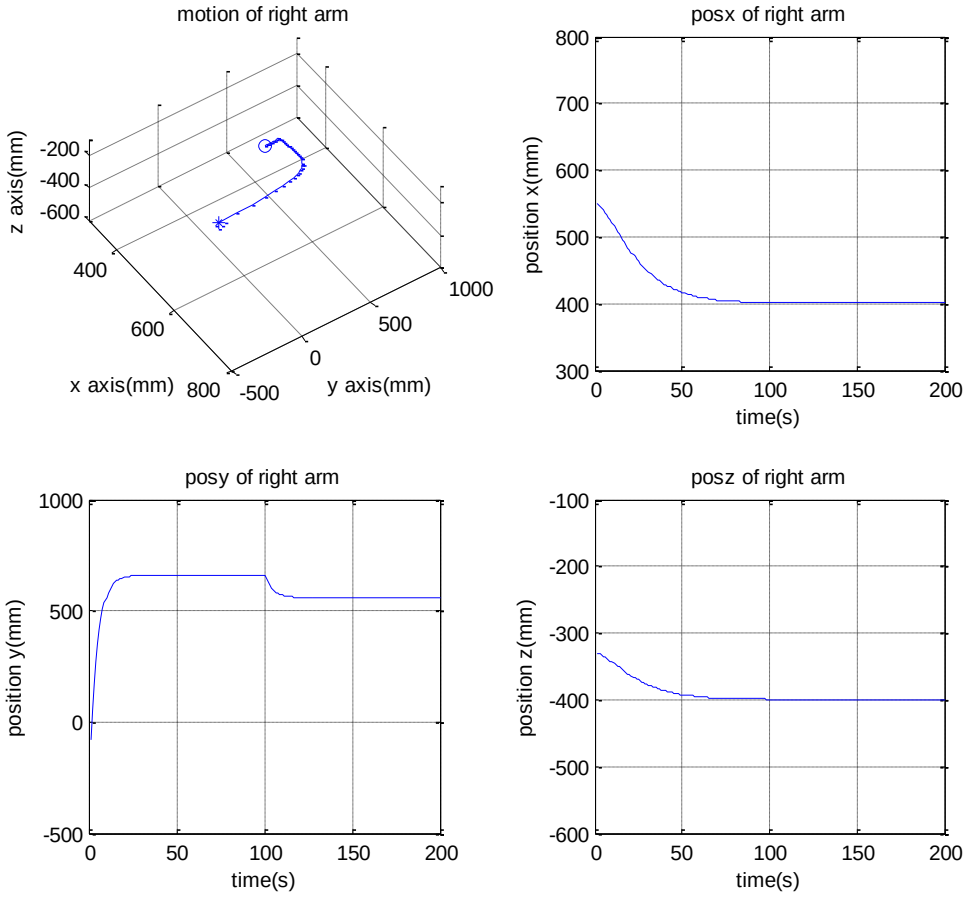


Figure 121 Generated Motion Trajectories of the Right Arm in Experiment 3A-4

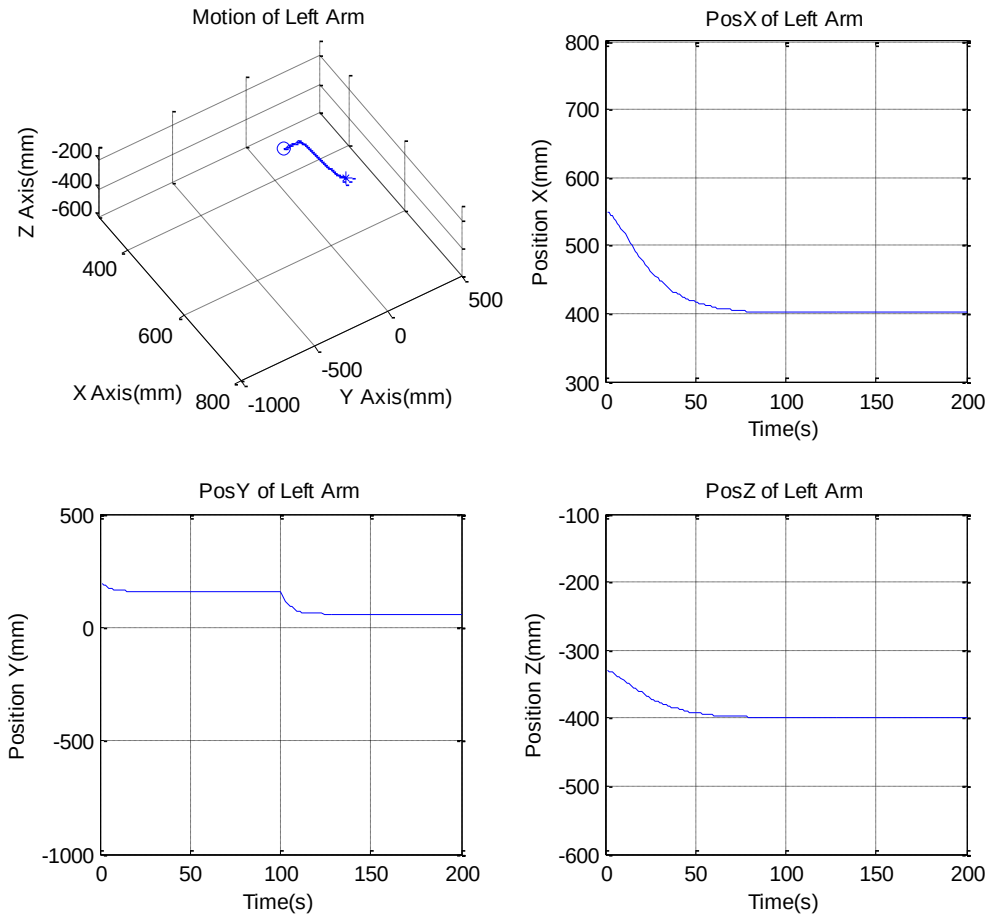


Figure 122 Generated Motion Trajectories of the Left Arm in Experiment 3A-4

ISAC finds that it cannot push the box to the right using either of the arms in Experiment 3A-5.

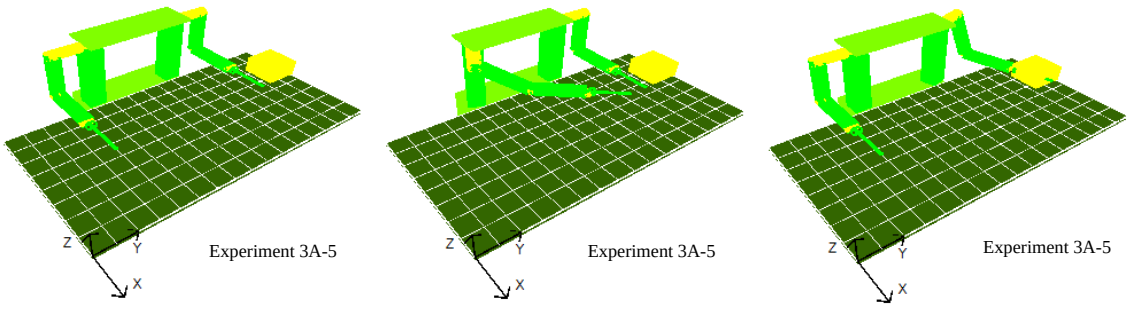


Figure 123 Simulation Results of Experiment 3A-5

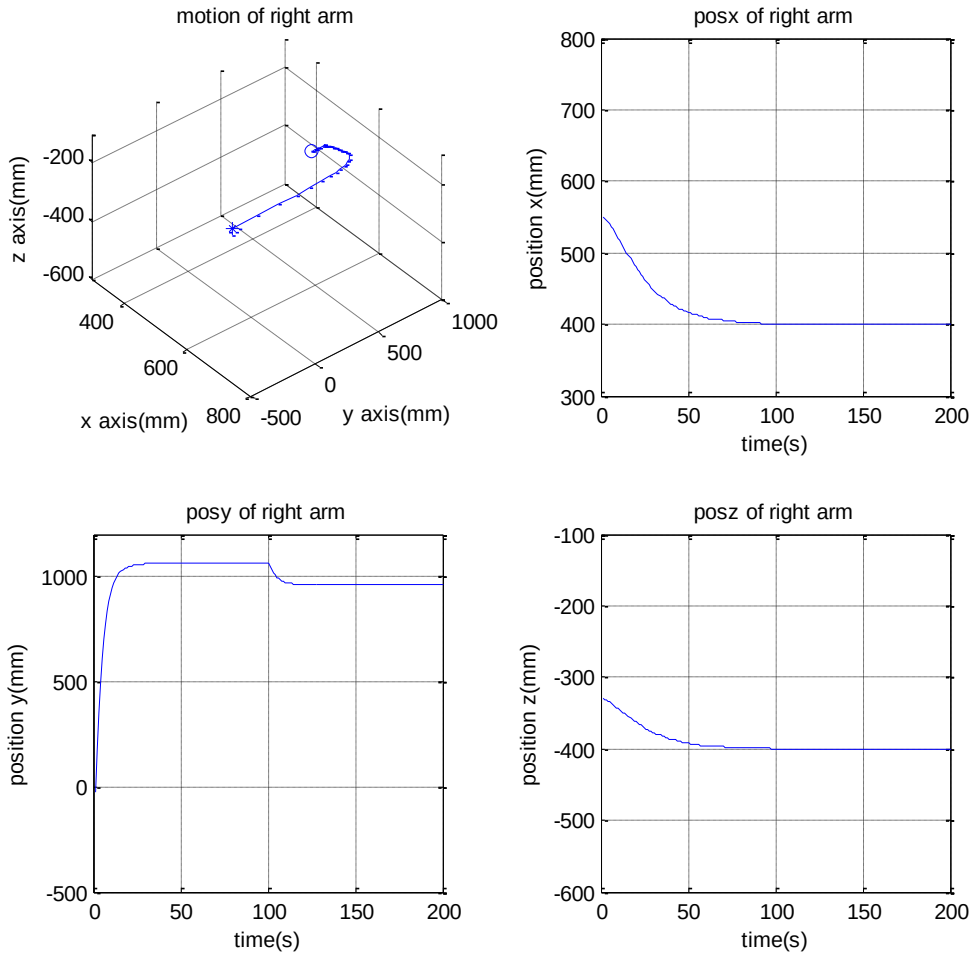


Figure 124 Generated Motion Trajectories of the Right Arm in Experiment 3A-5

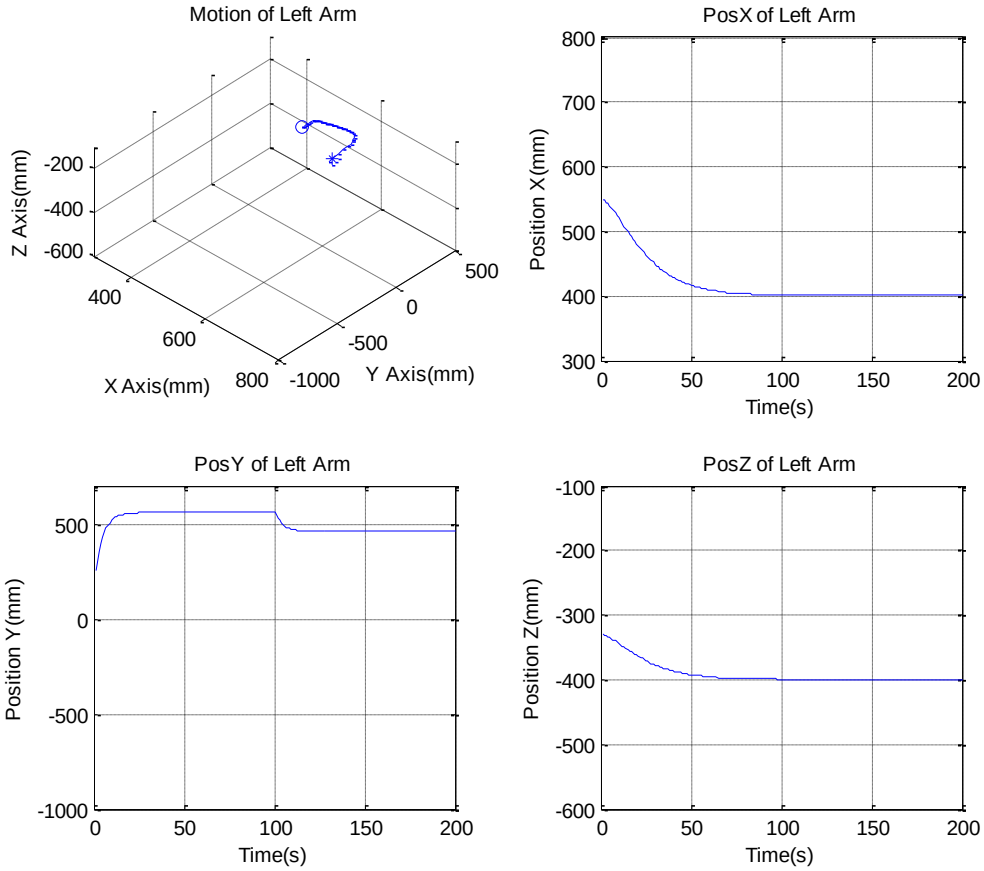


Figure 125 Generated Motion Trajectories of the Left Arm in Experiment 3A-5

ISAC pushes the box to the right using its right arm Experiment 3A-6.

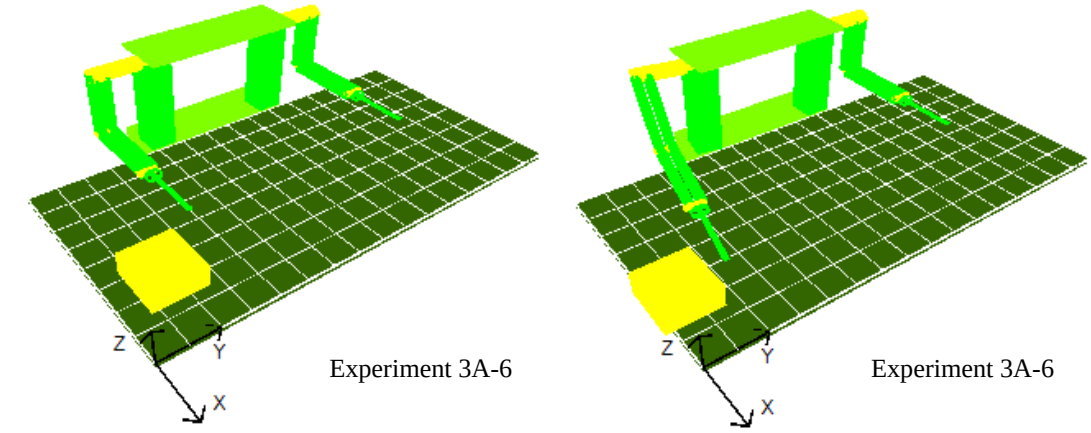


Figure 126 Simulation Results of Experiment 3A-6

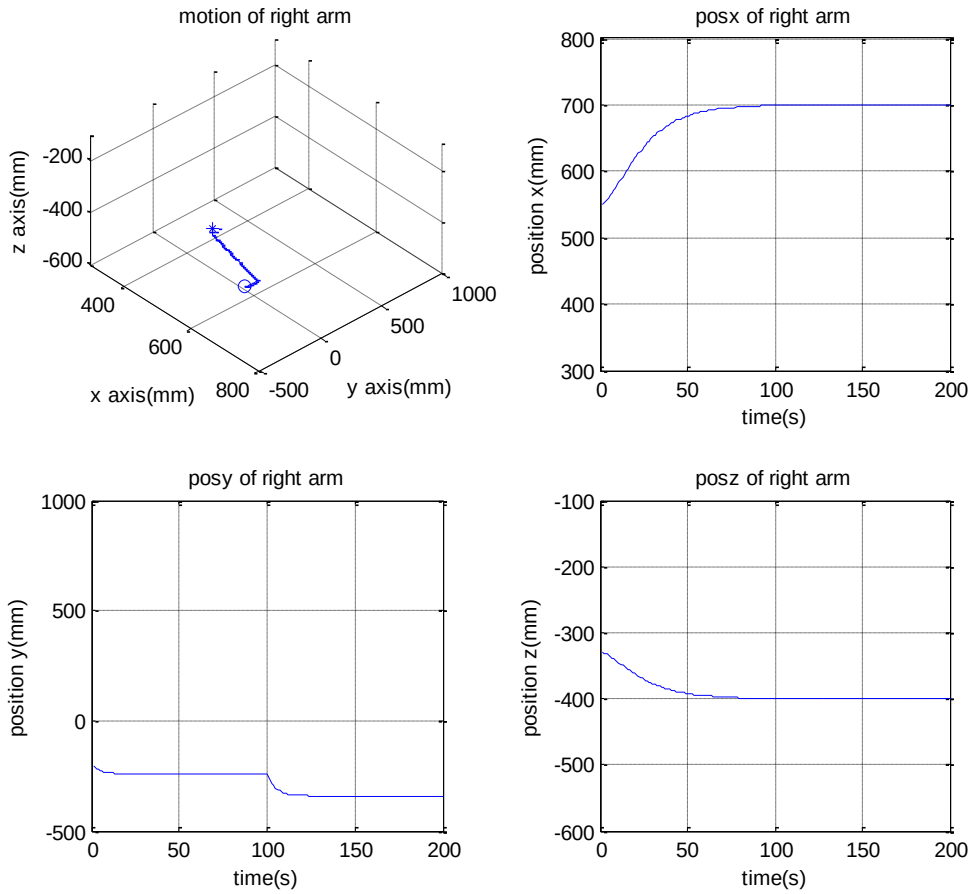


Figure 127 Generated Motion Trajectories of the Right Arm in Experiment 3A-6

ISAC pushes the box to the right using its right arm Experiment 3A-7.

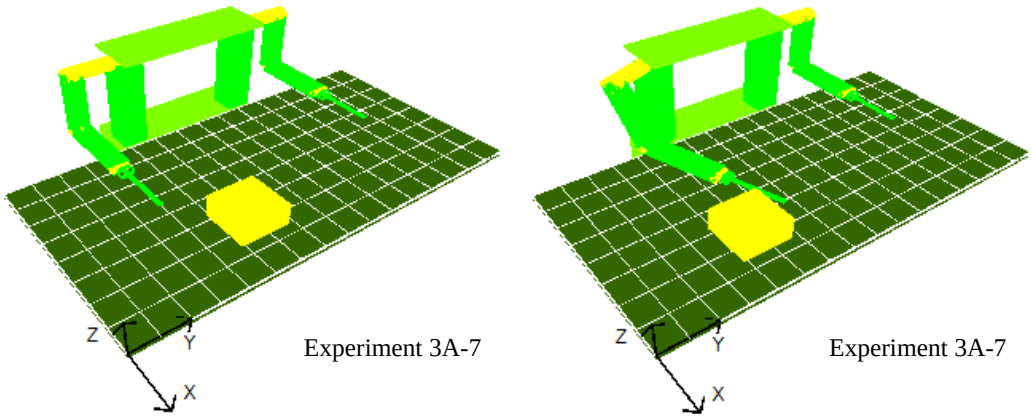


Figure 128 Simulation Results of Experiment 3A-7

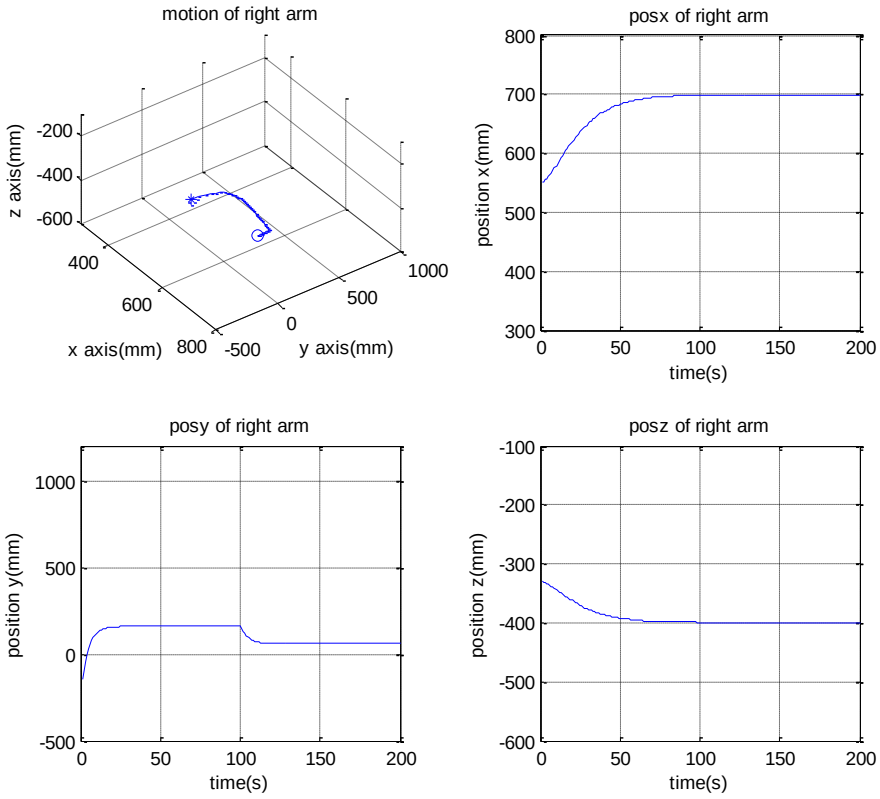


Figure 129 Generated Motion Trajectories of the Right Arm in Experiment 3A-7

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3A-8. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

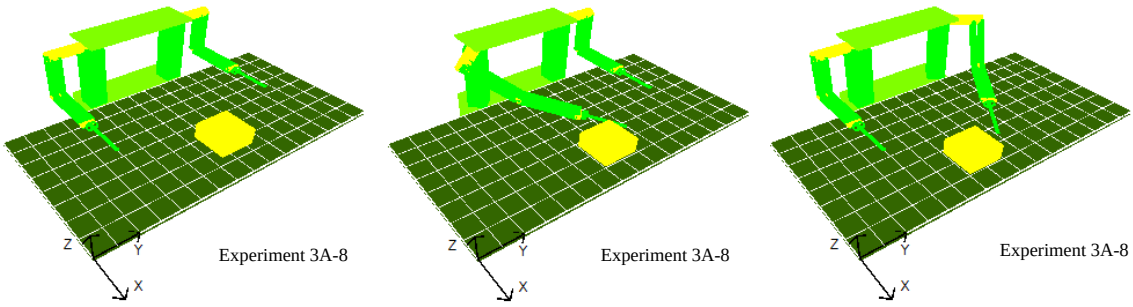


Figure 130 Simulation Results of Experiment 3A-8

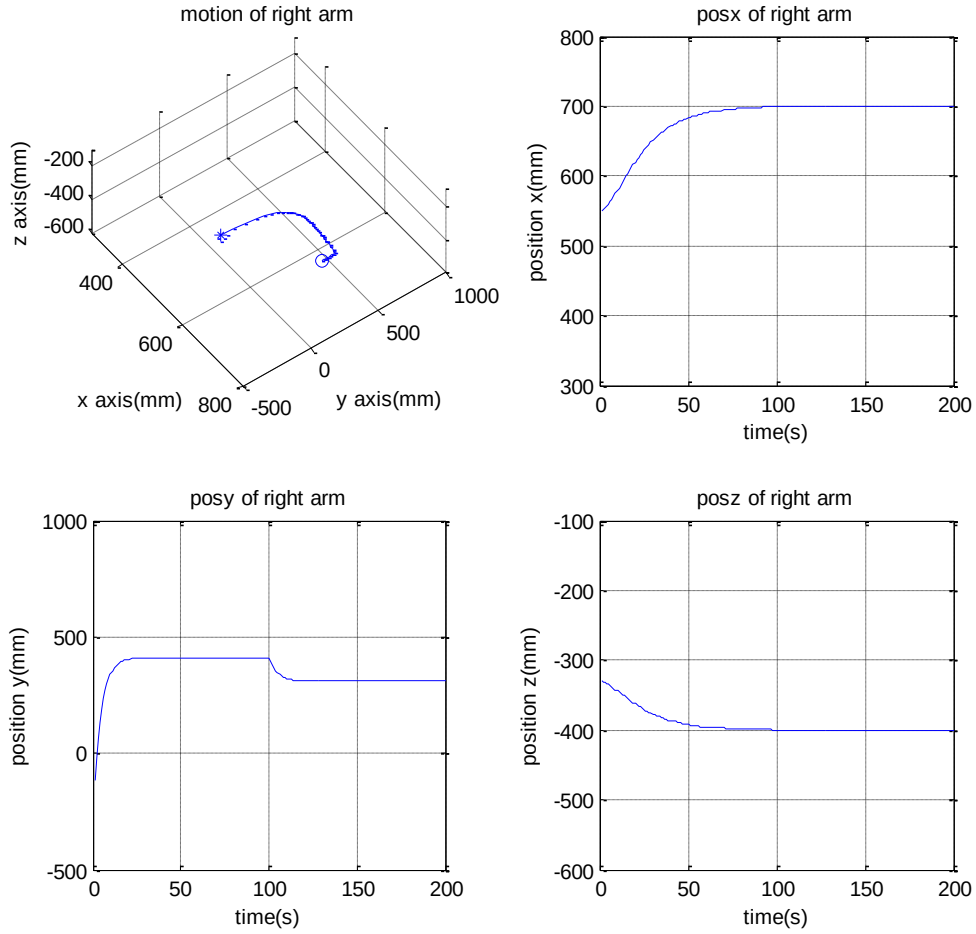


Figure 131 Generated Motion Trajectories of the Right Arm in Experiment 3A-8

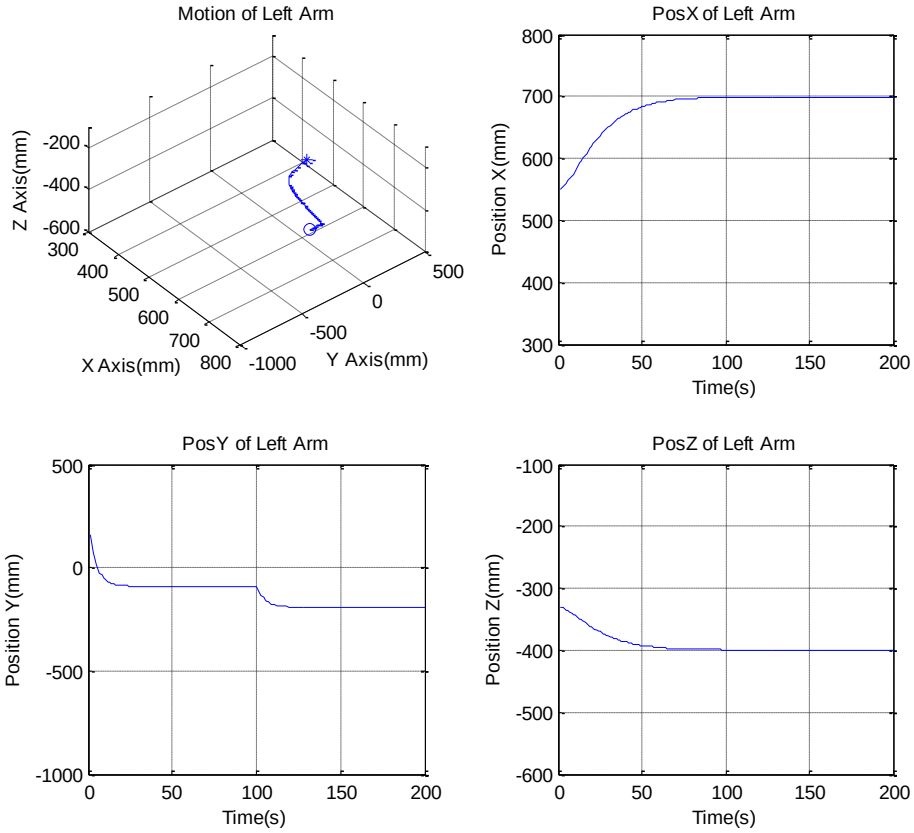


Figure 132 Generated Motion Trajectories of the Left Arm in Experiment 3A-8

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3A-9. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

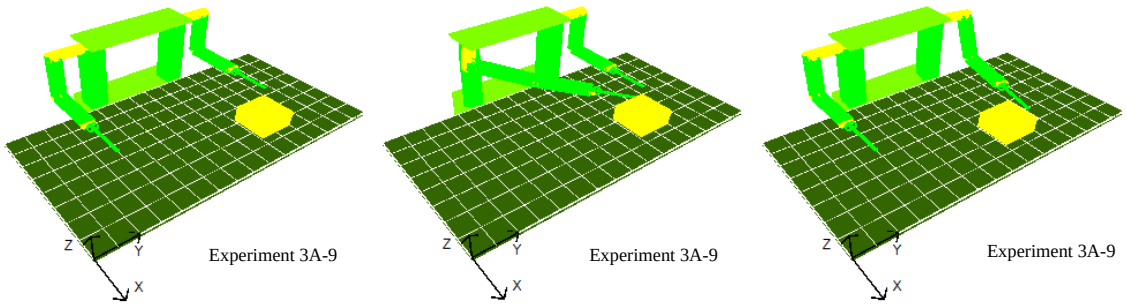


Figure 133 Simulation Results of Experiment 3A-9

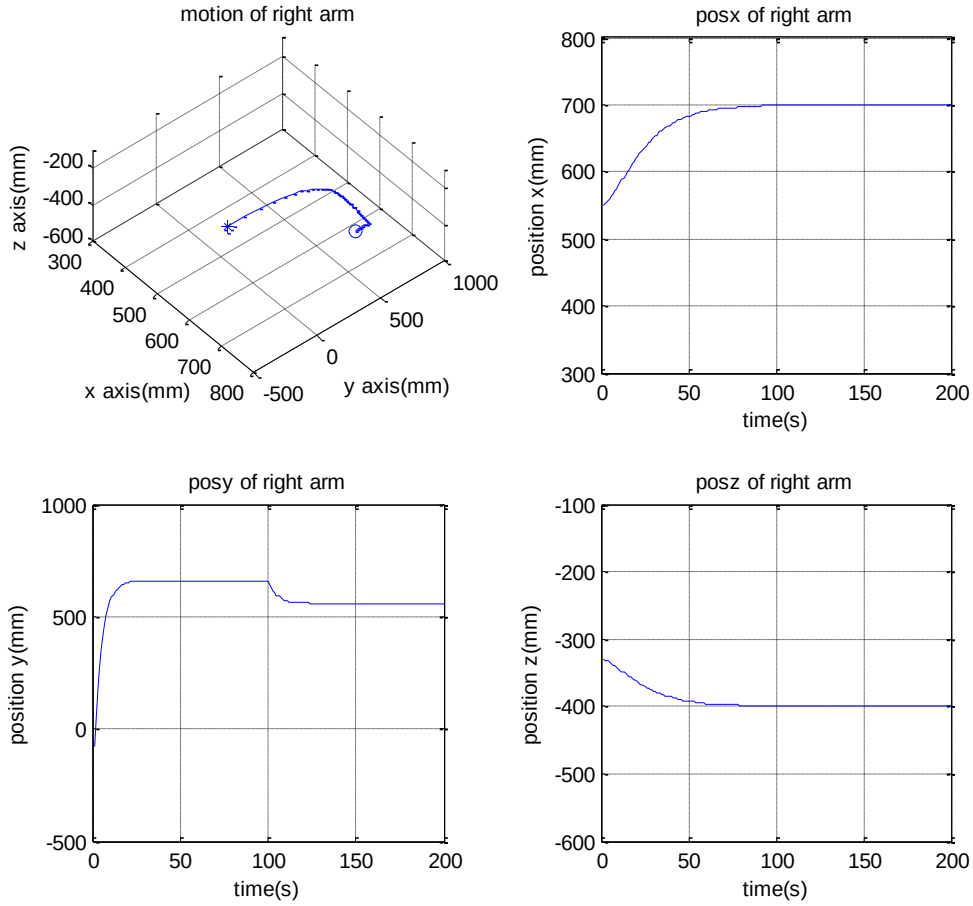


Figure 134 Generated Motion Trajectories of the Right Arm in Experiment 3A-9

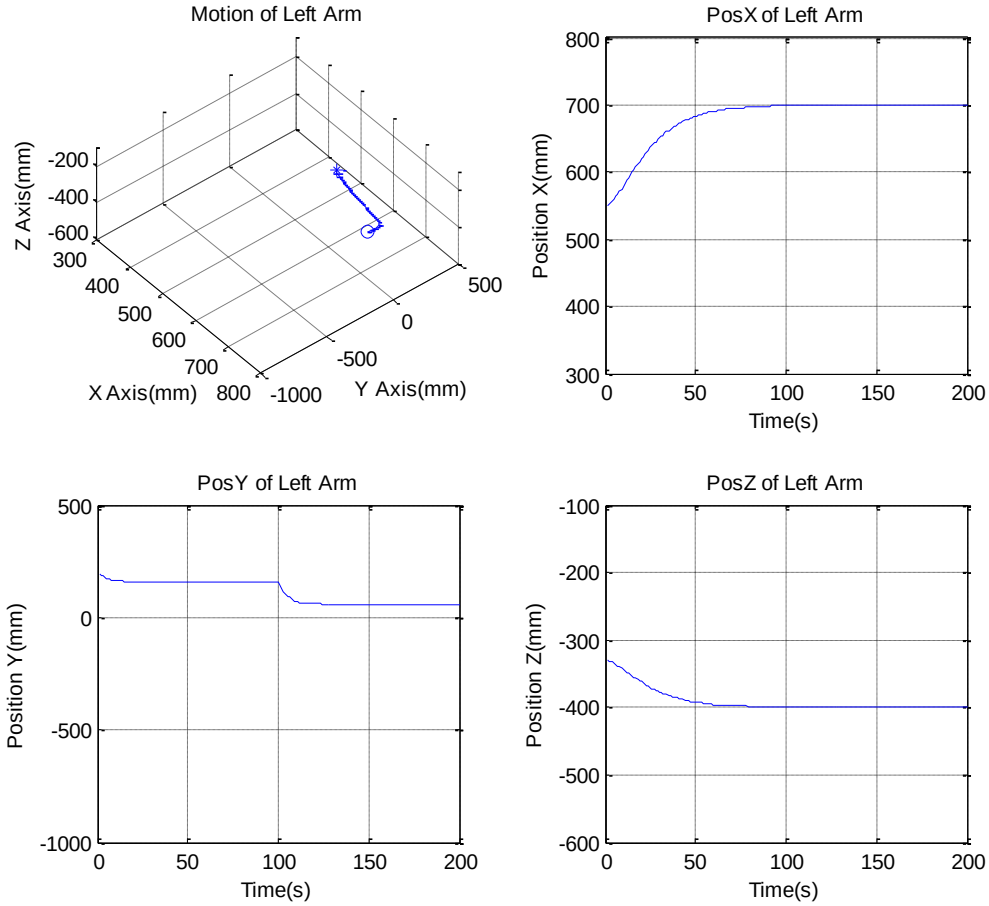


Figure 135 Generated Motion Trajectories the Left Arm in Experiment 3A-9

ISAC finds that it cannot push the box to the right using either of the arms in Experiment 3A-10.

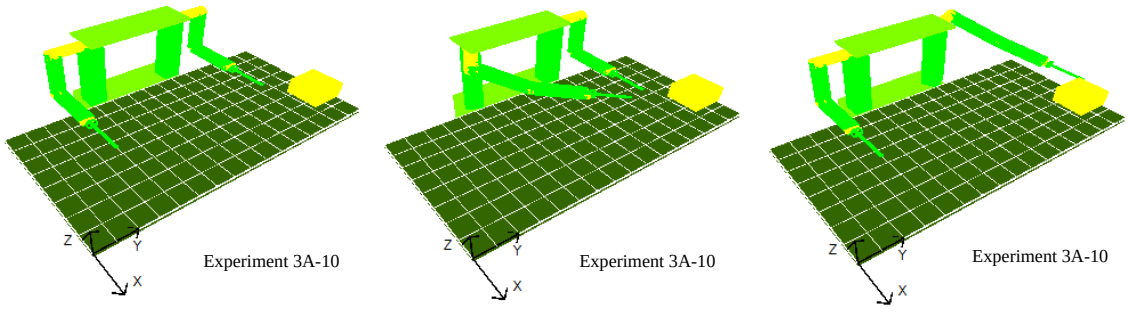


Figure 136 Simulation Results of Experiment 3A-10

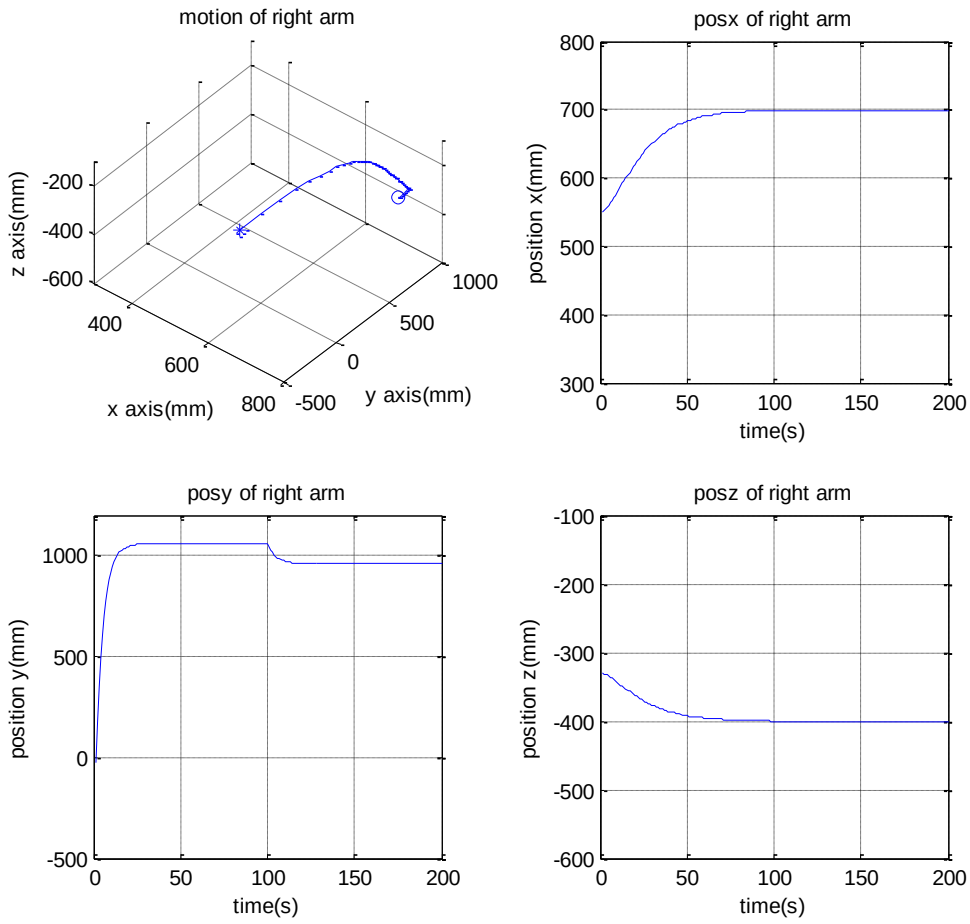


Figure 137 Generated Motion Trajectories of the Right Arm in Experiment 3A-10

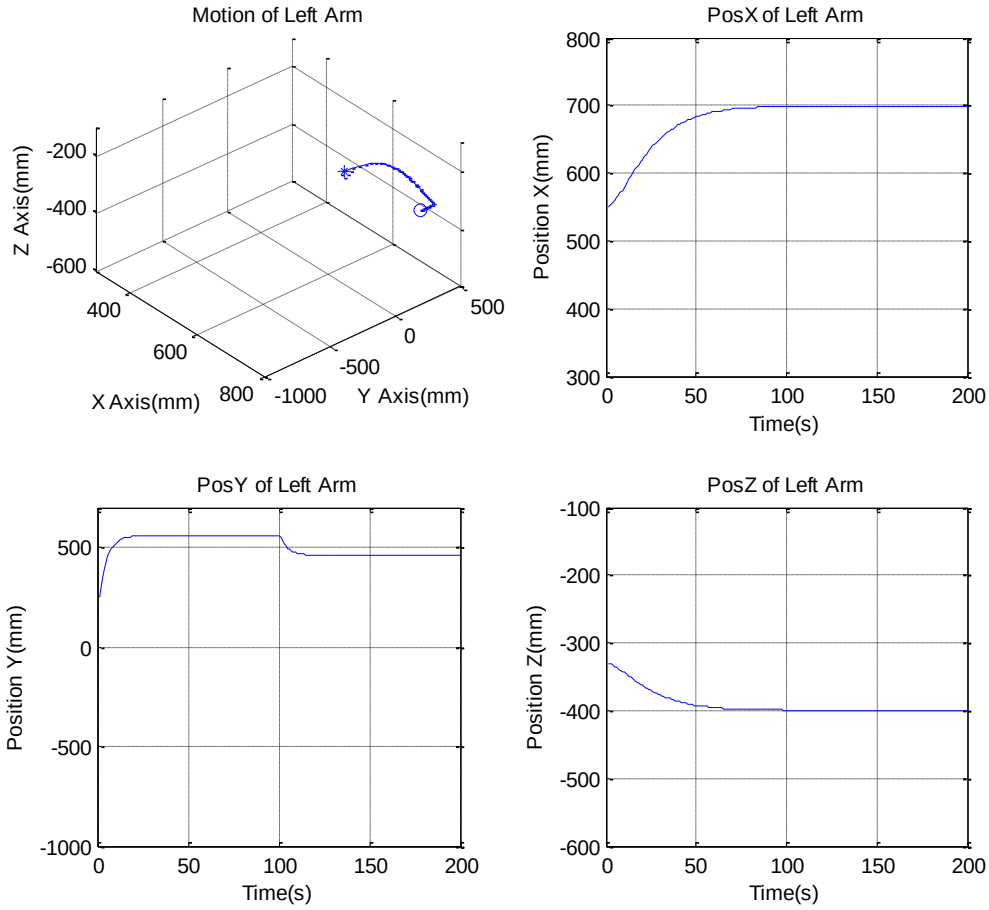


Figure 138 Generated Motion Trajectories of the Left Arm in Experiment 3A-10

Experiment 3B

ISAC pushes the box to the right using its right arm in Experiment 3B-1a-1.

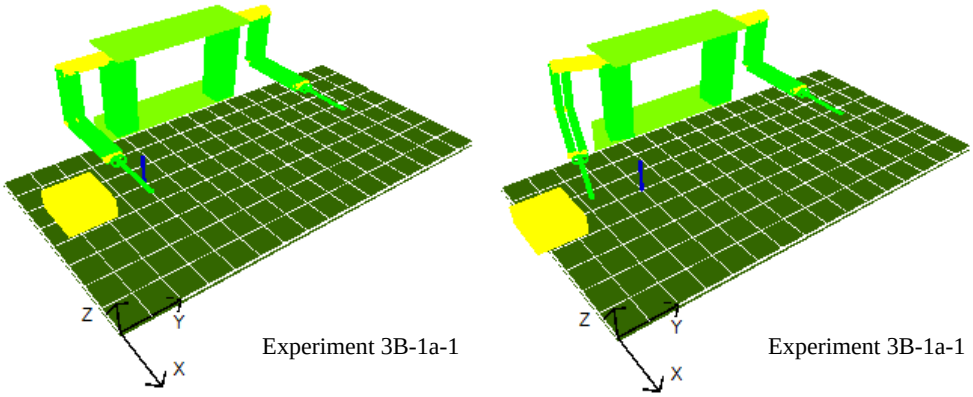


Figure 139 Simulation Results of Experiment 3B-1a-1

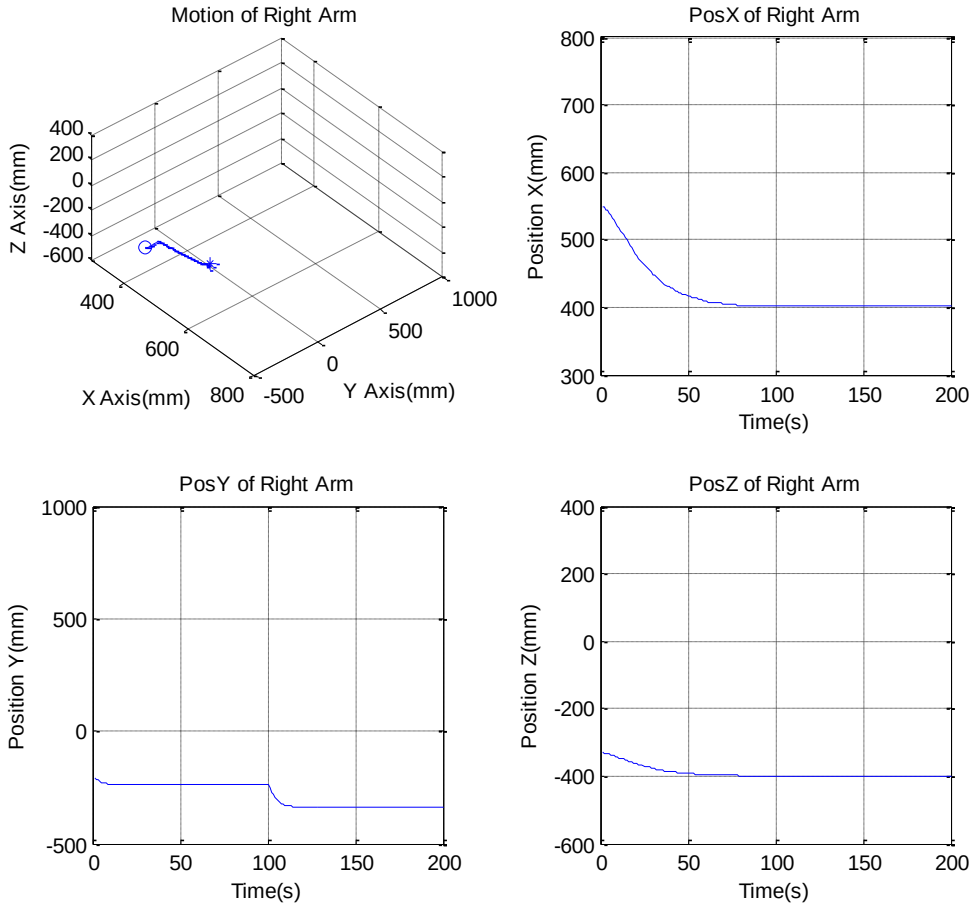


Figure 140 Generated Motion Trajectories of the Right Arm in Experiment 3B-1a-1

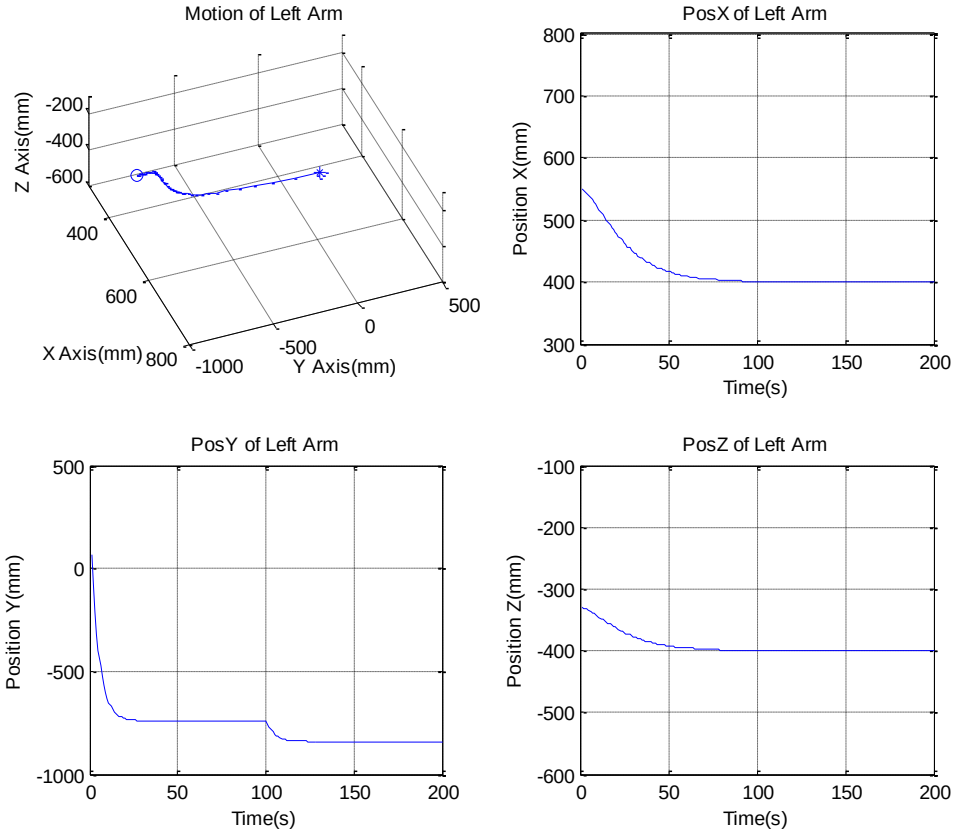


Figure 141 Generated Motion Trajectories of the Left Arm in Experiment 3B-1a-1

ISAC pushes the box to the right using its right arm in Experiment 3B-1a-2.

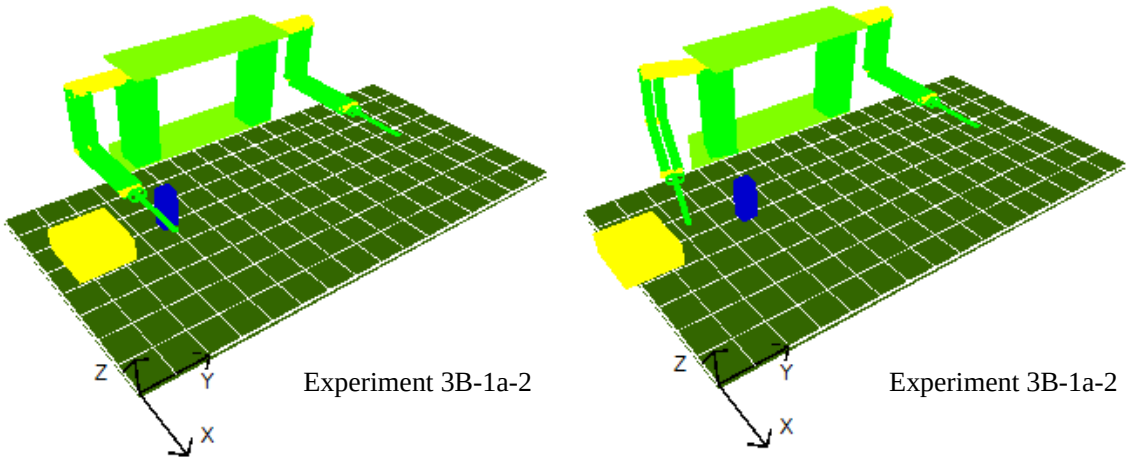


Figure 142 Simulation Results of Experiment 3B-1a-2

ISAC finds that it cannot push the box to the right using either of the arms in Experiment 3B-1a-3.

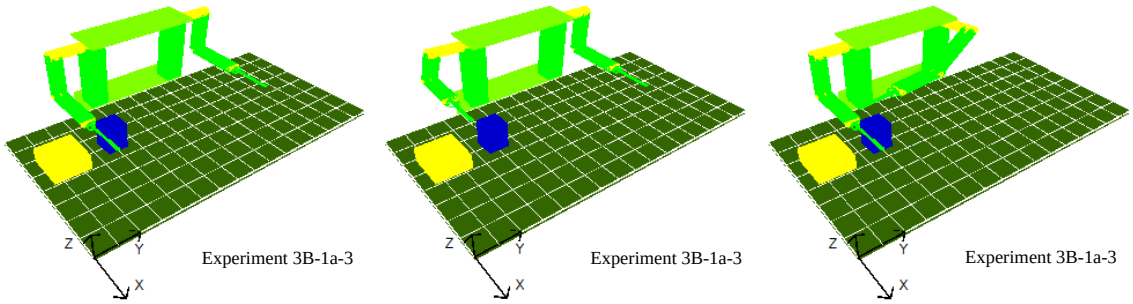


Figure 143 Simulation Results of Experiment 3B-1a-3

ISAC finds that it cannot push the box to the right using either of the arms in Experiment 3B-1a-4.

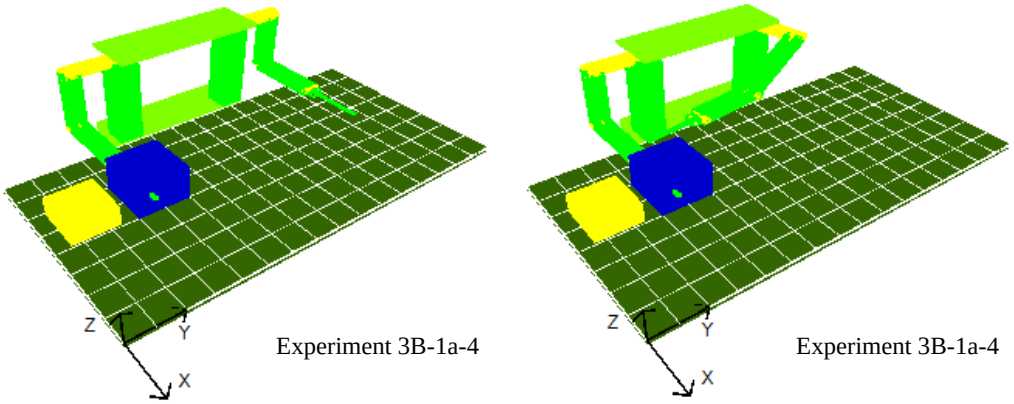


Figure 144 Simulation Results of Experiment 3B-1a-3

ISAC pushes the box to the right using its right arm in Experiment 3B-1b-1.

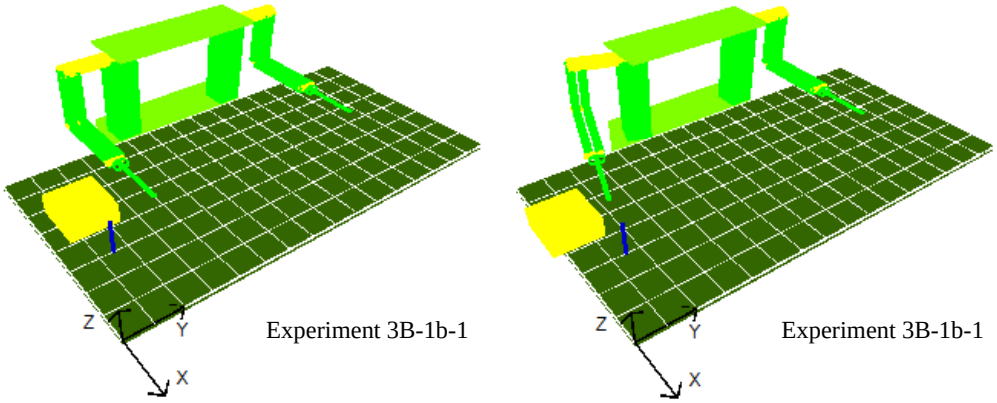


Figure 145 Simulation Results of Experiment 3B-1b-1

ISAC pushes the box to the right using its right arm in Experiment 3B-1b-2.

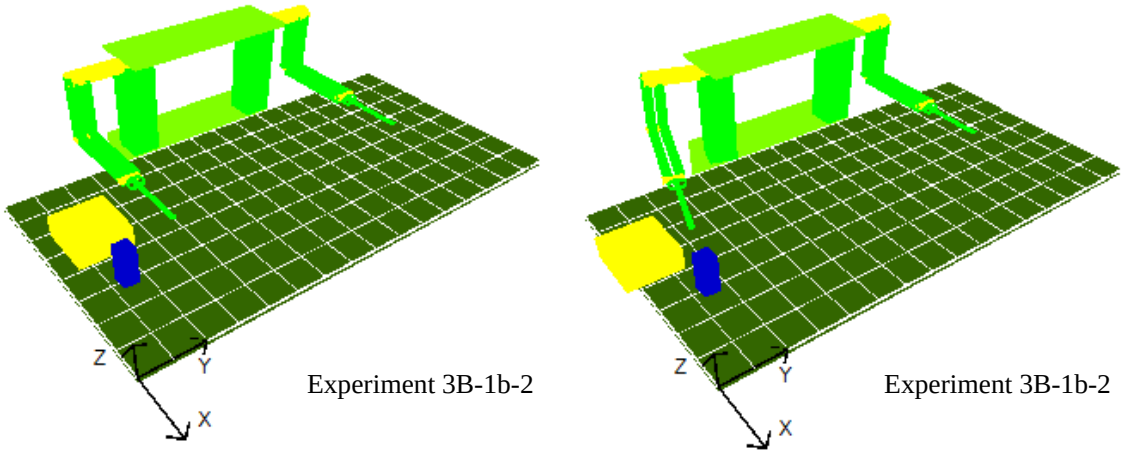


Figure 146 Simulation Results of Experiment 3B-1b-2

ISAC pushes the box to the right using its right arm in Experiment 3B-1b-3.

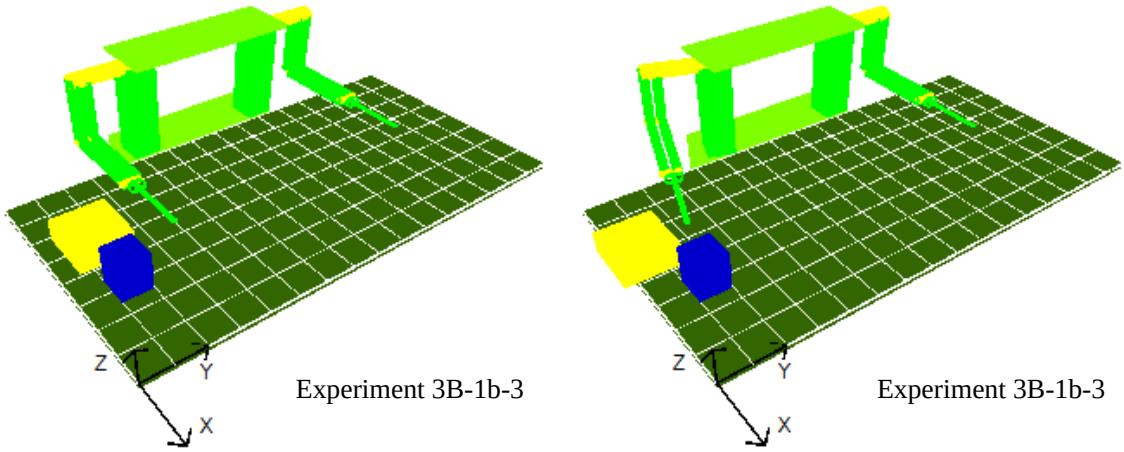


Figure 147 Simulation Results of Experiment 3B-1b-3

ISAC finds that it cannot push the box to the right using its either of the arms in Experiment 3B-1b-4.

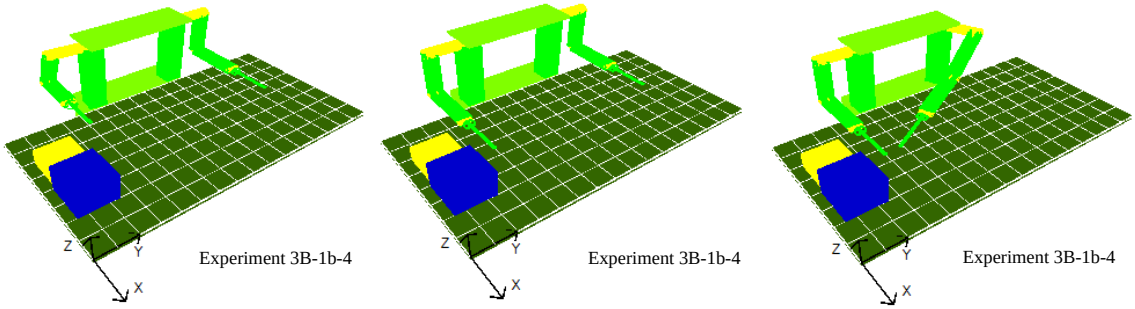


Figure 148 Simulation Results of Experiment 3B-1b-4

ISAC pushes the box to the right using its right arm in Experiment 3B-1c-1.

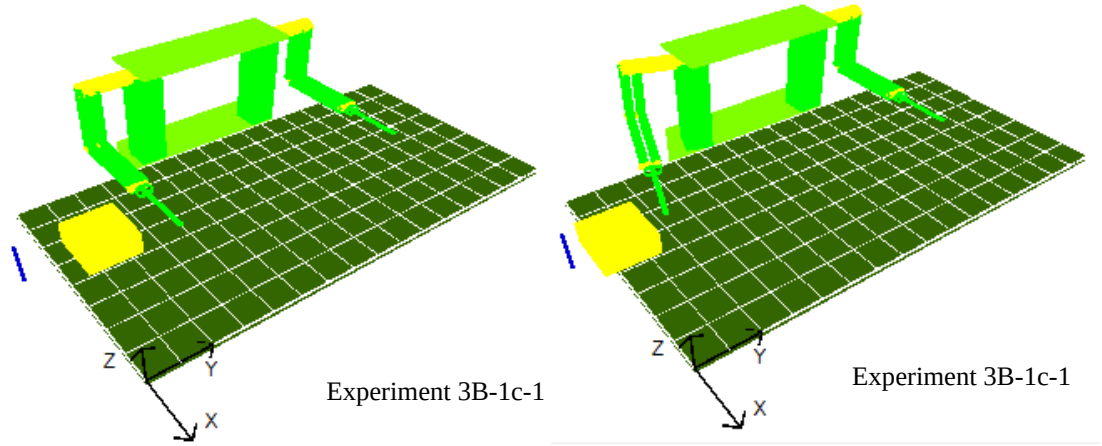


Figure 149 Simulation Results of Experiment 3B-1c-1

ISAC pushes the box to the right using its right arm in Experiment 3B-1c-2.

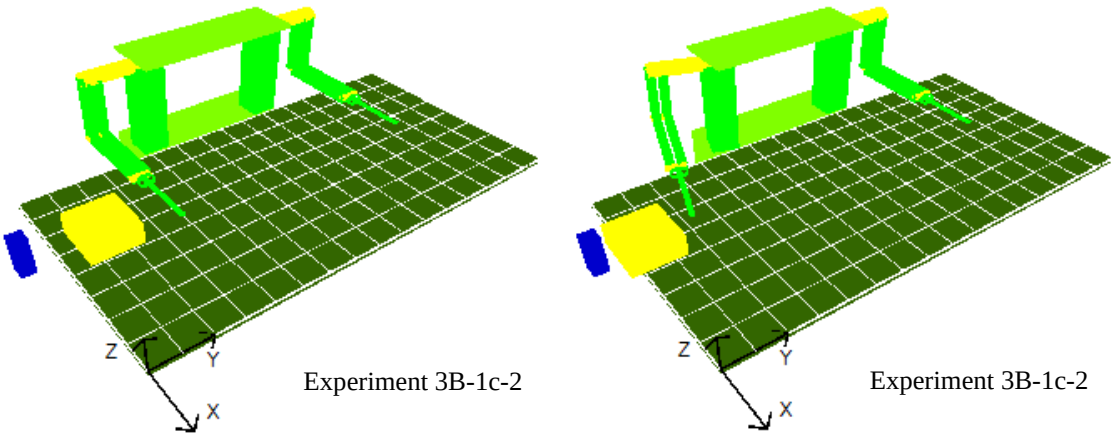


Figure 150 Simulation Results of Experiment 3B-1c-2

ISAC pushes the box to the right using its right arm in Experiment 3B-1c-3.

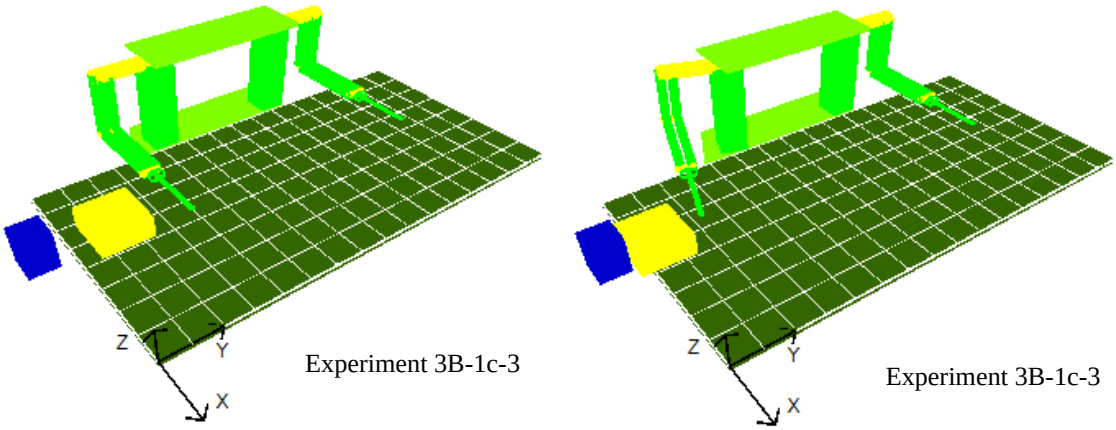


Figure 151 Simulation Results of Experiment 3B-1c-3

ISAC pushes the box to the right using its right arm in Experiment 3B-1c-4.

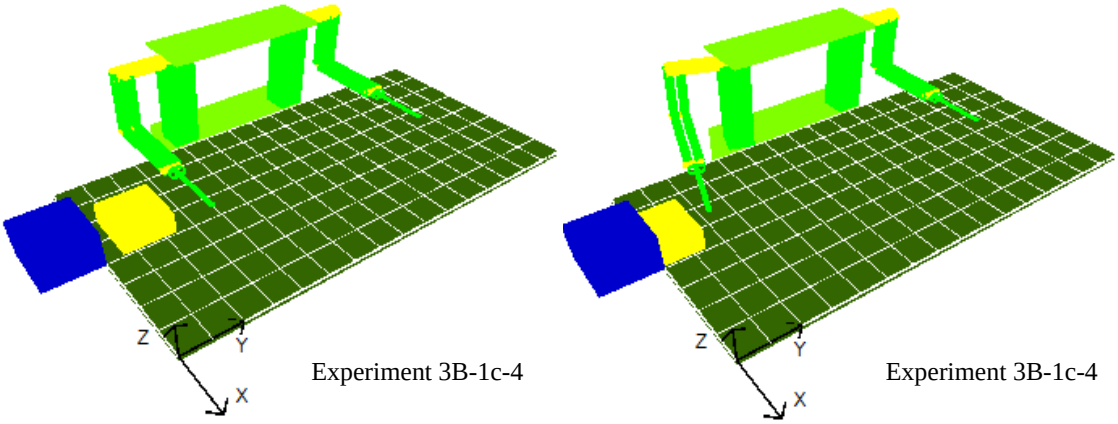


Figure 152 Simulation Results of Experiment 3B-1c-4

ISAC pushes the box to the right using its right arm in Experiment 3B-1d-1.

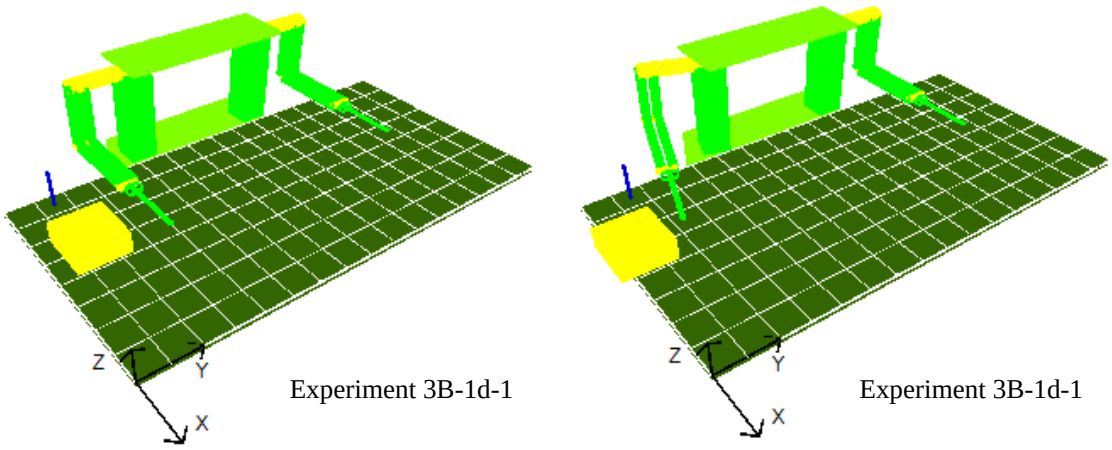


Figure 153 Simulation Results of Experiment 3B-1d-1

ISAC pushes the box to the right using its right arm in Experiment 3B-1d-2.

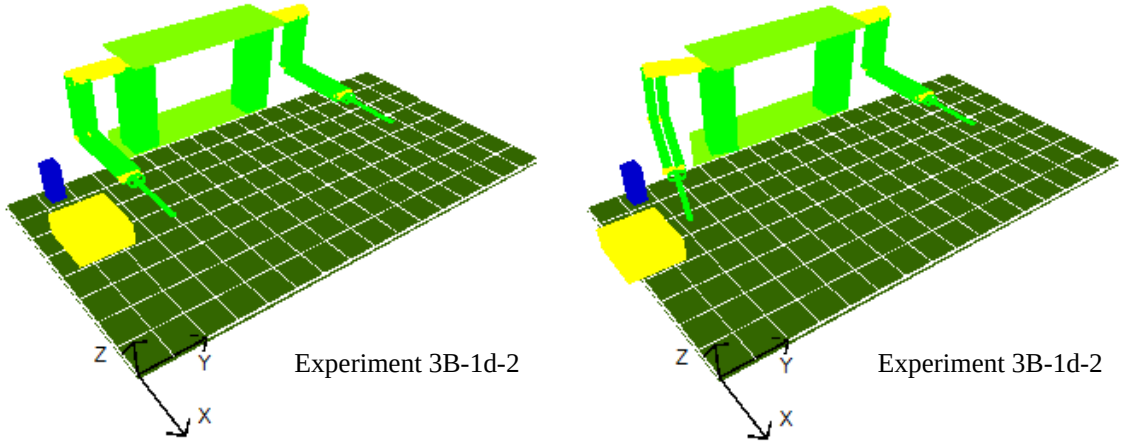


Figure 154 Simulation Results of Experiment 3B-1d-2

ISAC pushes the box to the right using its right arm in Experiment 3B-1d-3.

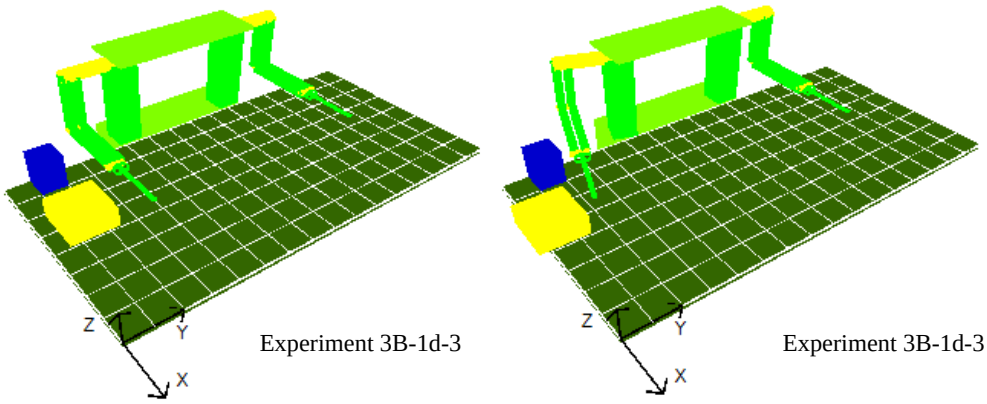


Figure 155 Simulation Results of Experiment 3B-1d-3

ISAC finds that it cannot push the box to the right using both arms in Experiment 3B-1d-4.

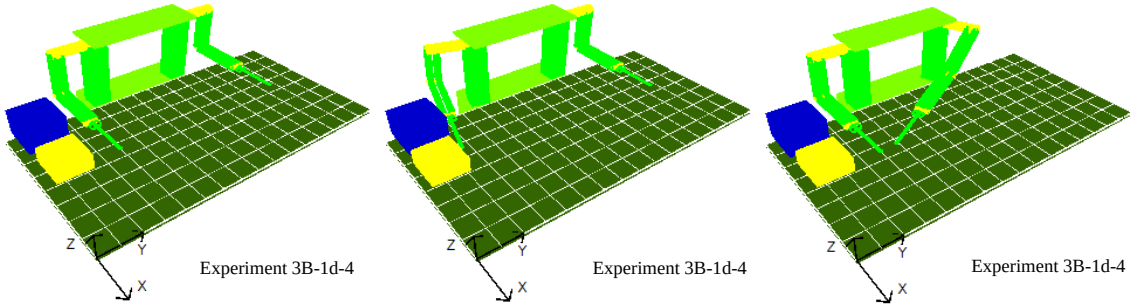


Figure 156 Simulation Results of Experiment 3B-1d-4

ISAC finds that it cannot push the box to the right using either of arms in Experiment 3B-2a-1.

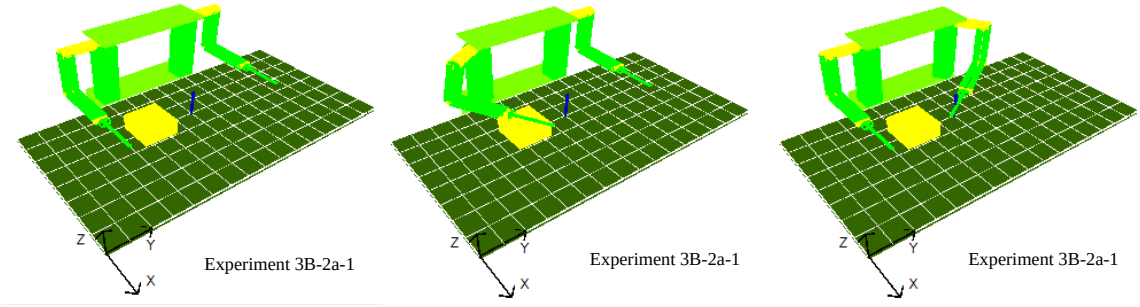


Figure 157 Simulation Results of Experiment 3B-2a-1

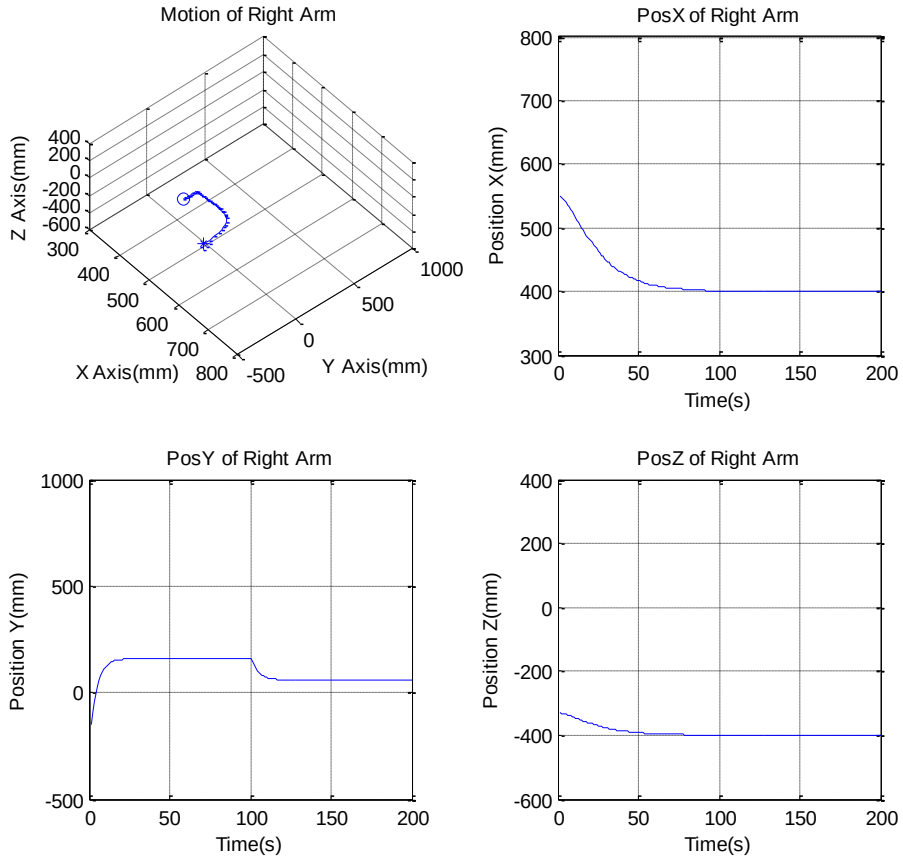


Figure 158 Generated Motion Trajectories of the Right Arm in Experiment 3B-2a-1

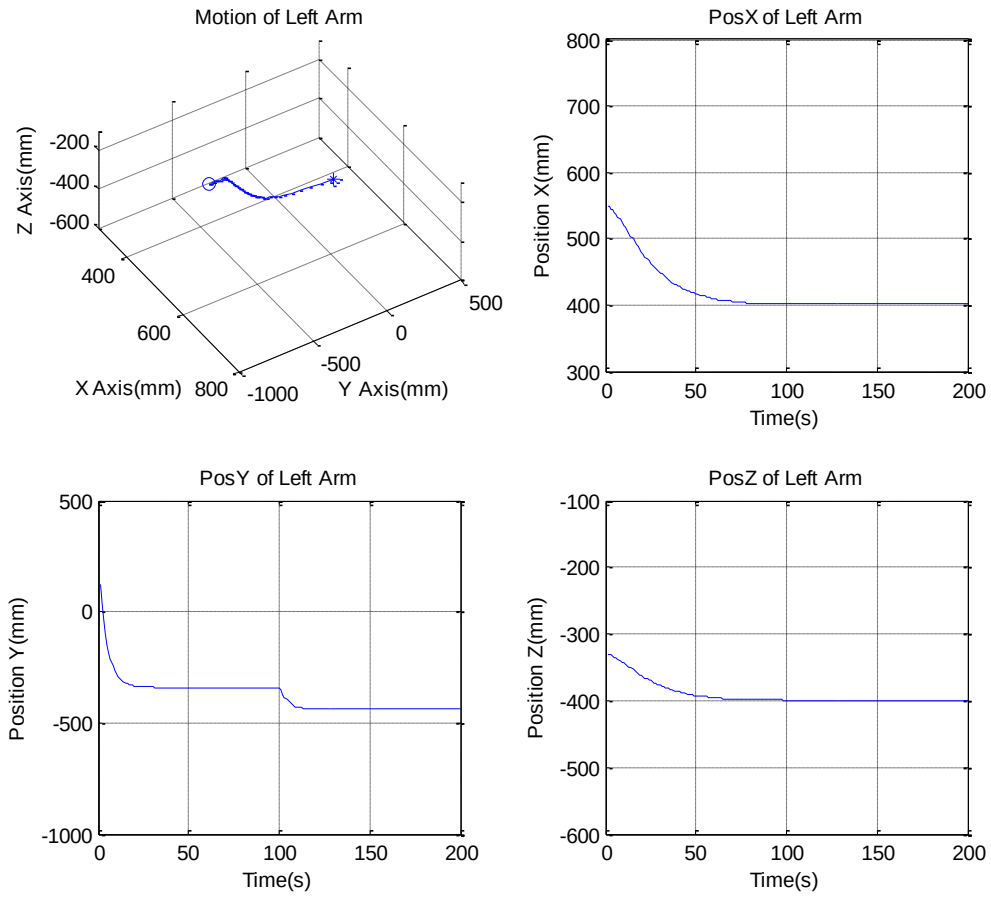


Figure 159 Generated Motion Trajectories of the Left Arm in Experiment 3B-2a-1

ISAC finds that it cannot push the box to the right using either of the arms in Experiment 3B-2a-2.

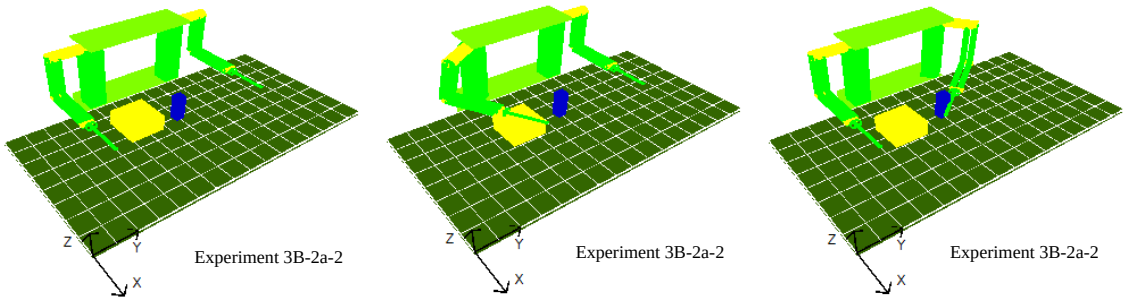


Figure 160 Simulation Results of Experiment 3B-2a-2

ISAC finds that it cannot push the box to the right using either of the arms in Experiment 3B-2a-3.

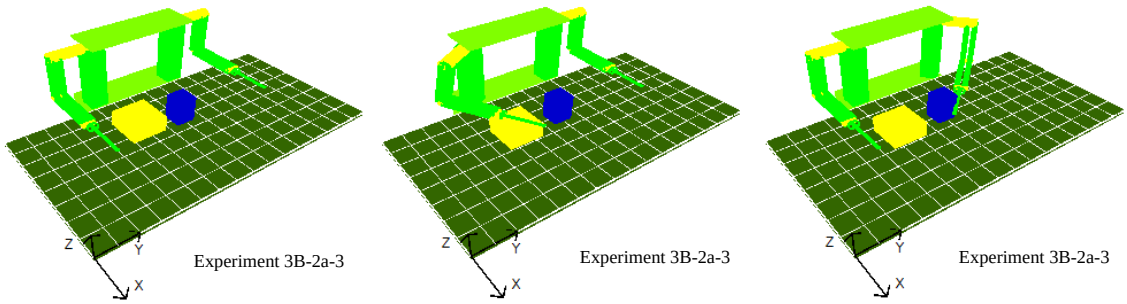


Figure 161 Simulation Results of Experiment 3B-2a-3

ISAC finds that it cannot push the box to the right using either of the arms in Experiment 3B-2a-4.

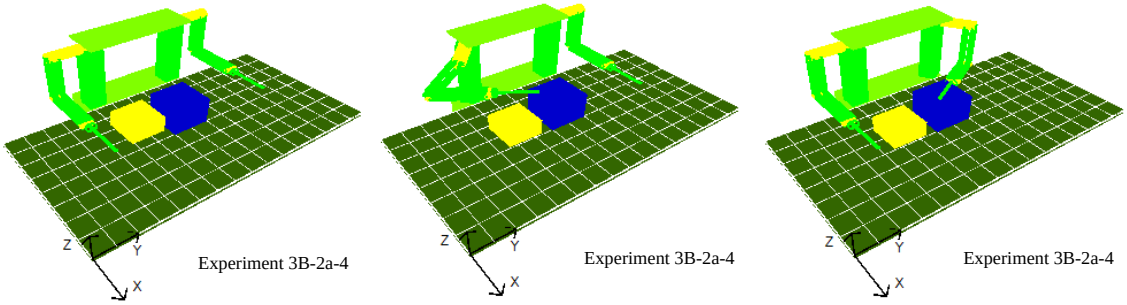


Figure 162 Simulation Results of Experiment 3B-2a-4

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-2b-1. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

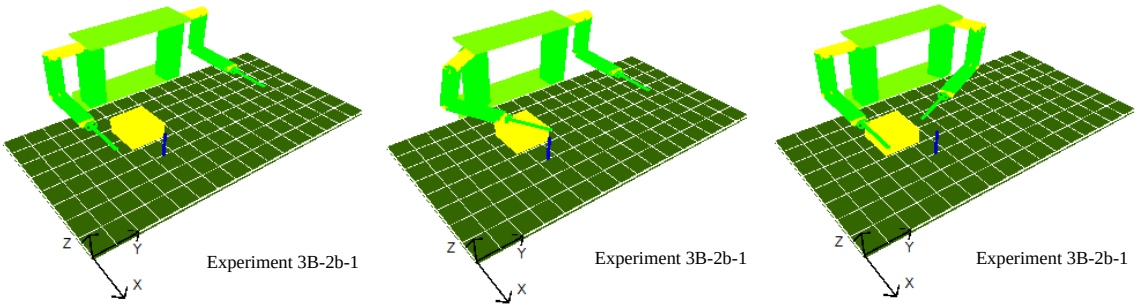


Figure 163 Simulation Results of Experiment 3B-2b-1

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-2b-2. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

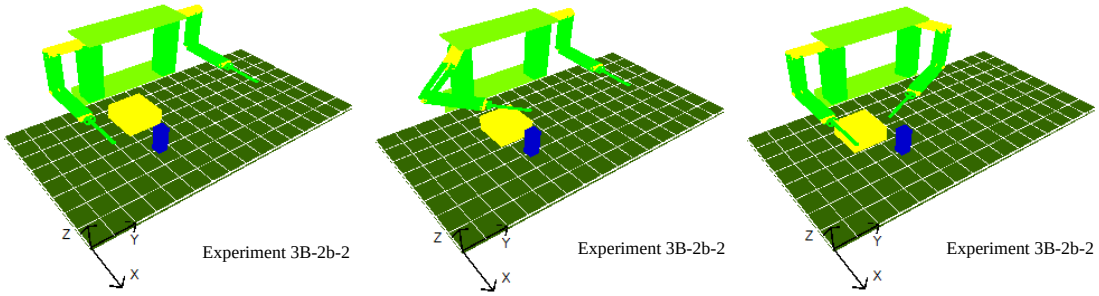


Figure 164 Simulation Results of Experiment 3B-2b-2

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-2b-3. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

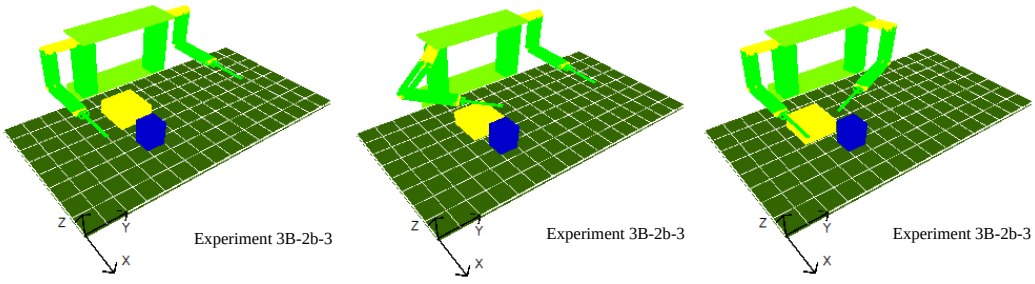


Figure 165 Simulation Results of Experiment 3B-2b-3

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-2b-4. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

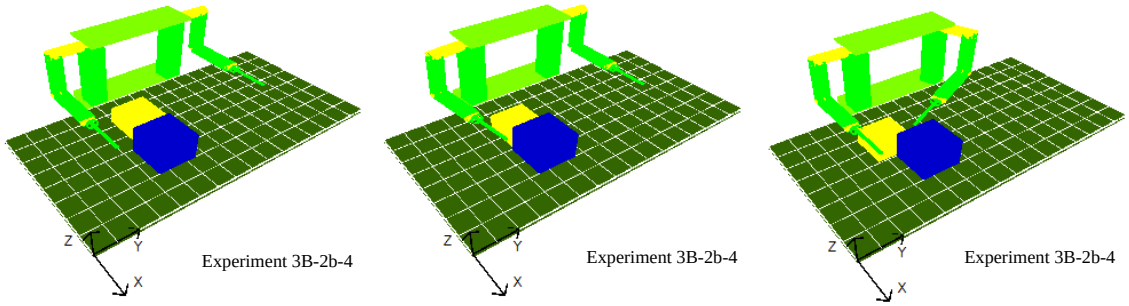


Figure 166 Simulation Results of Experiment 3B-2b-4

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-2c-1. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

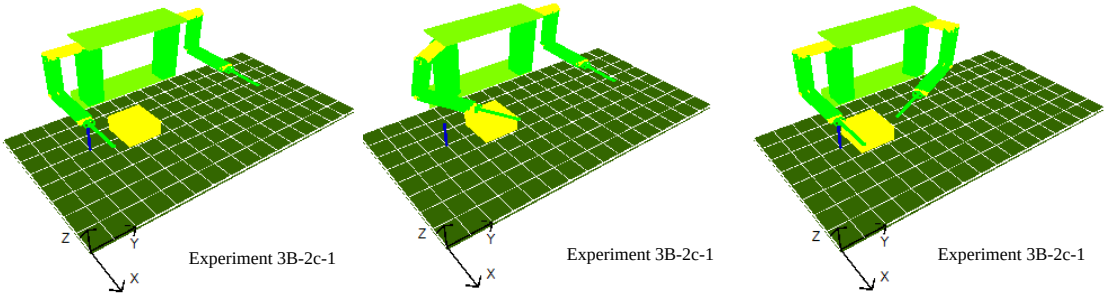


Figure 167 Simulation Results of Experiment 3B-2c-1

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-2c-2. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

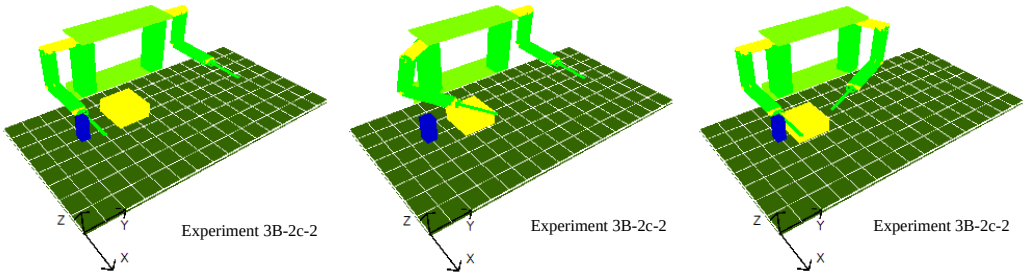


Figure 168 Simulation Results of Experiment 3B-2c-2

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-2c-3. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

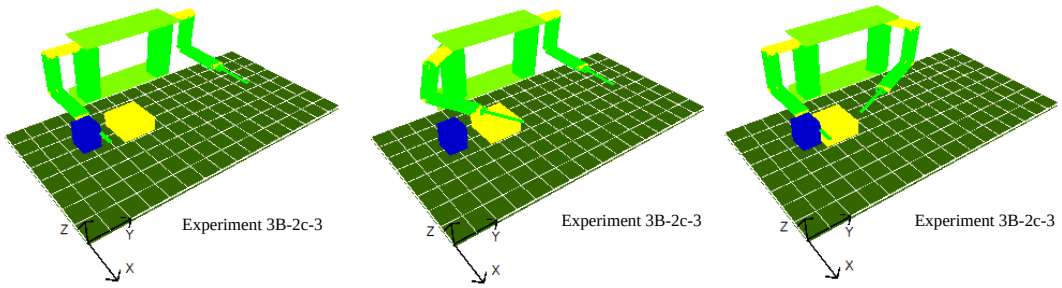


Figure 169 Simulation Results of Experiment 3B-2c-3

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-2c-4. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

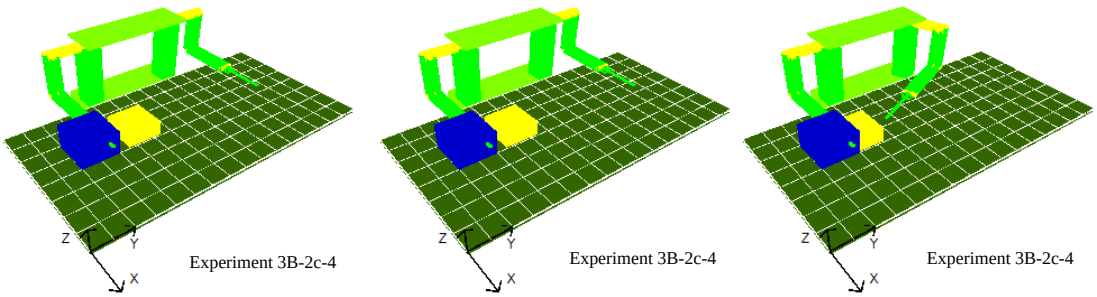


Figure 170 Simulation Results of Experiment 3B-2c-4

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-2d-1. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

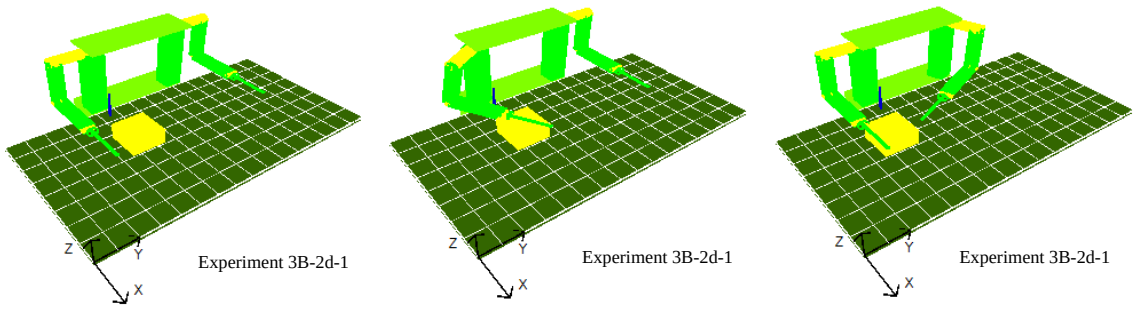


Figure 171 Simulation Results of Experiment 3B-2d-1

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-2d-2. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

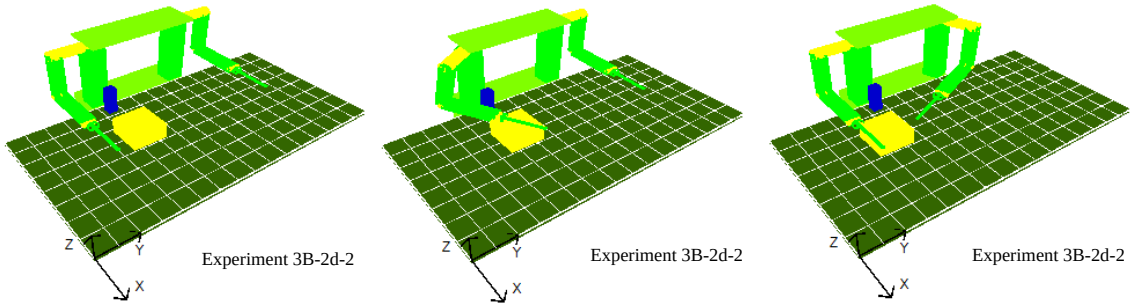


Figure 172 Simulation Results of Experiment 3B-2d-2

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-2d-3. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

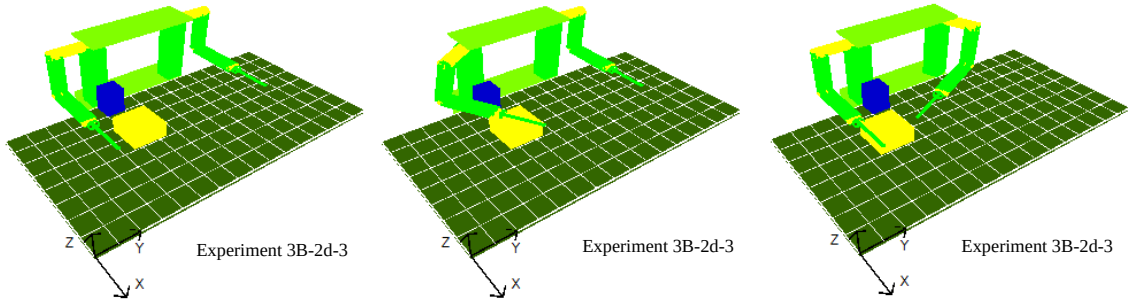


Figure 173 Simulation Results of Experiment 3B-2d-3

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-2d-4. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

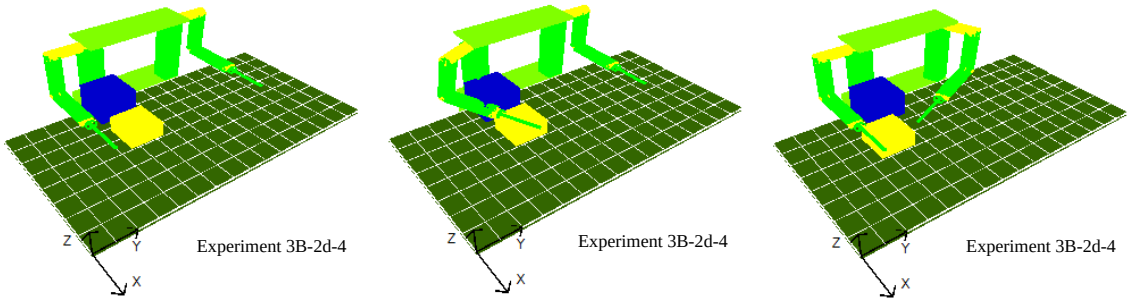


Figure 174 Simulation Results of Experiment 3B-2d-4

ISAC finds that it cannot push the box using both arms.

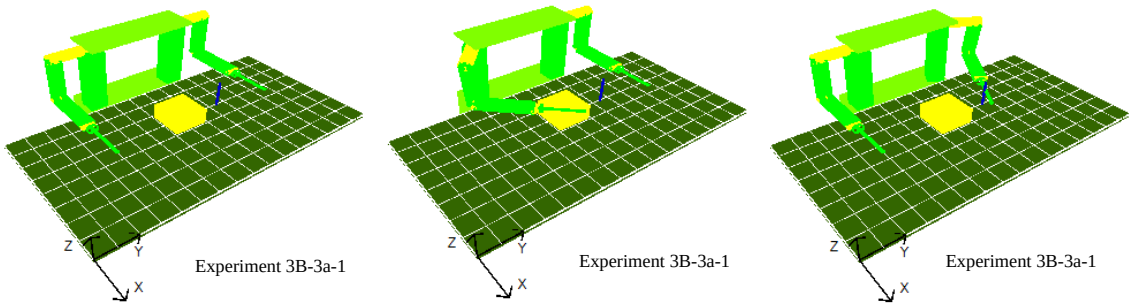


Figure 175 Simulation Results of Experiment 3B-3a-1

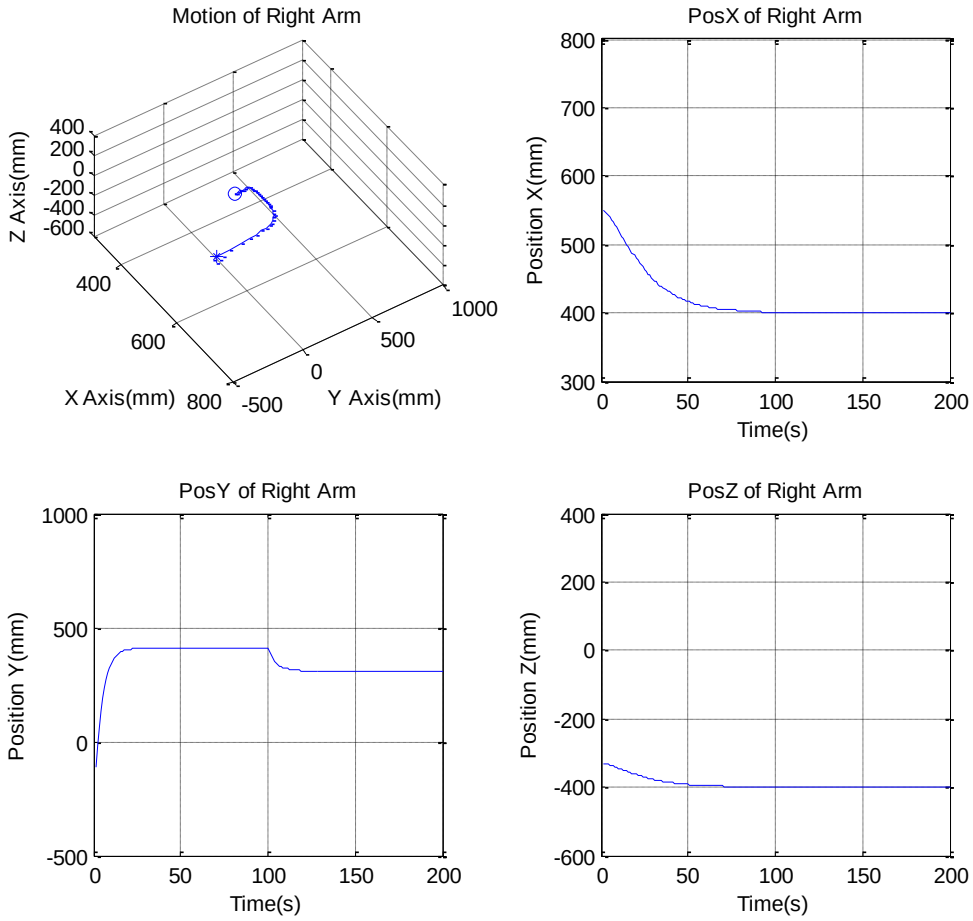


Figure 176 Generated Motion Trajectories of the Right Arm in Experiment 3B-3a-1

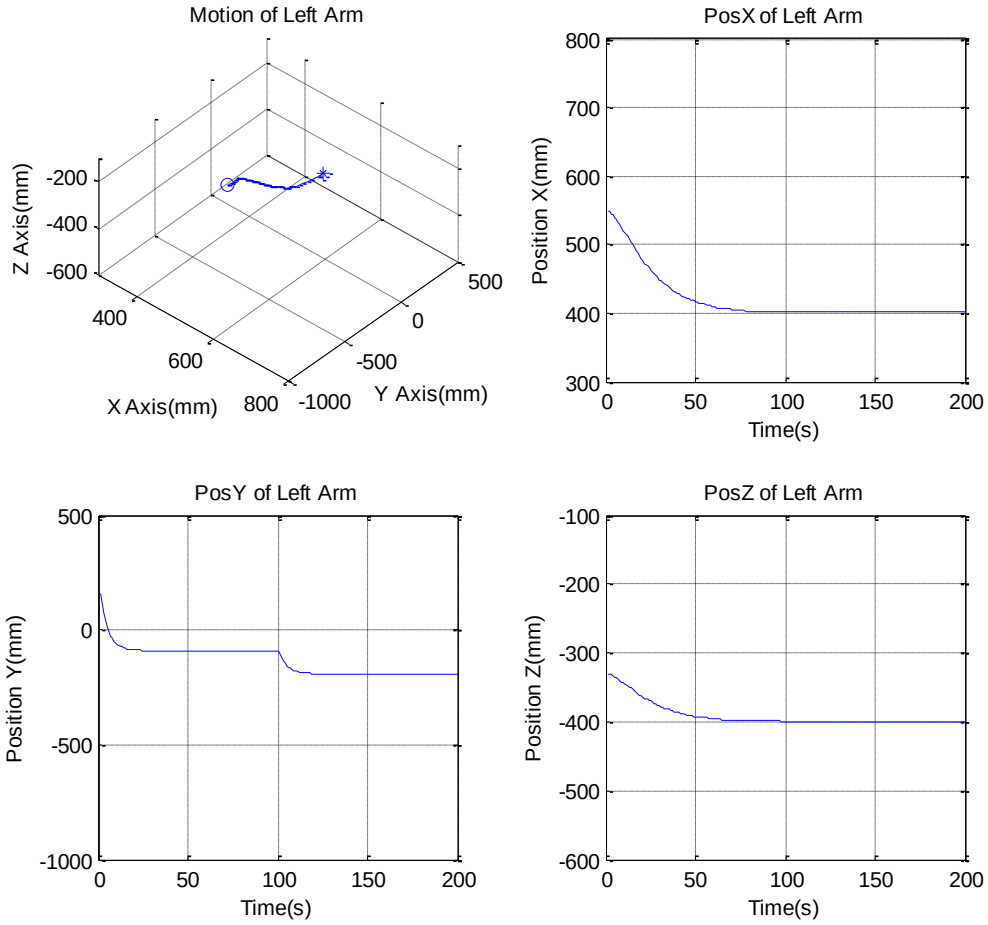


Figure 177 Generated Motion Trajectories of the Left Arm in Experiment 3B-3a-1

ISAC finds that it cannot push the box using both arms in Experiment 3B-3a-2.

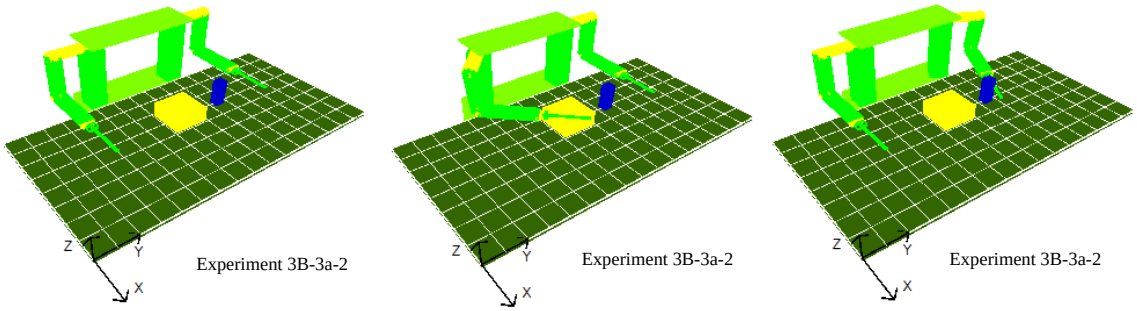


Figure 178 Simulation Results of Experiment 3B-3a-2

ISAC finds that it cannot push the box using both arms in Experiment 3B-3a-3.

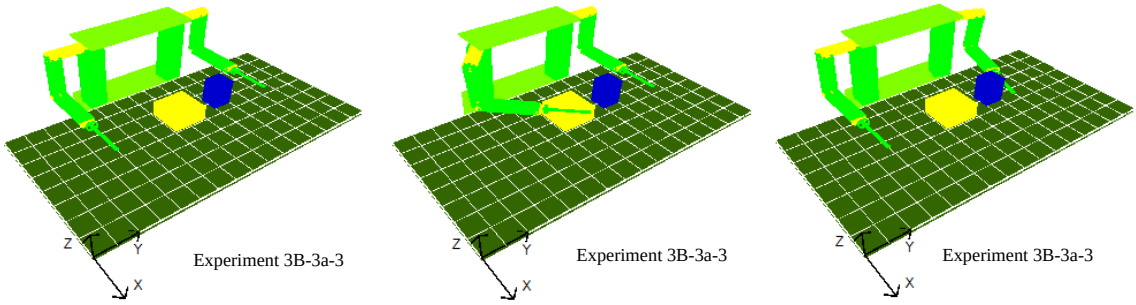


Figure 179 Simulation Results of Experiment 3B-3a-3

ISAC finds that it cannot push the box using both arms in Experiment 3B-3a-4.

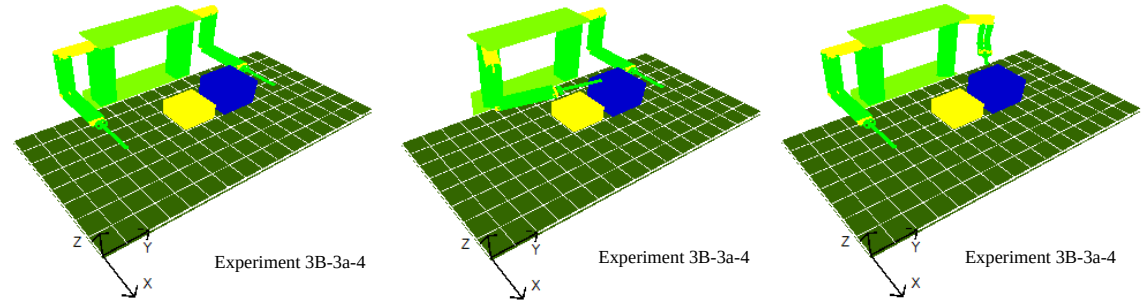


Figure 180 Simulation Results of Experiment 3B-3a-4

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-3b-1. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

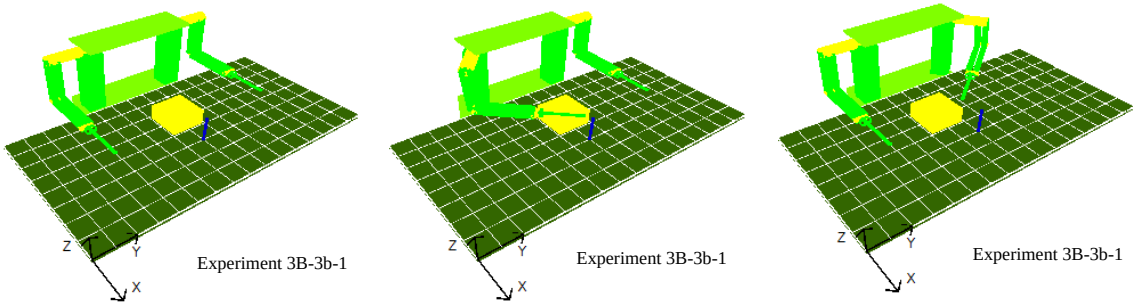


Figure 181 Simulation Results of Experiment 3B-3b-1

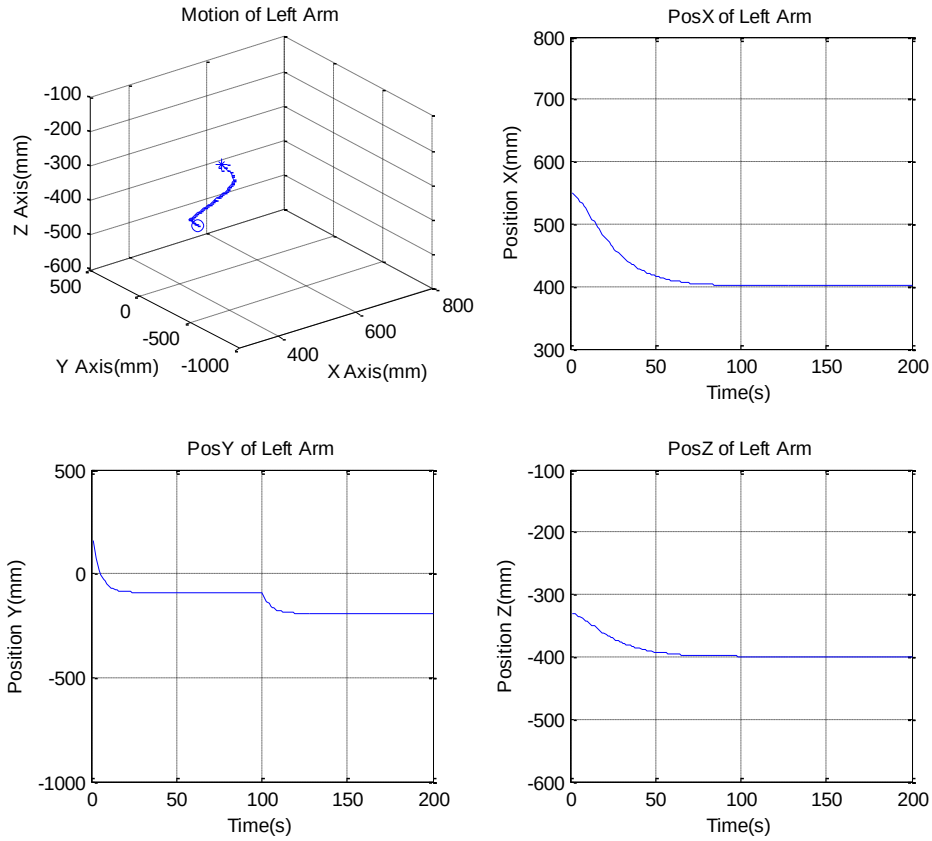


Figure 182 Generated Motion Trajectories of the Right Arm in Experiment 3B-3b-1

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-3b-2. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

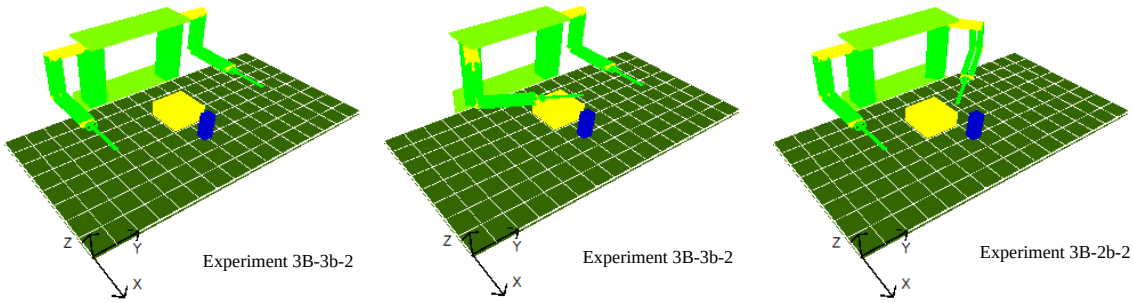


Figure 183 Simulation Results of Experiment 3B-3b-2

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-3b-3. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

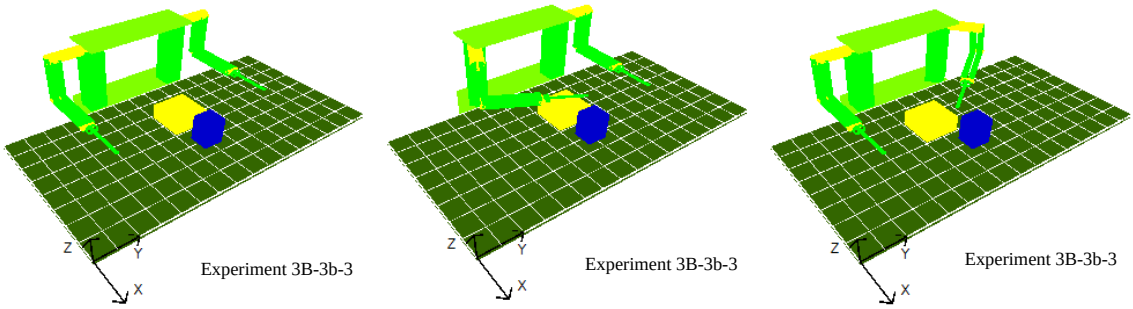


Figure 184 Simulation Results of Experiment 3B-3b-3

ISAC finds that it cannot push the box using either of the arms.

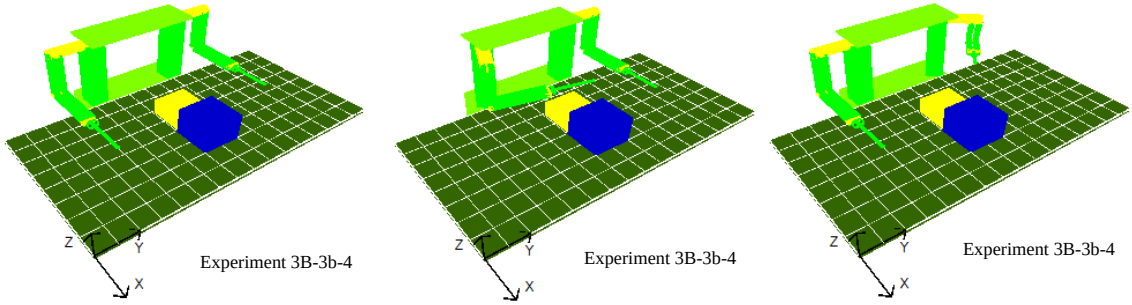


Figure 185 Simulation Results of Experiment 3B-3b-4

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-3c-1. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

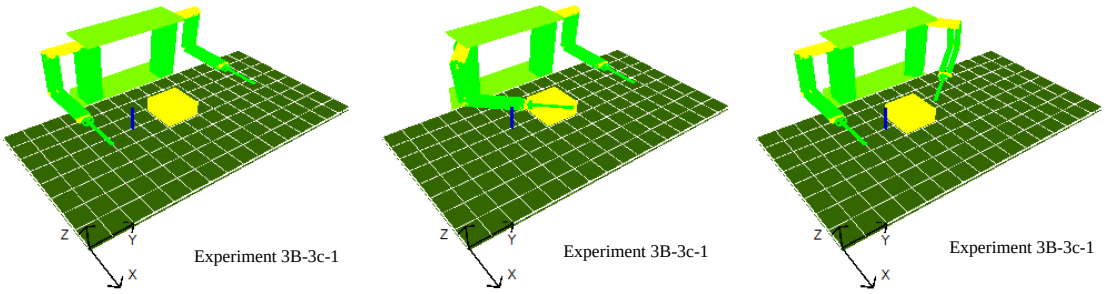


Figure 186 Simulation Results of Experiment 3B-3c-1

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-3c-2. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

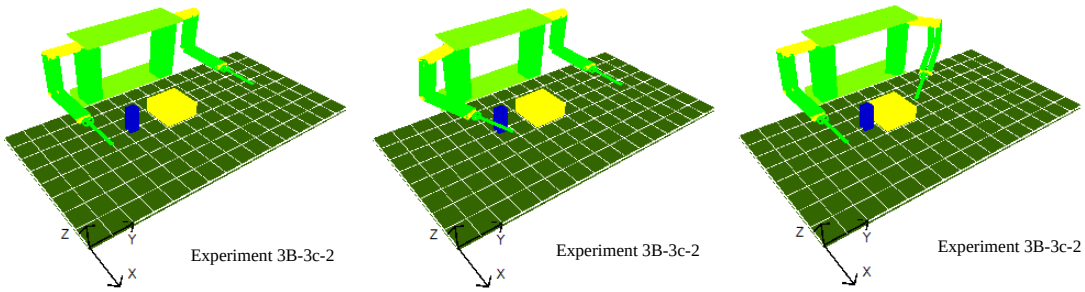


Figure 187 Simulation Results of Experiment 3B-3c-2

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-3c-3. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

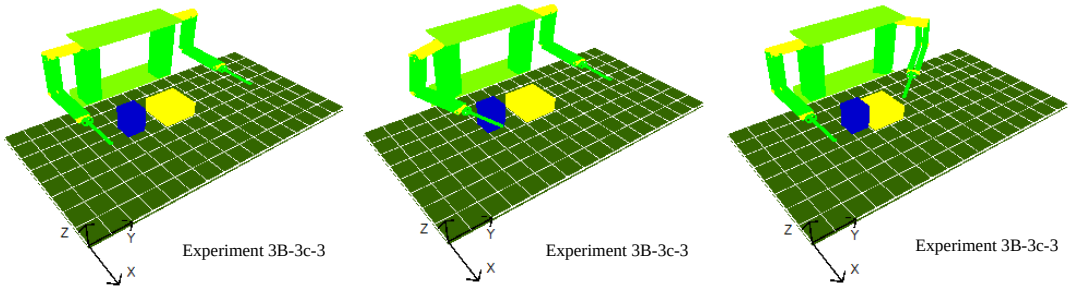


Figure 188 Simulation Results of Experiment 3B-3c-3

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-3c-4. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

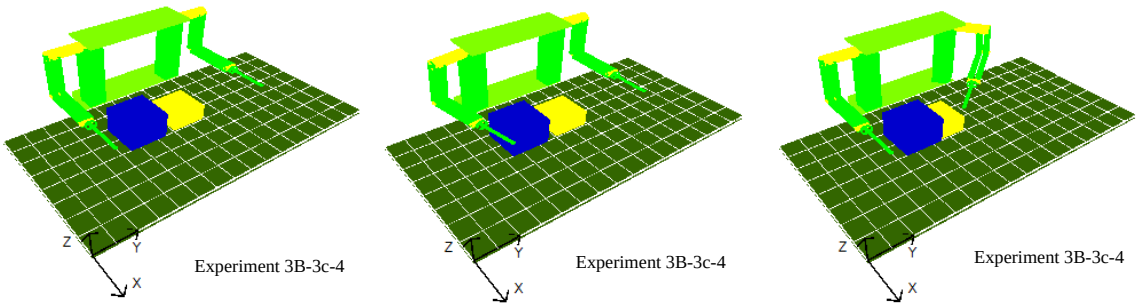


Figure 189 Simulation Results of Experiment 3B-3c-4

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-3d-1. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

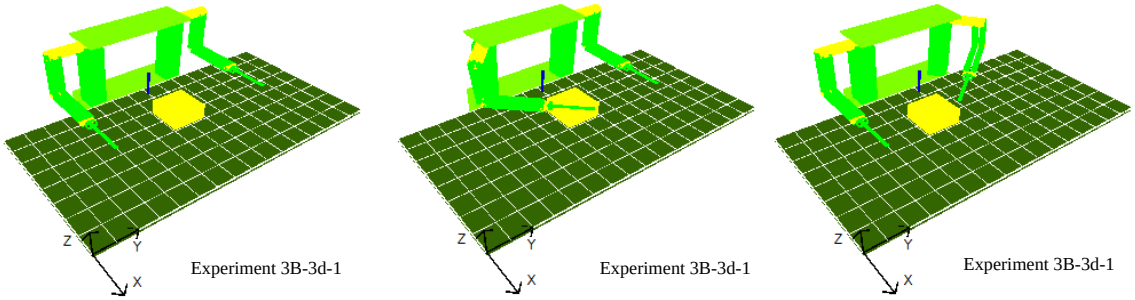


Figure 190 Simulation Results of Experiment 3B-3d-1

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-3d-2. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

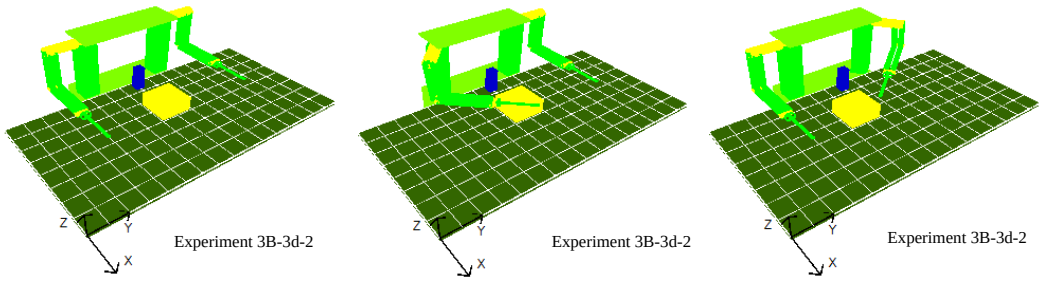


Figure 191 Simulation Results of Experiment 3B-3d-2

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-3d-3. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

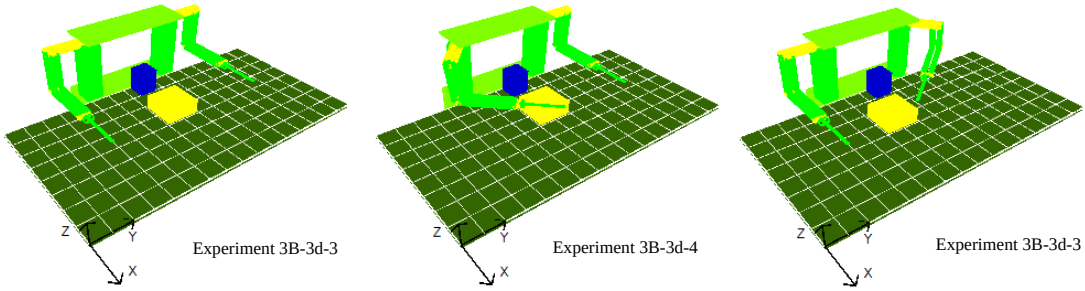


Figure 192 Simulation Results of Experiment 3B-3d-3

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-3d-4. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

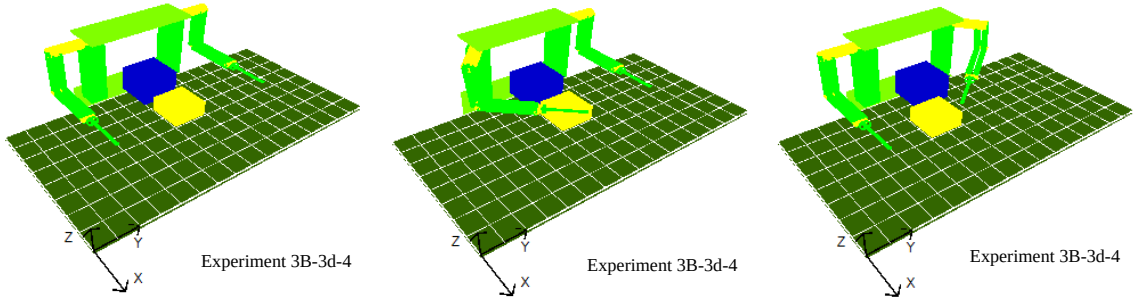


Figure 193 Simulation Results of Experiment 3B-3d-4

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-4a-1. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

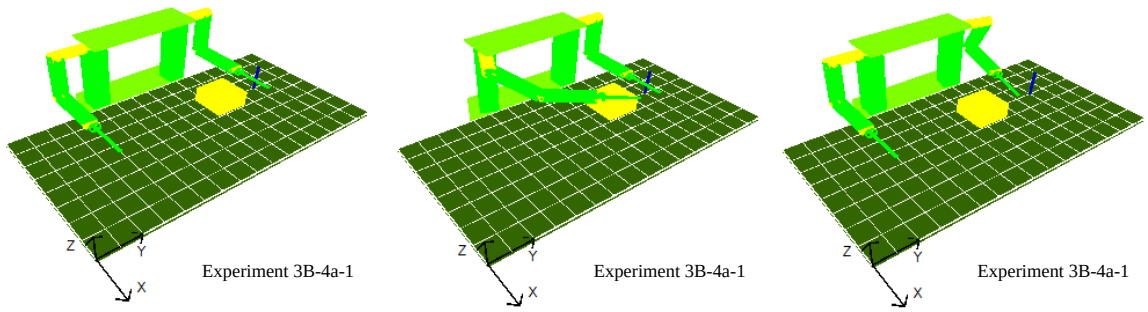


Figure 194 Simulation Results of Experiment 3B-4a-1

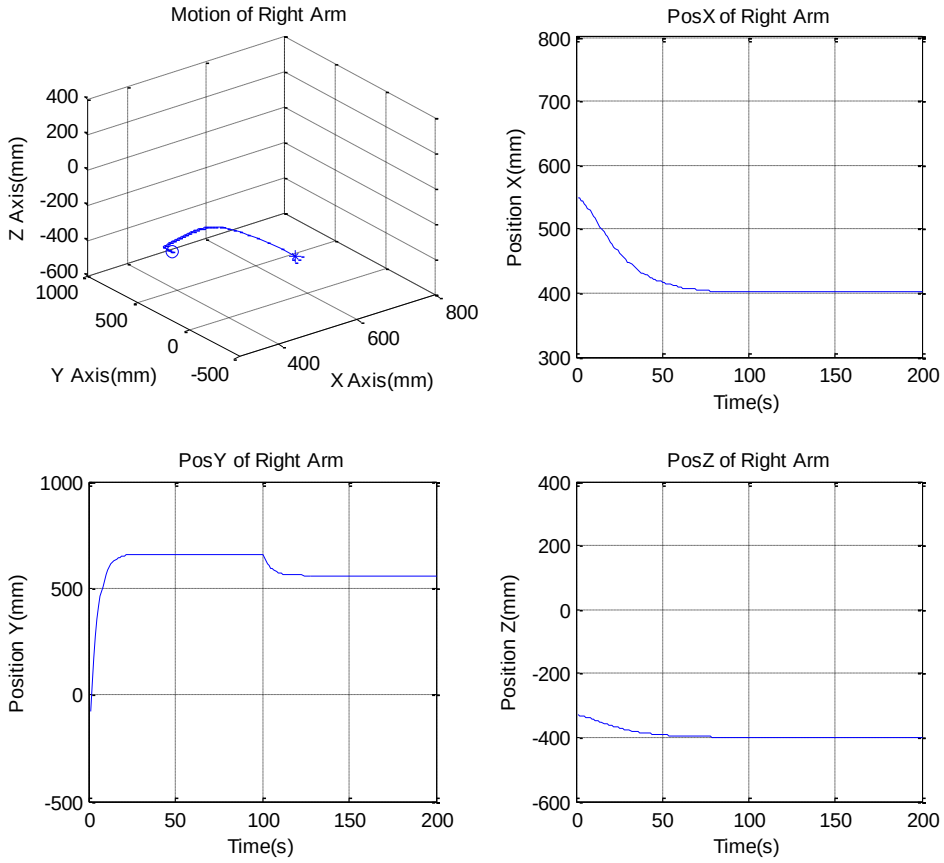


Figure 195 Generated Motion Trajectories of the Right Arm in Experiment 3B-4a-1

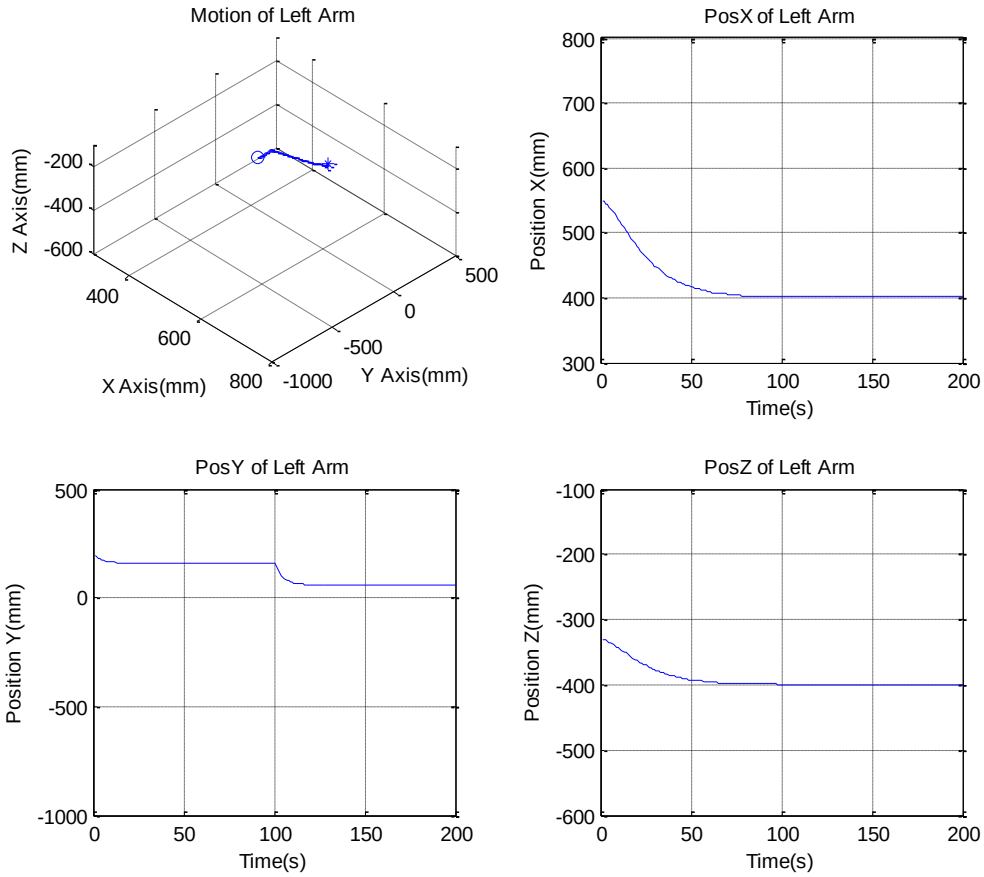


Figure 196 Generated Motion Trajectories of the Left Arm in Experiment 3B-4a-1

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-4a-2. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

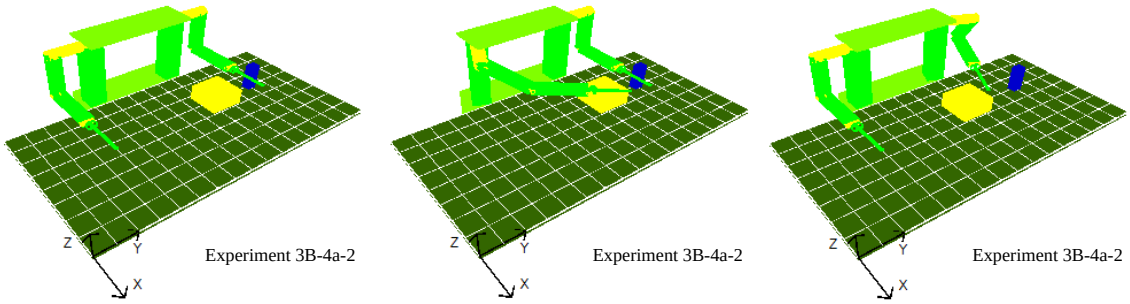


Figure 197 Simulation Results of Experiment 3B-4a-2

ISAC finds that it cannot push the box to the right using either of the arms.

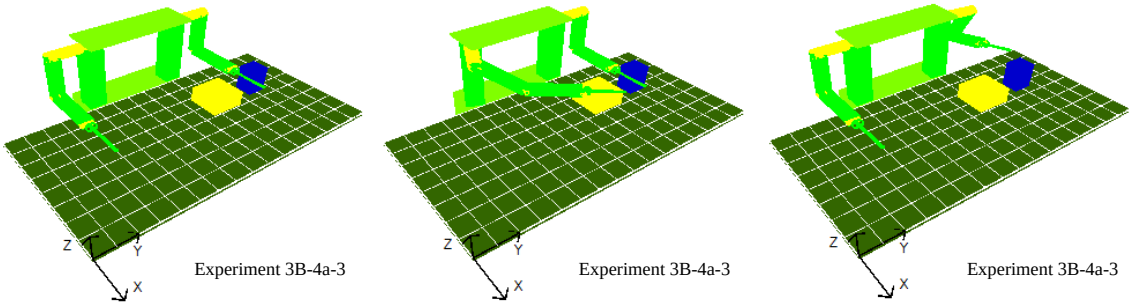


Figure 198 Simulation Results of Experiment 3B-4a-3

ISAC finds that it cannot push the box to the right using either of the arms.

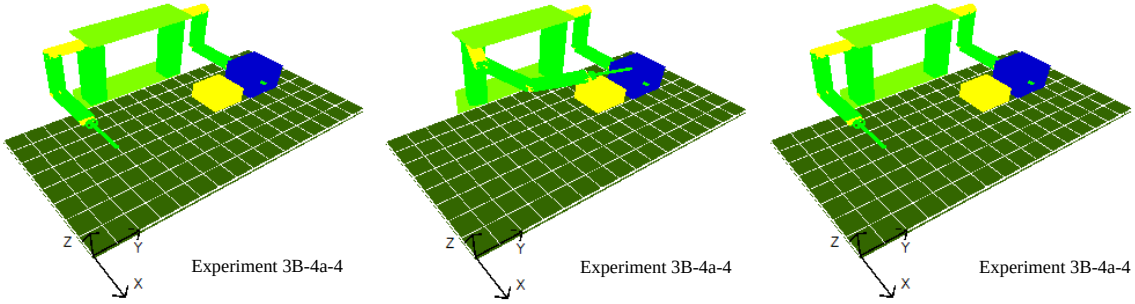


Figure 199 Simulation Results of Experiment 3B-4a-4

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-4b-1. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

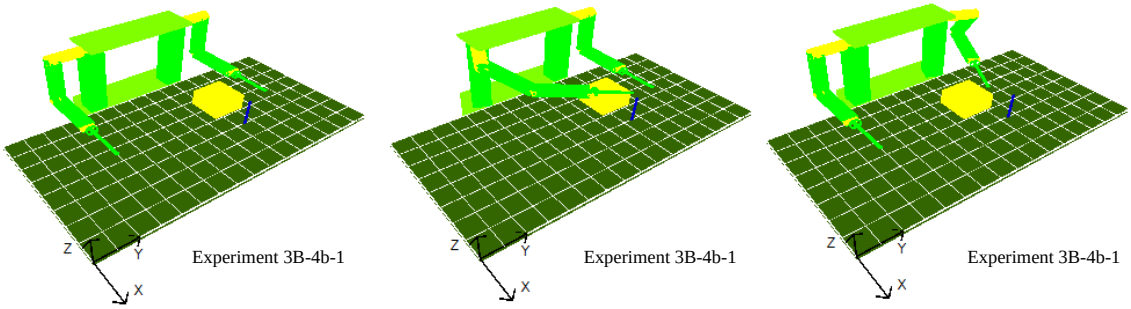


Figure 200 Simulation Results of Experiment 3B-4b-1

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-4b-2. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

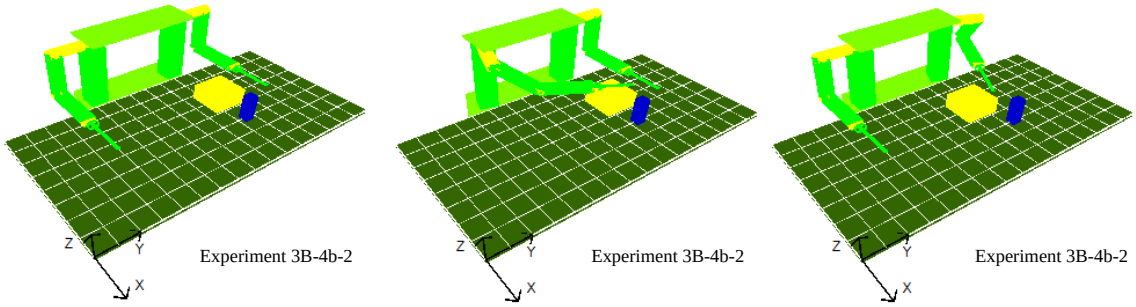


Figure 201 Simulation Results of Experiment 3B-4b-2

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-4b-3. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

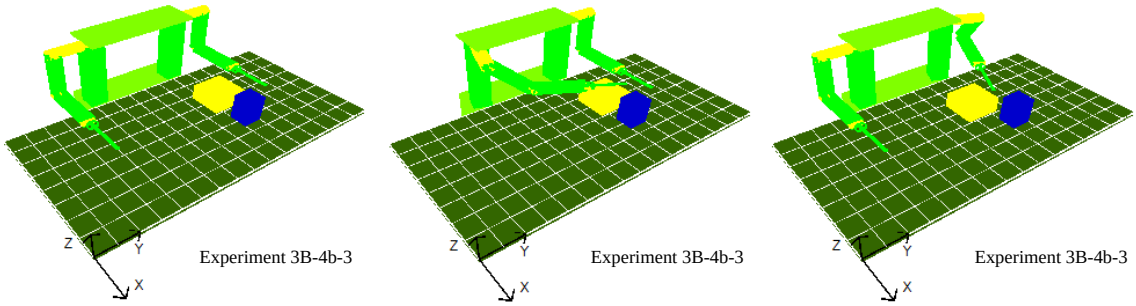


Figure 202 Simulation Results of Experiment 3B-4b-3

ISAC finds that it cannot push the box to the right either of the arms.

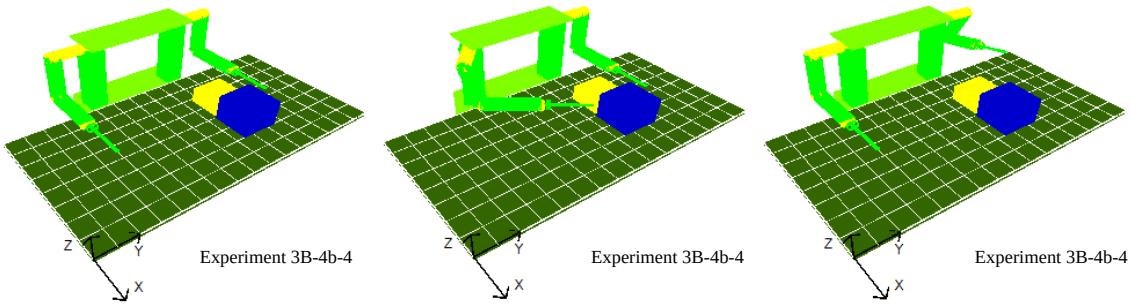


Figure 203 Simulation Results of Experiment 3B-4b-4

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-4c-1. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

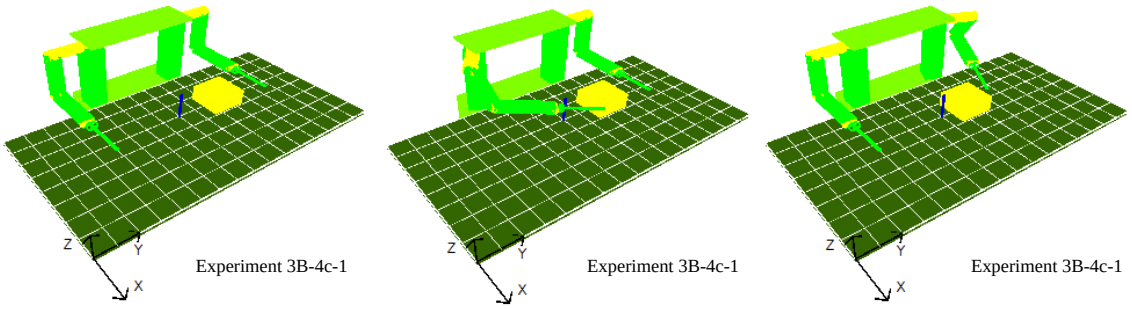


Figure 204 Simulation Results of Experiment 3B-4c-1

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-4c-2. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

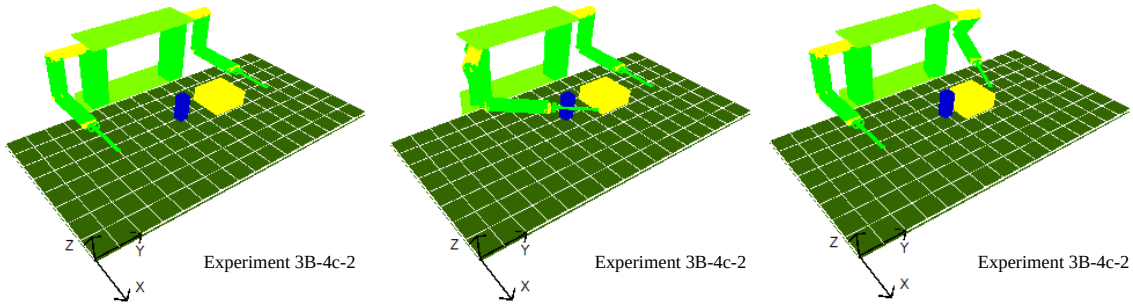


Figure 205 Simulation Results of Experiment 3B-4c-2

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-4c-3. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

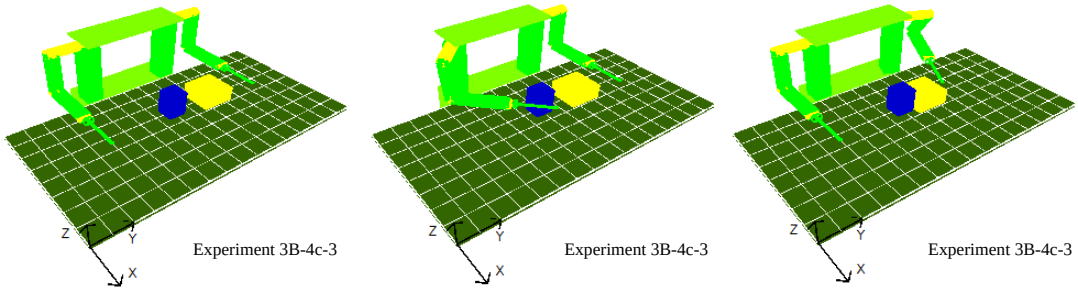


Figure 206 Simulation Results of Experiment 3B-4c-3

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-4c-4. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

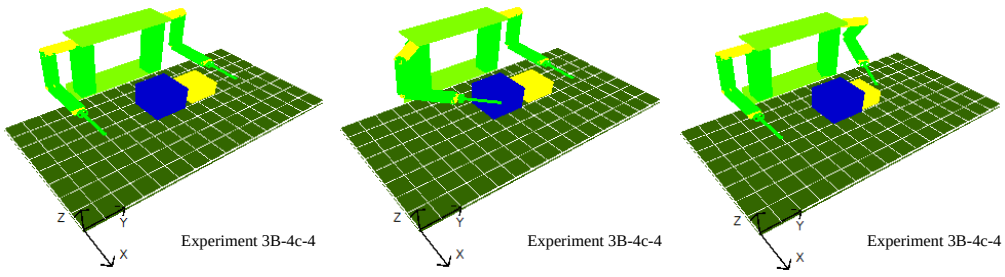


Figure 207 Simulation Results of Experiment 3B-4c-4

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-4d-1. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

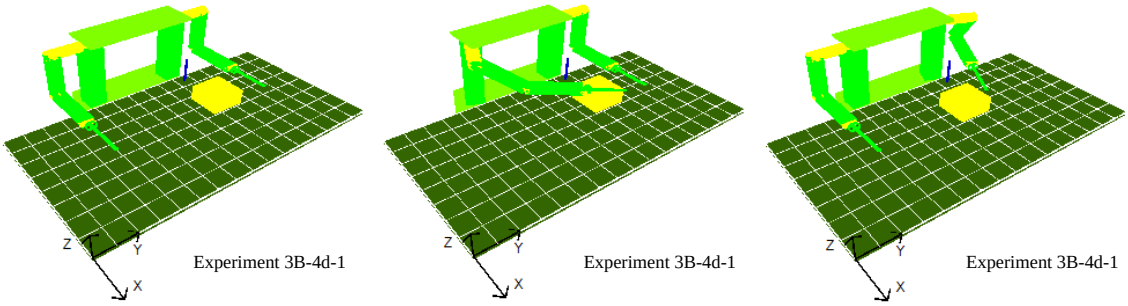


Figure 208 Simulation Results of Experiment 3B-4d-1

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-4d-2. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

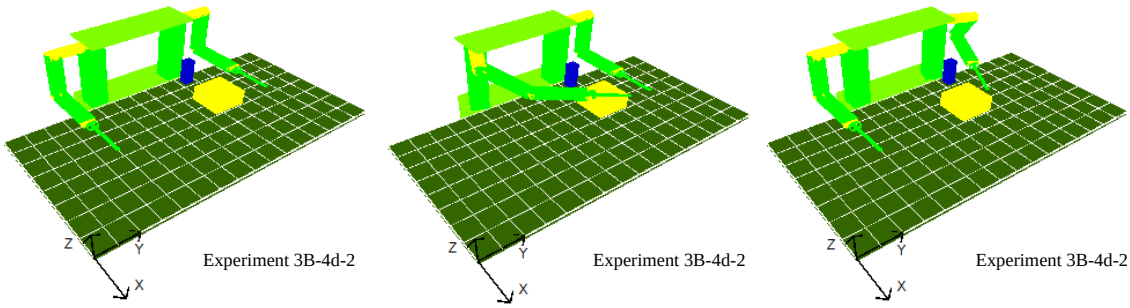


Figure 209 Simulation Results of Experiment 3B-4d-2

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-4d-3. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

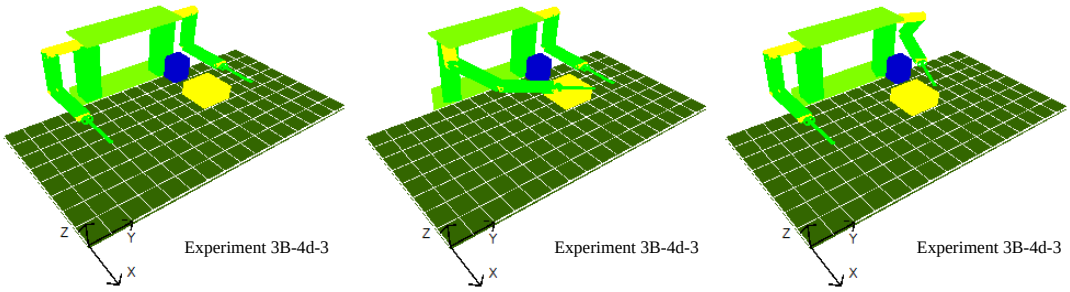


Figure 210 Simulation Results of Experiment 3B-4d-3

ISAC finds that it cannot push the box to the right using its right arm in Experiment 3B-4d-4. Then it switches the generated behavior sequence to the left arm and pushes the box to the right.

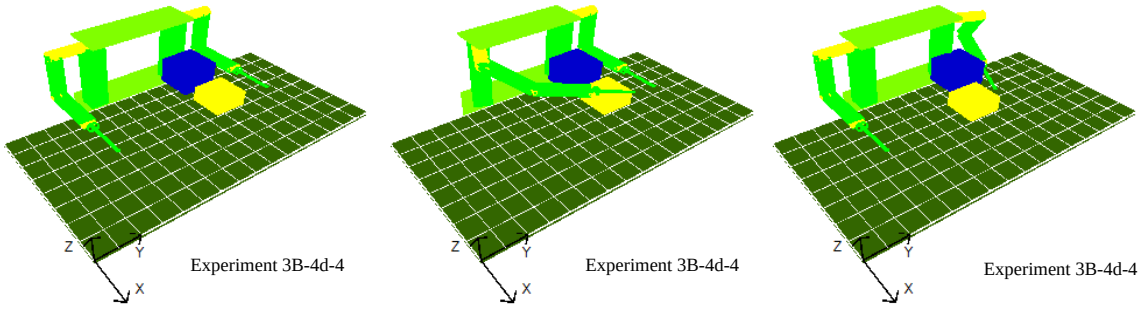


Figure 211 Simulation Results of Experiment 3B-4d-4

ISAC pushes the box to the right using its right arm in Experiment 3B-5a-1.

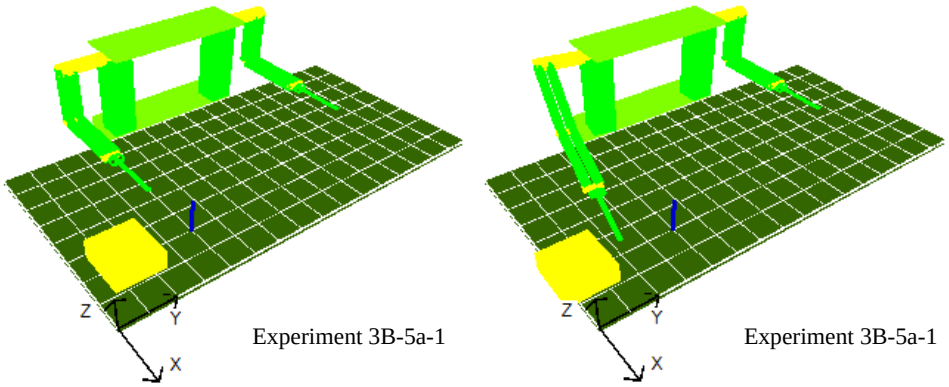


Figure 212 Simulation Results of Experiment 3B-5a-1

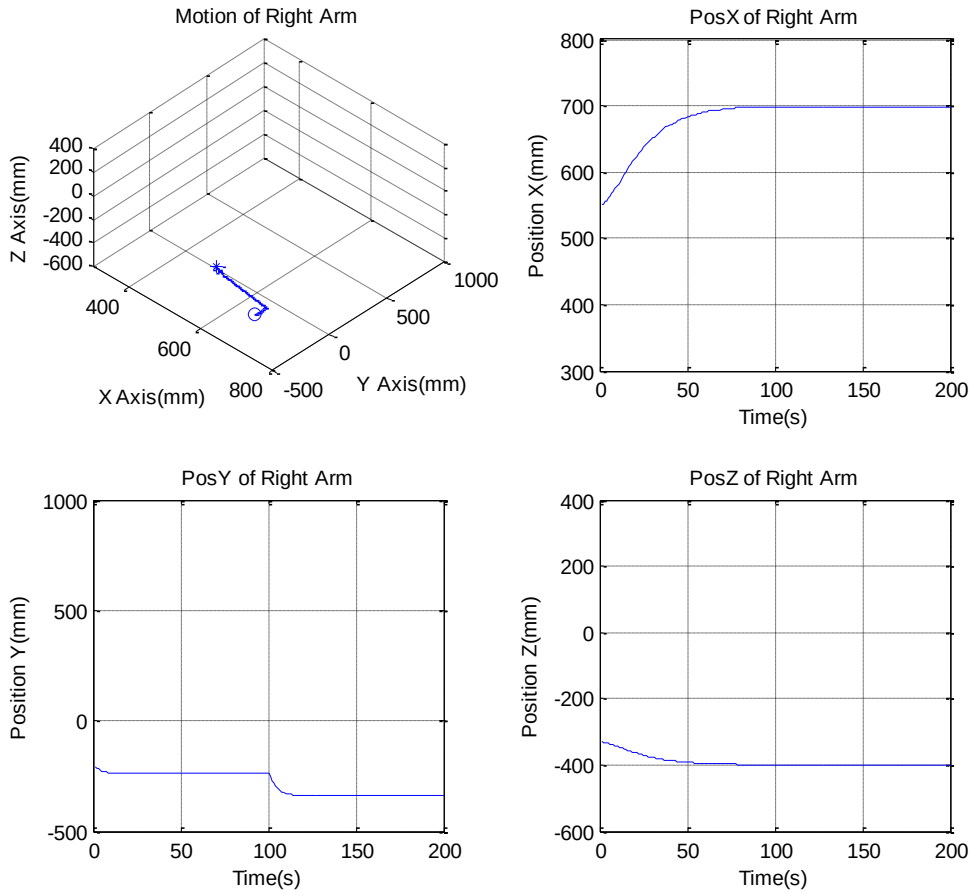


Figure 213 Generated Motion Trajectories of the Right Arm in Experiment 3B-5a-1

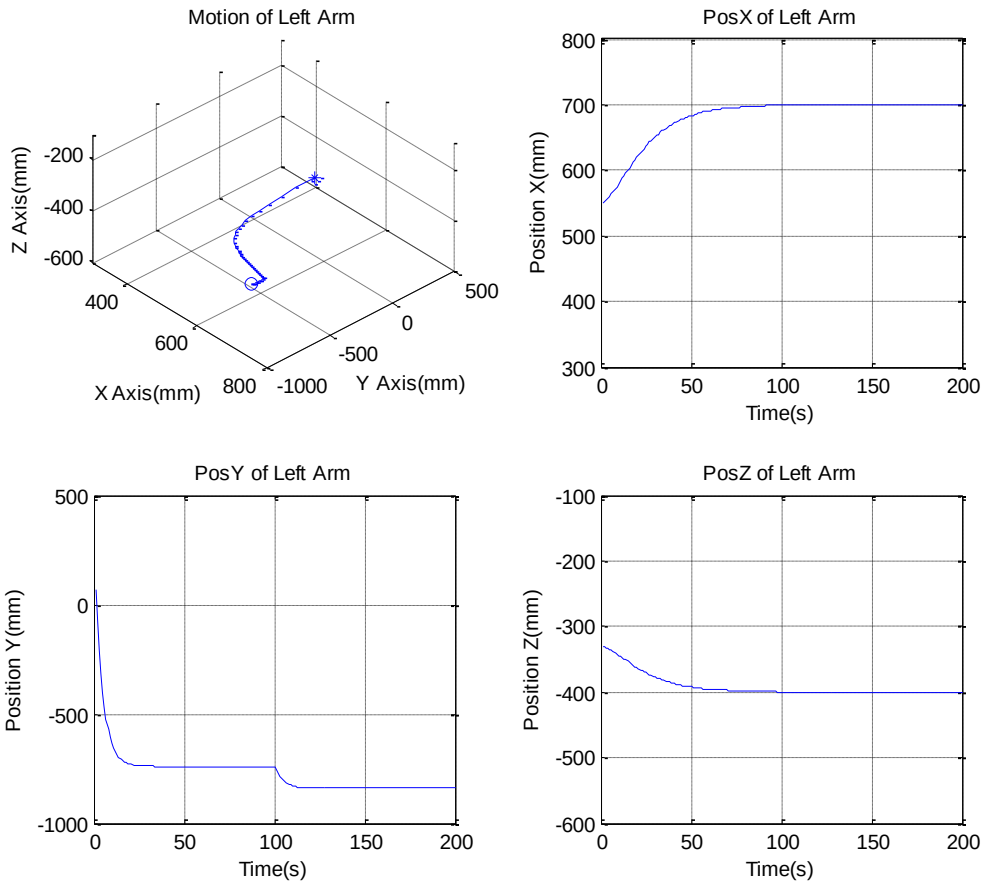


Figure 214 Generated Motion Trajectories of the Left Arm in Experiment 3B-5a-1

ISAC pushes the box to the right using its right arm in Experiment 3B-5a-2.

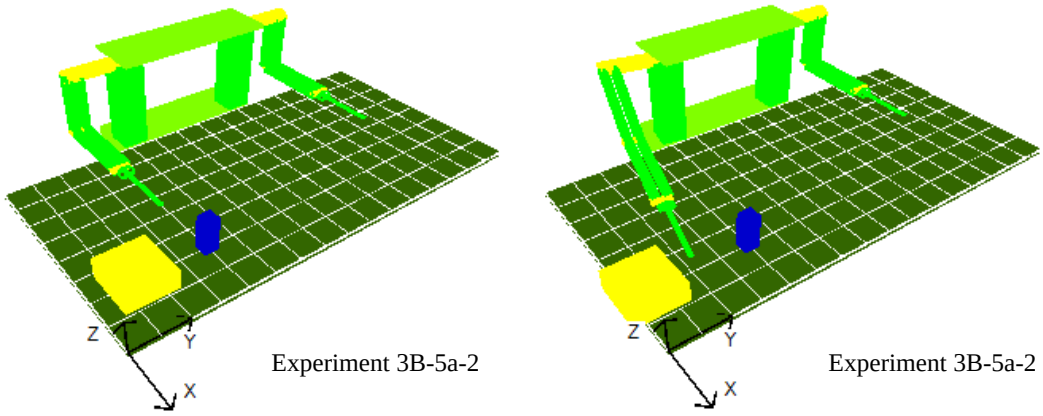


Figure 215 Simulation Results of Experiment 3B-5a-2

ISAC finds that it cannot push the box to the right using either of the arms in Experiment 3B-5a-3.

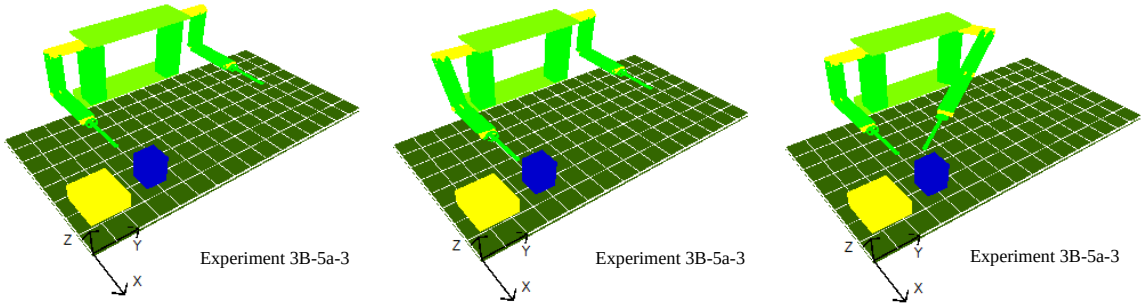


Figure 216 Simulation Results of Experiment 3B-5a-3

ISAC finds that it cannot push the box to the right using either of the arms in Experiment 3B-5a-3.

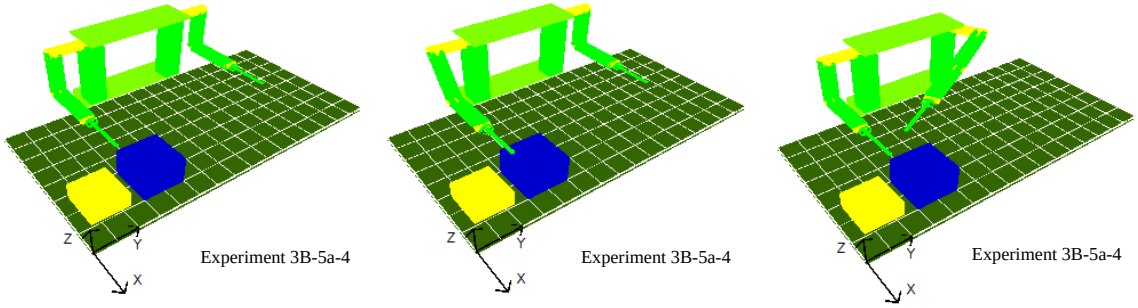


Figure 217 Simulation Results of Experiment 3B-5a-4

ISAC pushes the box to the right using its right arm in Experiment 3B-5b-1.

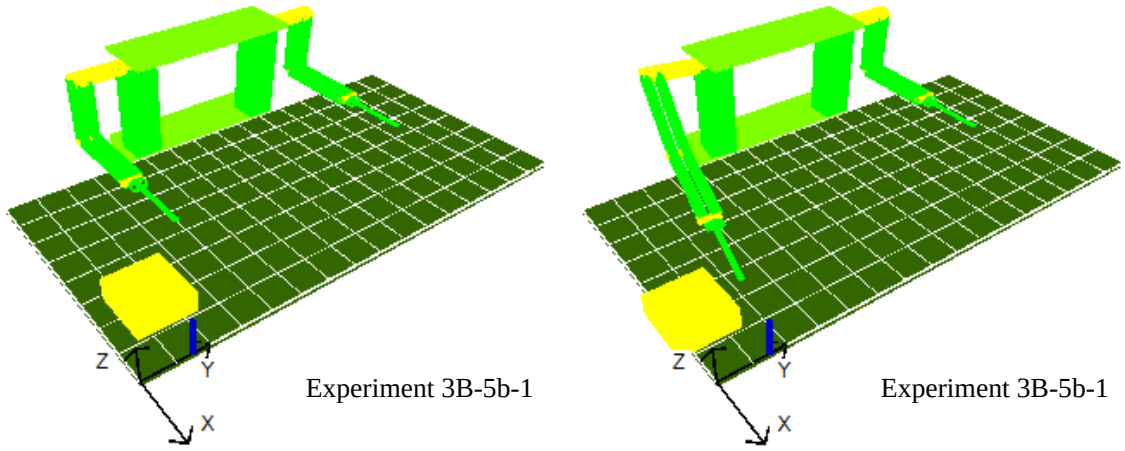


Figure 218 Simulation Results of Experiment 3B-5b-1

ISAC pushes the box to the right using its right arm in Experiment 3B-5b-2.

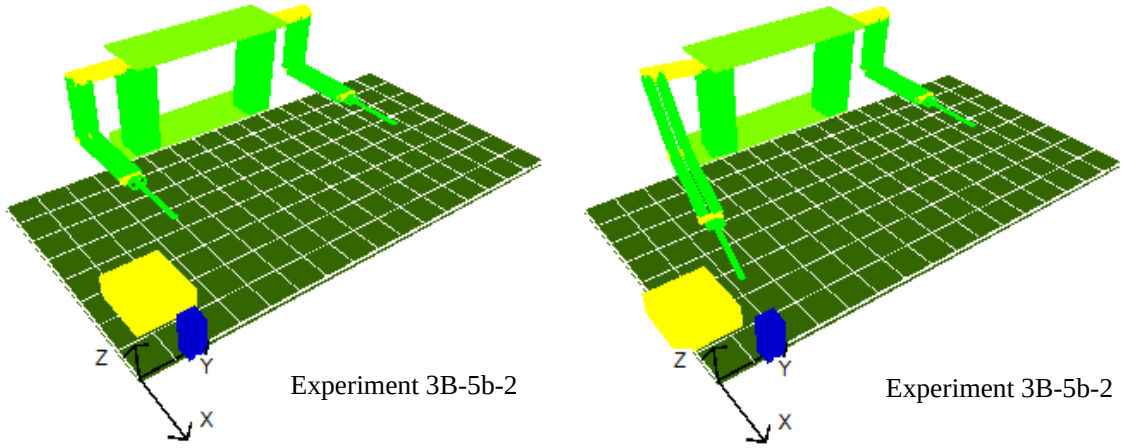


Figure 219 Simulation Results of Experiment 3B-5b-2

ISAC pushes the box to the right using its right arm in Experiment 3B-5b-3.

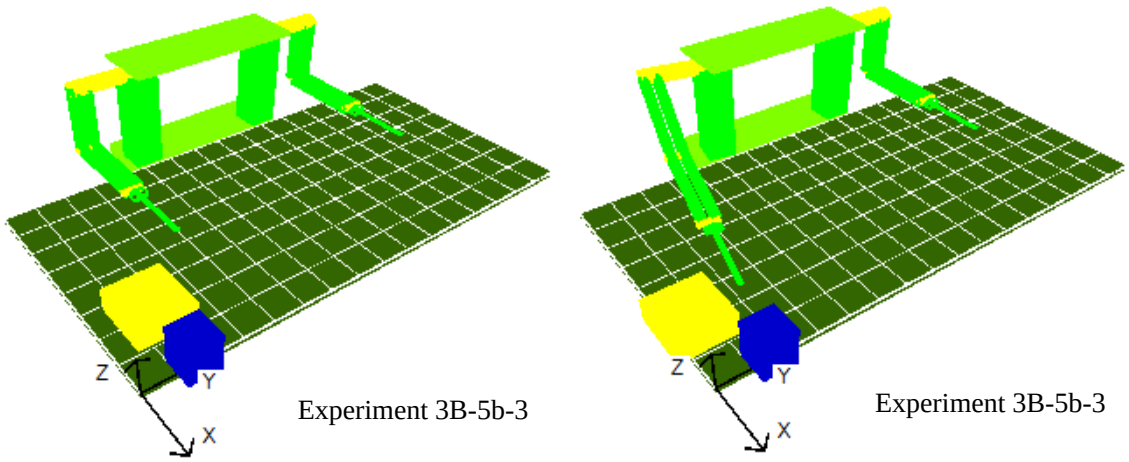


Figure 220 Simulation Results of Experiment 3B-5b-3

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-5b-4. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

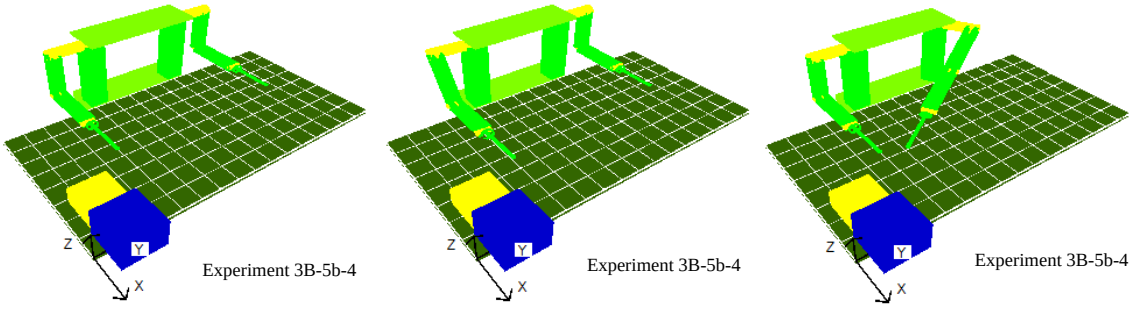


Figure 221 Simulation Results of Experiment 3B-5b-4

ISAC pushes the box to the right using its right arm in Experiment 3B-5c-1.

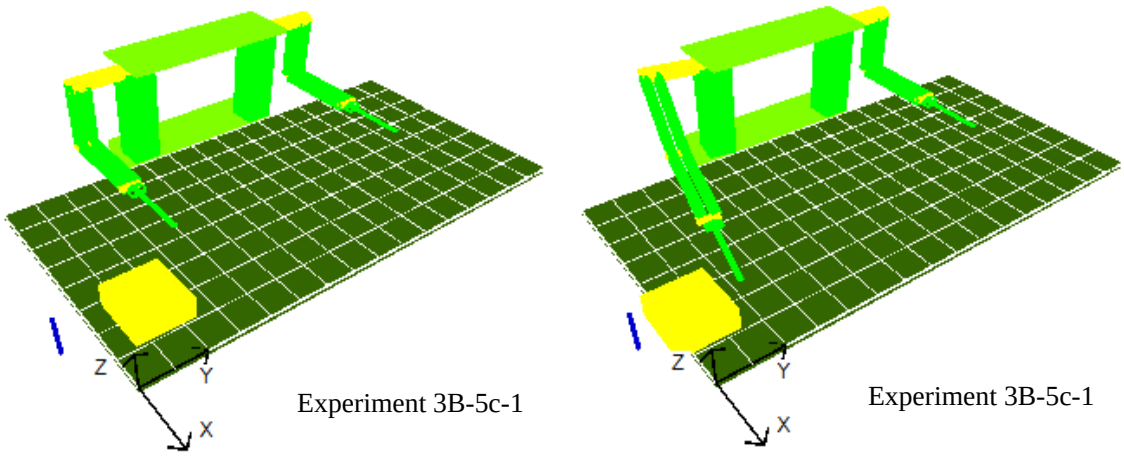


Figure 222 Simulation Results of Experiment 3B-5c-1

ISAC pushes the box to the right using its right arm in Experiment 3B-5c-2.

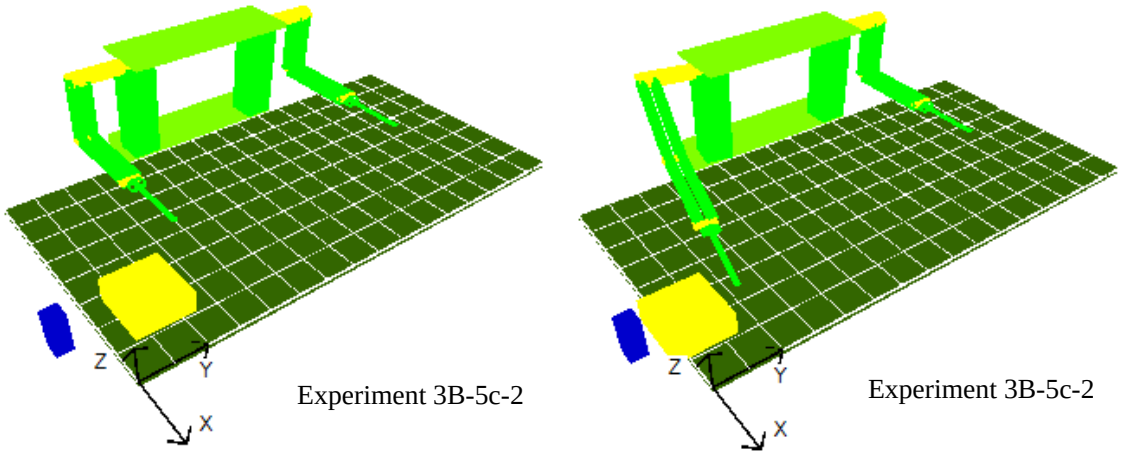


Figure 223 Simulation Results of Experiment 3B-5c-2

ISAC pushes the box to the right using its right arm in Experiment 3B-5c-3.

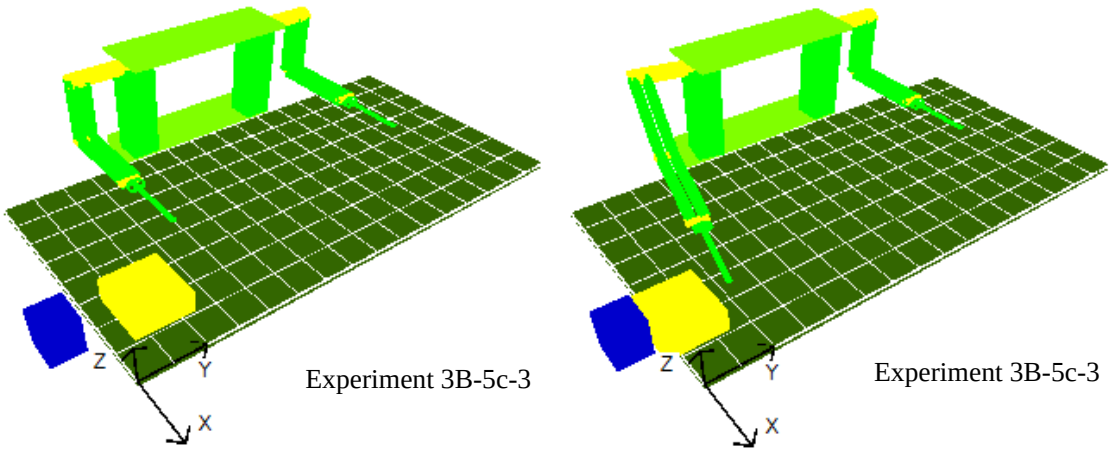


Figure 224 Simulation Results of Experiment 3B-5c-3

ISAC pushes the box to the right using its right arm in Experiment 3B-5c-4.

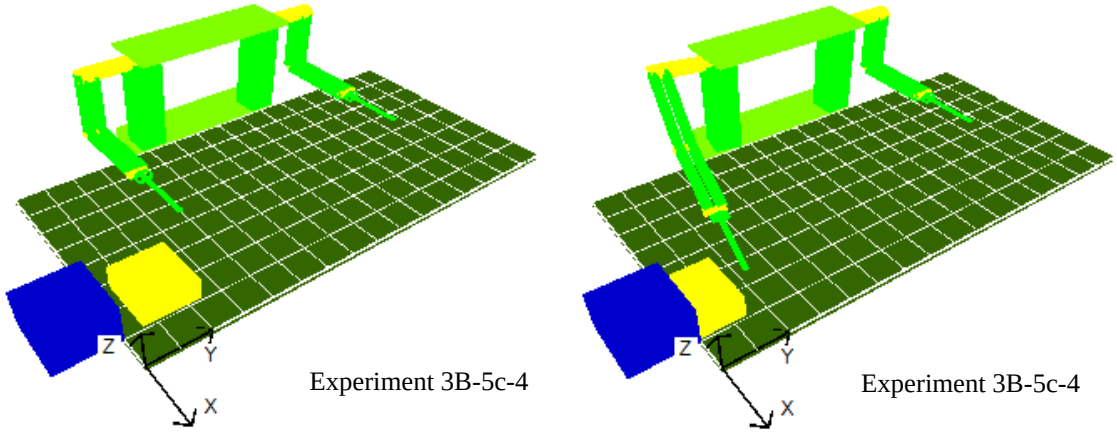


Figure 225 Simulation Results of Experiment 3B-5c-4

ISAC pushes the box to the right using its right arm in Experiment 3B-5d-1.

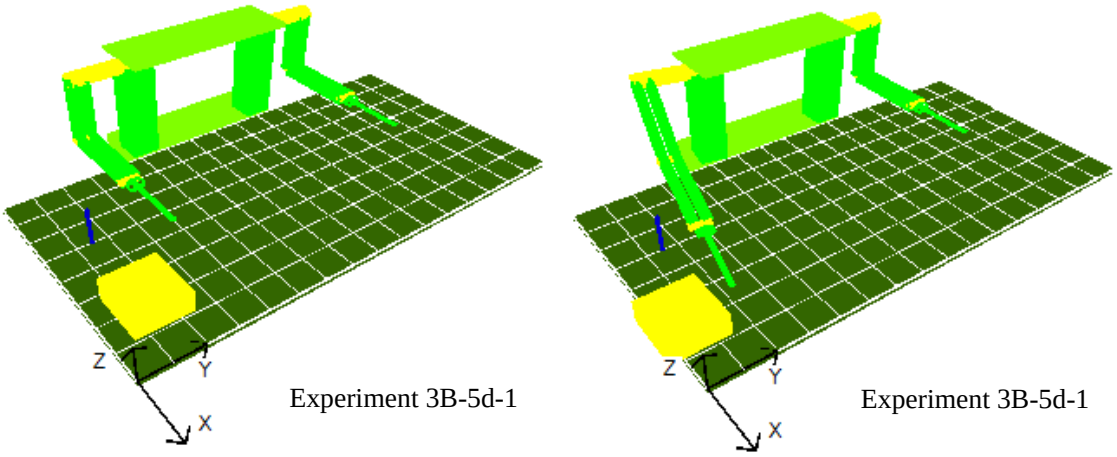


Figure 226 Simulation Results of Experiment 3B-5d-1

ISAC pushes the box to the right using its right arm in Experiment 3B-5d-2.

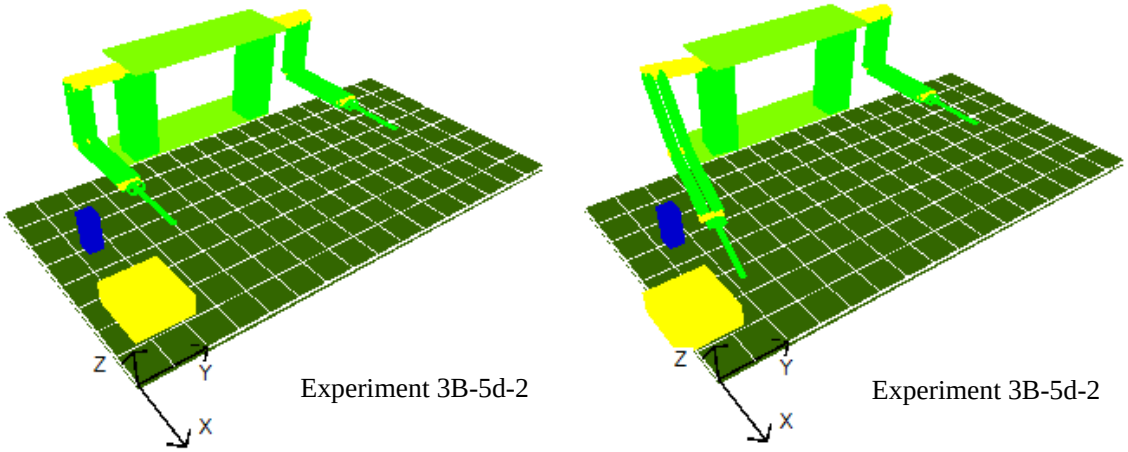


Figure 227 Simulation Results of Experiment 3B-5d-2

ISAC pushes the box to the right using its right arm in Experiment 3B-5d-3.

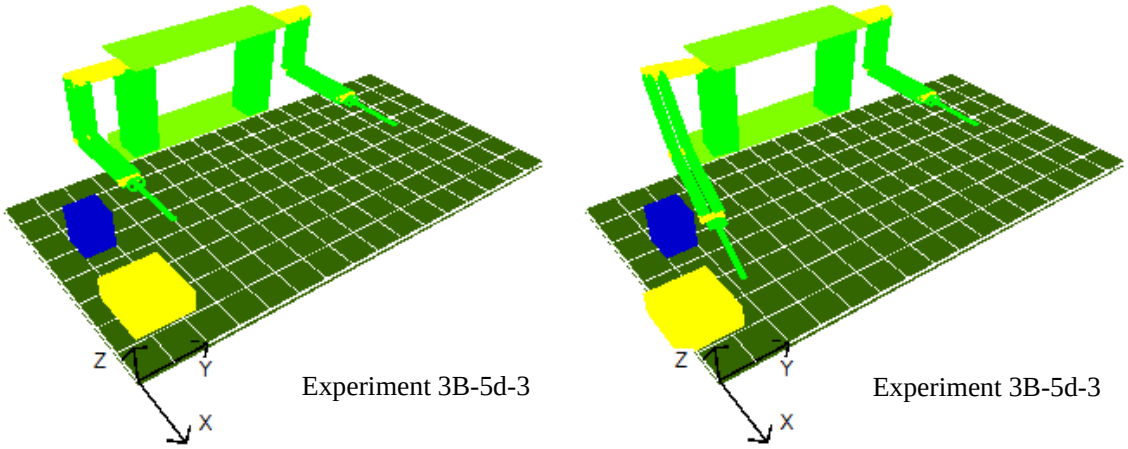


Figure 228 Simulation Results of Experiment 3B-5d-3

ISAC finds that it cannot push the box to the right using either of the arms in Experiment 3B-5d-4.

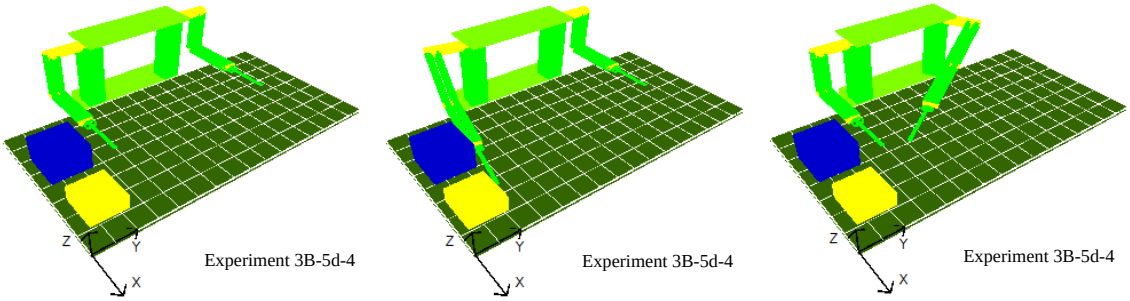


Figure 229 Simulation Results of Experiment 3B-5d-4

ISAC pushes the box to the right using its right arm in Experiment 3B-6a-1.

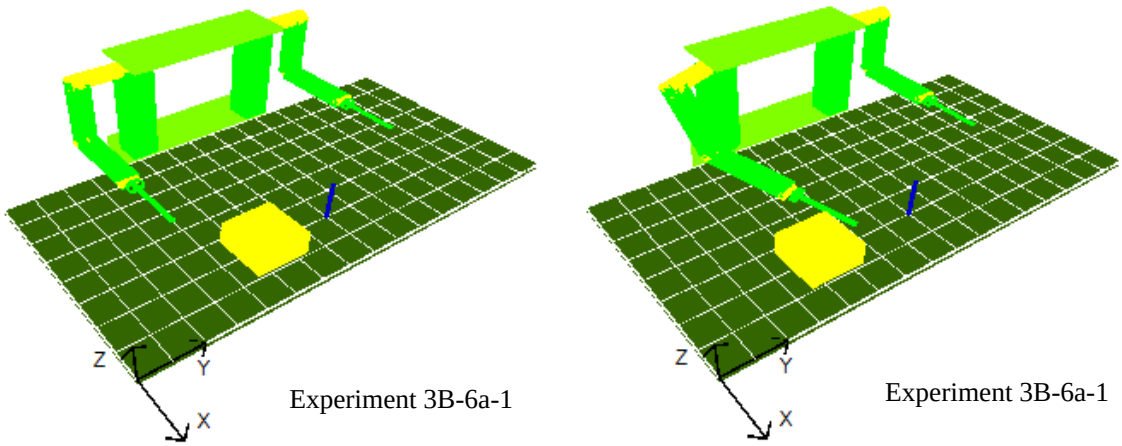


Figure 230 Simulation Results of Experiment 3B-6a-1

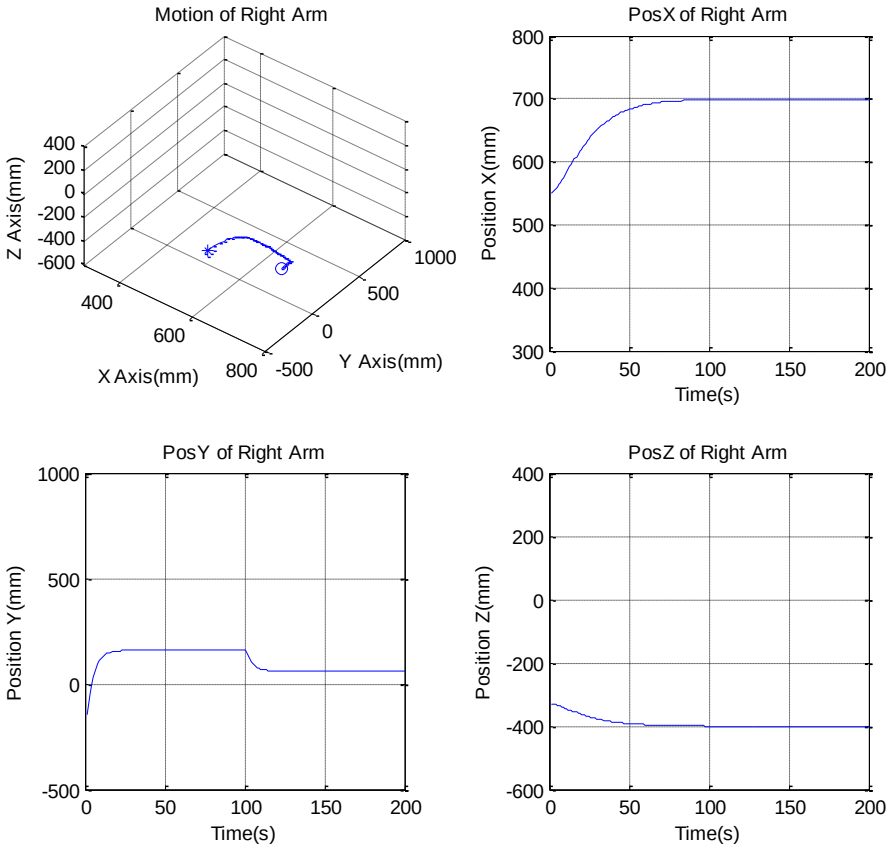


Figure 231 Generated Motion Trajectories of the Right Arm in Experiment 3B-6a-1

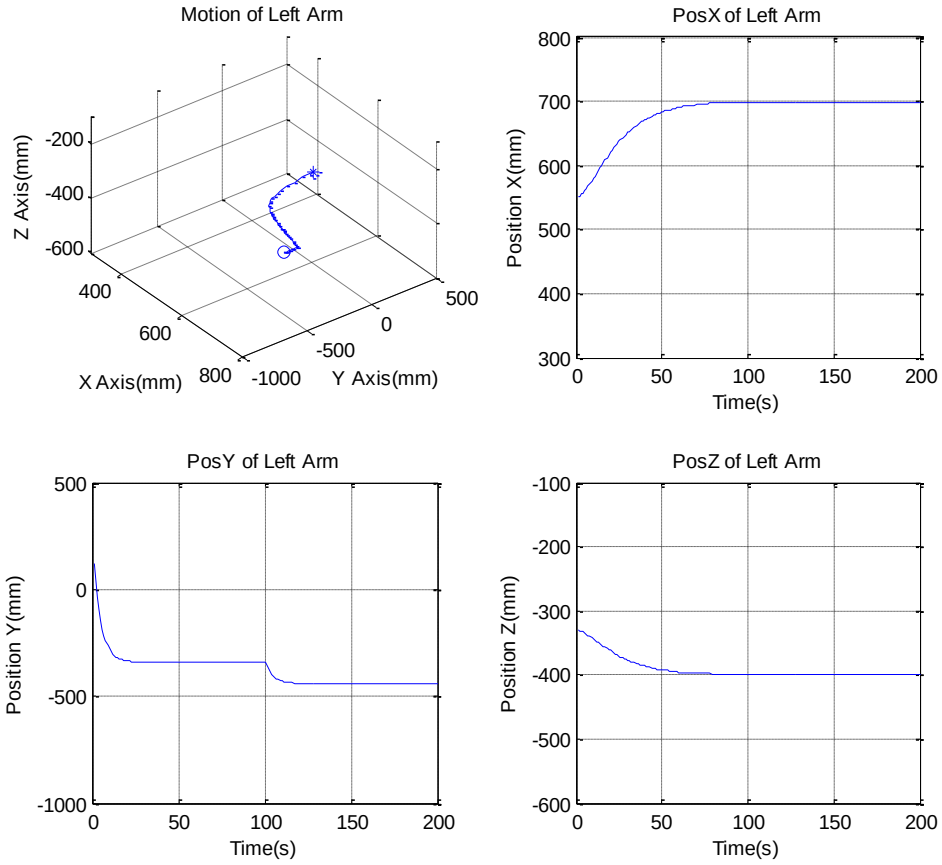


Figure 232 Generated Motion Trajectories of the Left Arm in Experiment 3B-6a-1.

ISAC pushes the box to the right using its right arm in Experiment 3B-6a-2.

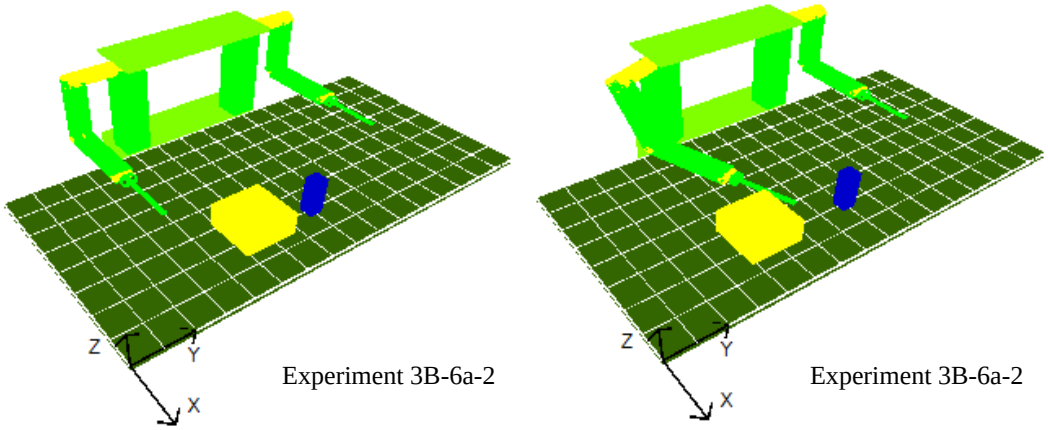


Figure 233 Simulation Results of Experiment 3B-6a-2

ISAC finds that it cannot push the box to the right either of the arms in Experiment 3B-6a-3.

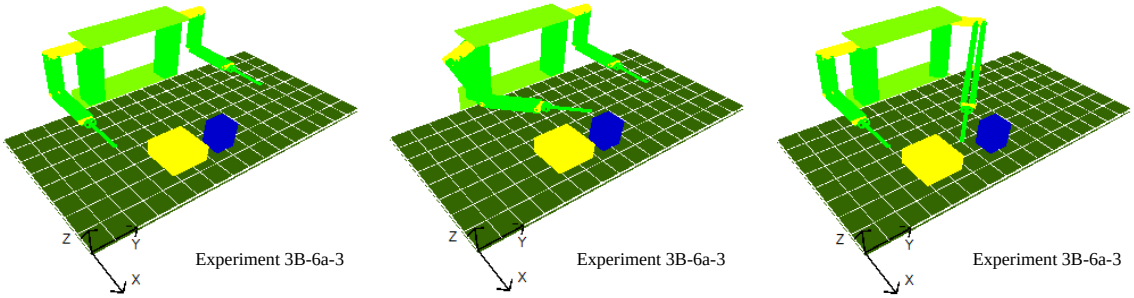


Figure 234 Simulation Results of Experiment 3B-6a-3

ISAC finds that it cannot push the box to the right using either of the arms in Experiment 3B-6a-4.

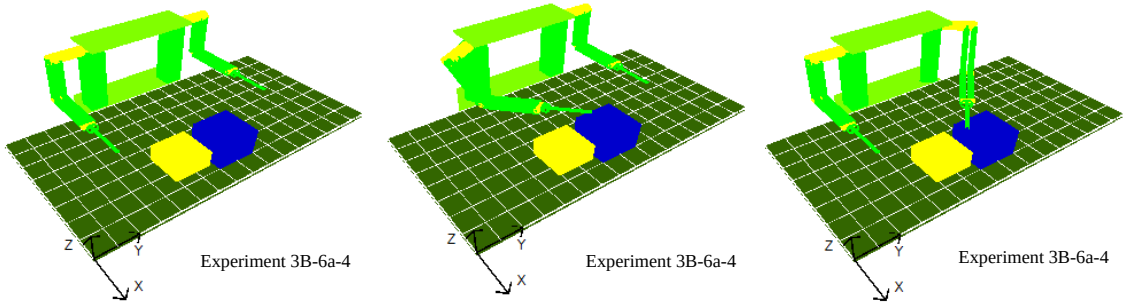


Figure 235 Simulation Results of Experiment 3B-6a-4

ISAC pushes the box to the right using its right arm in Experiment 3B-6b-1.

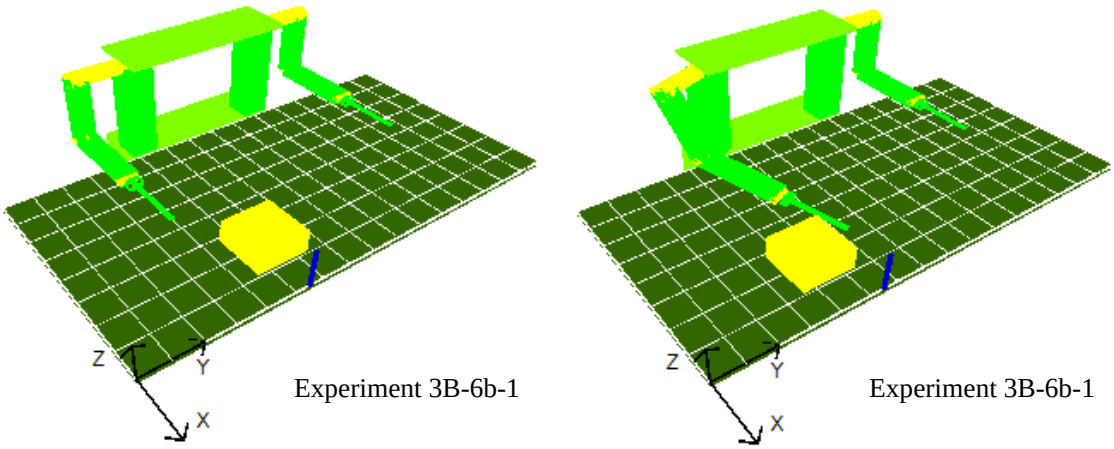


Figure 236 Simulation Results of Experiment 3B-6b-1

ISAC pushes the box to the right using its right arm in Experiment 3B-6b-2.

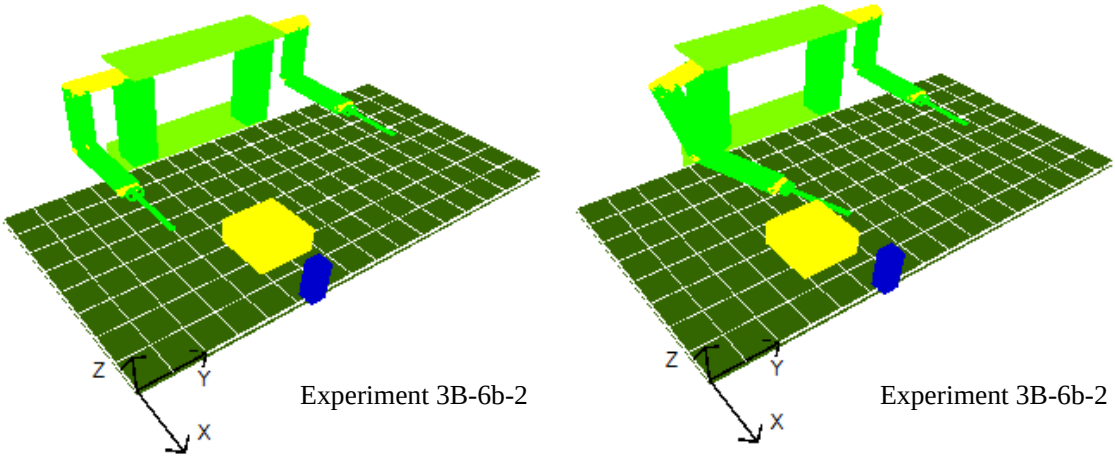


Figure 237 Simulation Results of Experiment 3B-6b-2

ISAC pushes the box to the right using its right arm in Experiment 3B-6b-3.

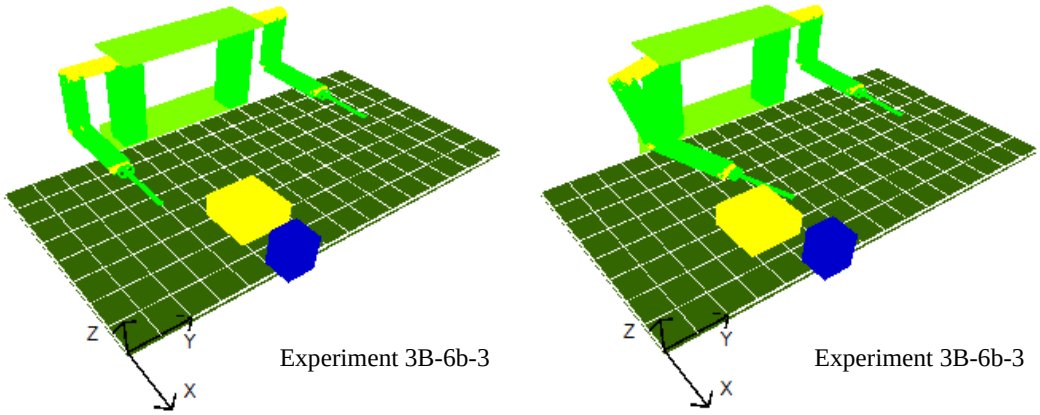


Figure 238 Simulation Results of Experiment 3B-6b-3

ISAC pushes the box to the right using its right arm in Experiment 3B-6b-4.

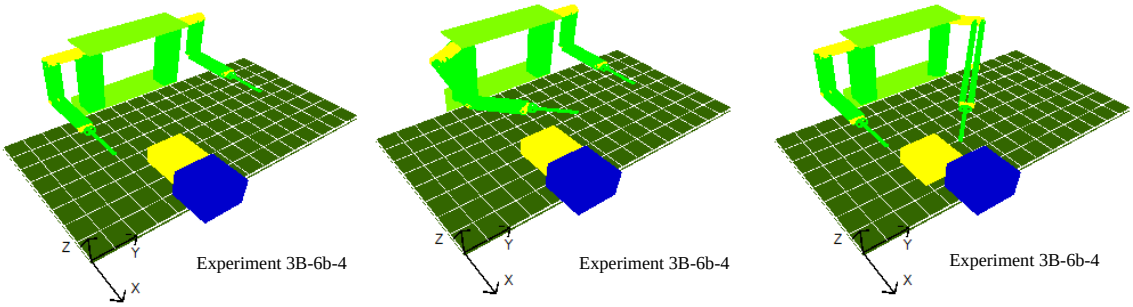


Figure 239 Simulation Results of Experiment 3B-6b-4

ISAC pushes the box to the right using its right arm in Experiment 3B-6c-1.

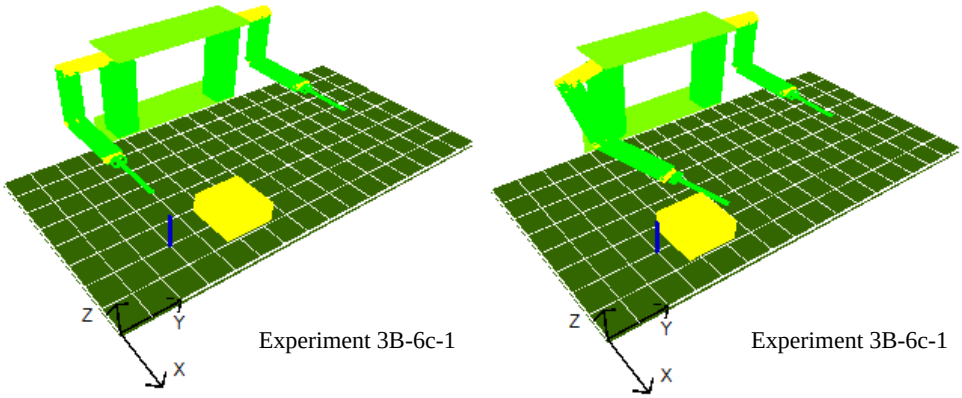


Figure 240 Simulation Results of Experiment 3B-6c-1

ISAC pushes the box to the right using its right arm in Experiment 3B-6c-2.

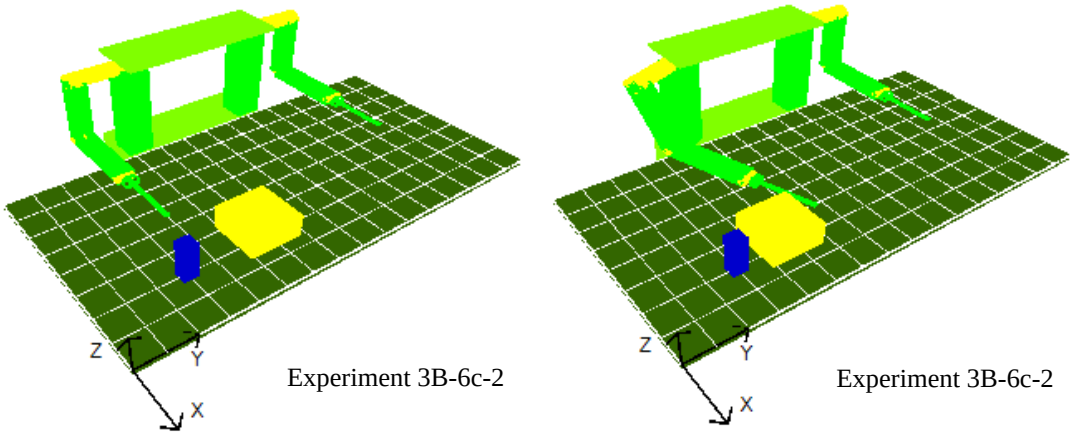


Figure 241 Simulation Results of Experiment 3B-6c-2

ISAC pushes the box to the right using its right arm in Experiment 3B-6c-3.

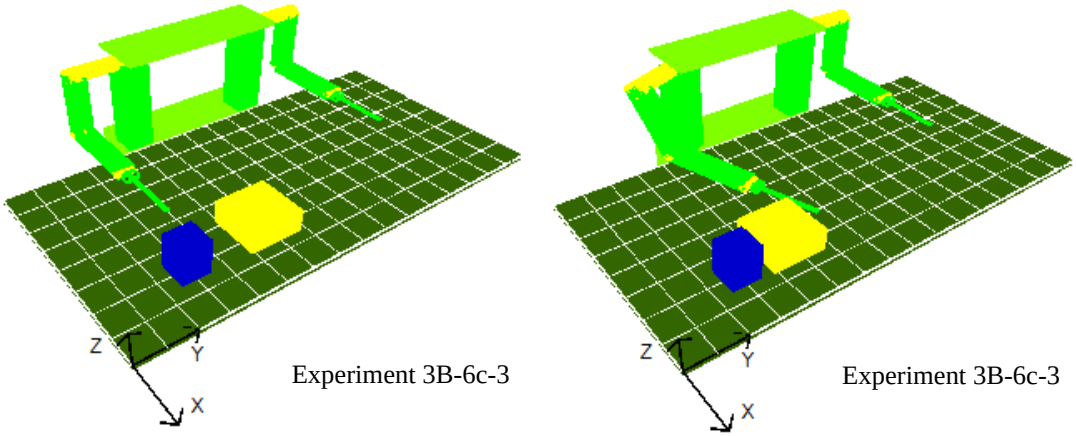


Figure 242 Simulation Results of Experiment 3B-6c-3

ISAC pushes the box to the right using its right arm in Experiment 3B-6c-4.

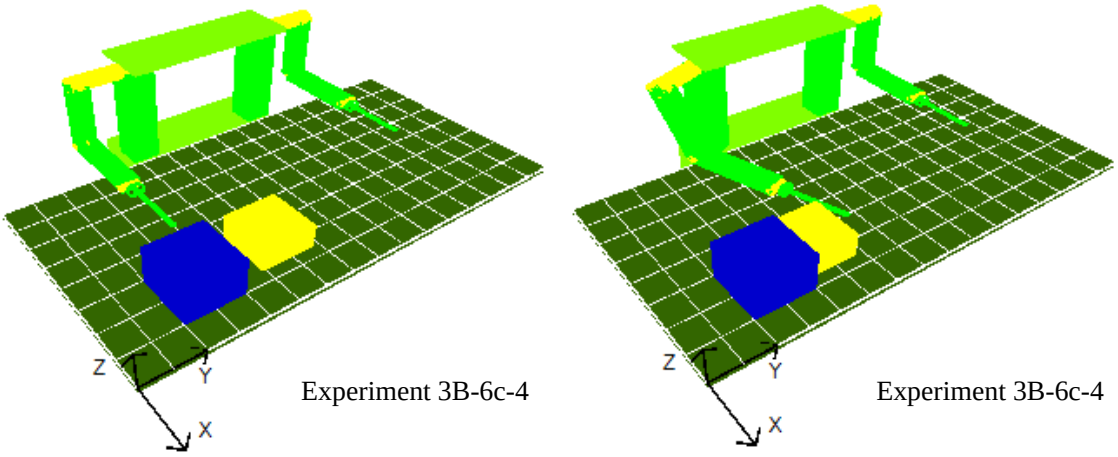


Figure 243 Simulation Results of Experiment 3B-6c-4

ISAC pushes the box to the right using its right arm in Experiment 3B-6d-1.

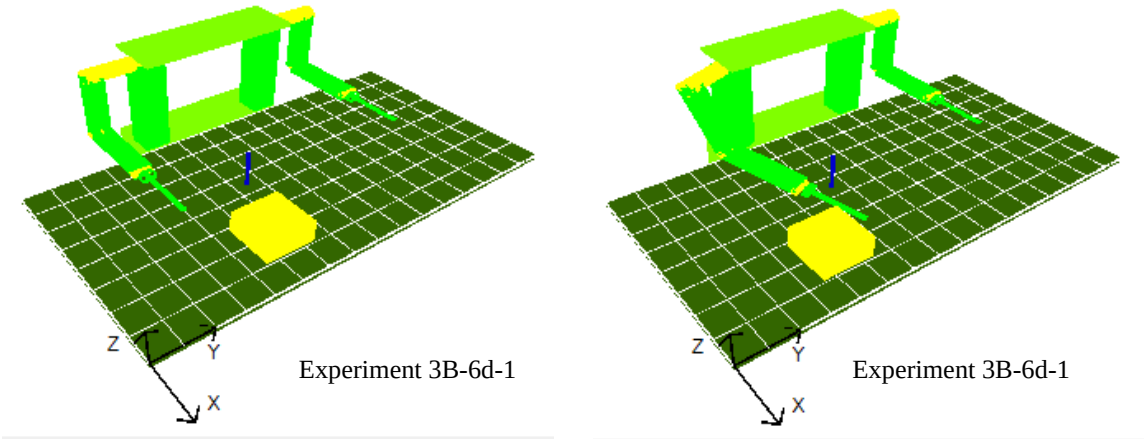


Figure 244 Simulation Results of Experiment 3B-6d-1

ISAC pushes the box to the right using its right arm in Experiment 3B-6d-2.

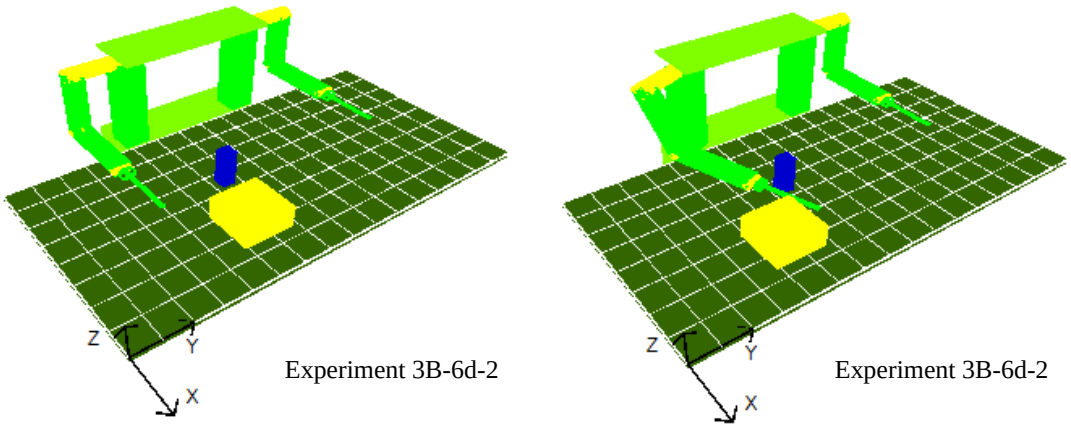


Figure 245 Simulation Results of Experiment 3B-6d-2

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-6d-3. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

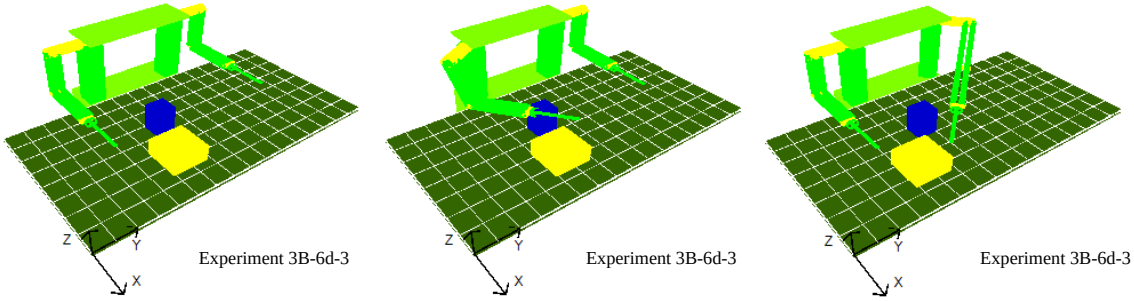


Figure 246 Simulation Results of Experiment 3B-6d-3

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-6d-4. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

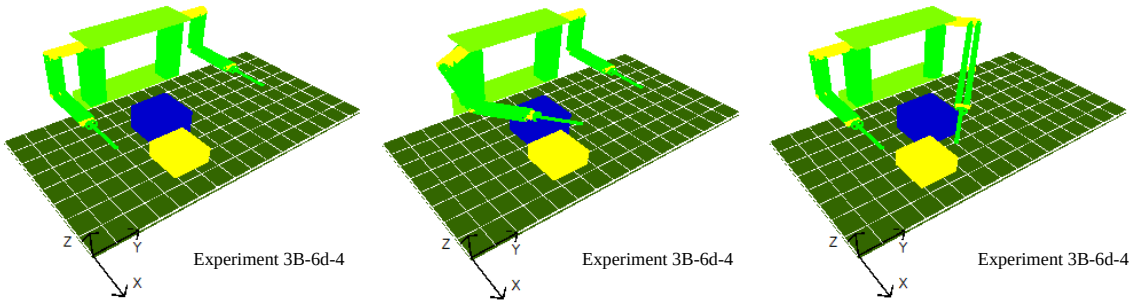


Figure 247 Simulation Results of Experiment 3B-6d-4

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-7a-1. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

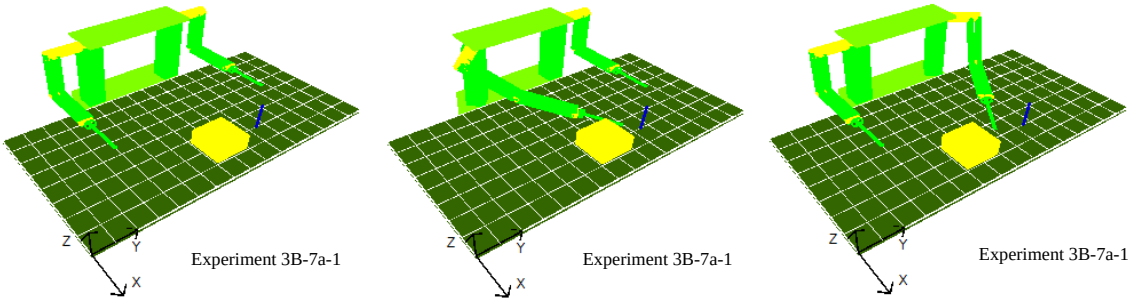


Figure 248 Simulation Results of Experiment 3B-7a-1

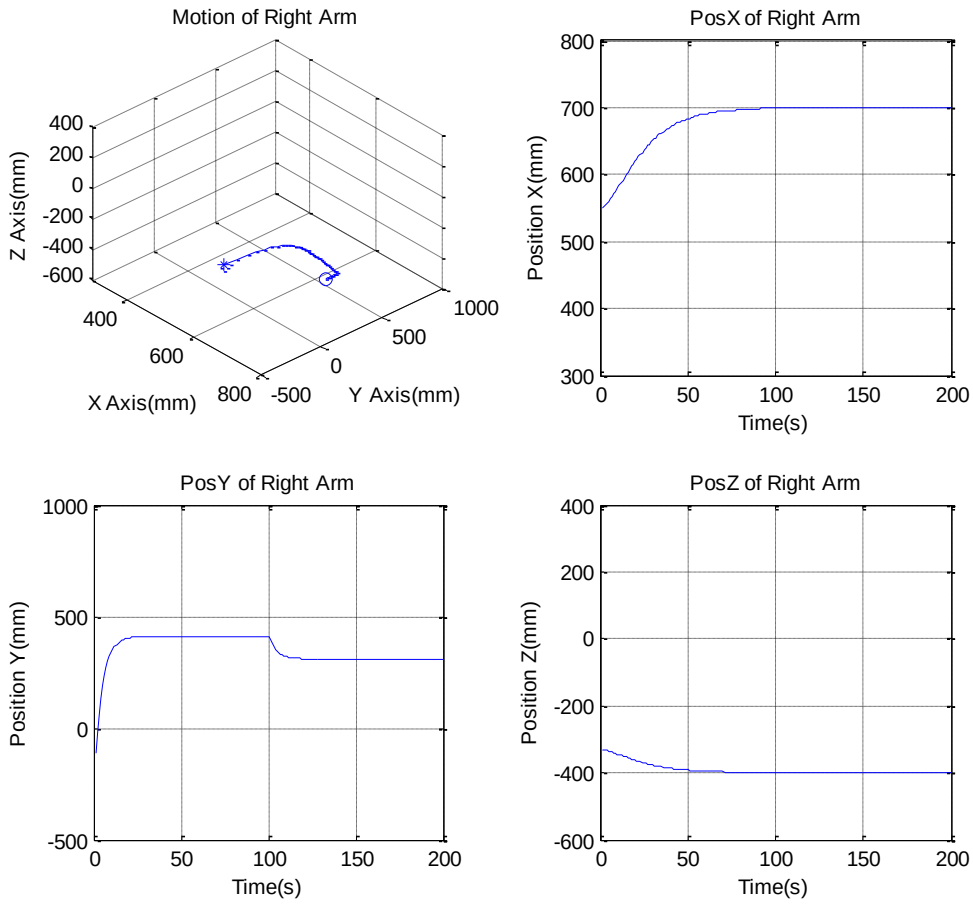


Figure 249 Generated Motion Trajectories of the Right Arm in Experiment 3B-7a-1

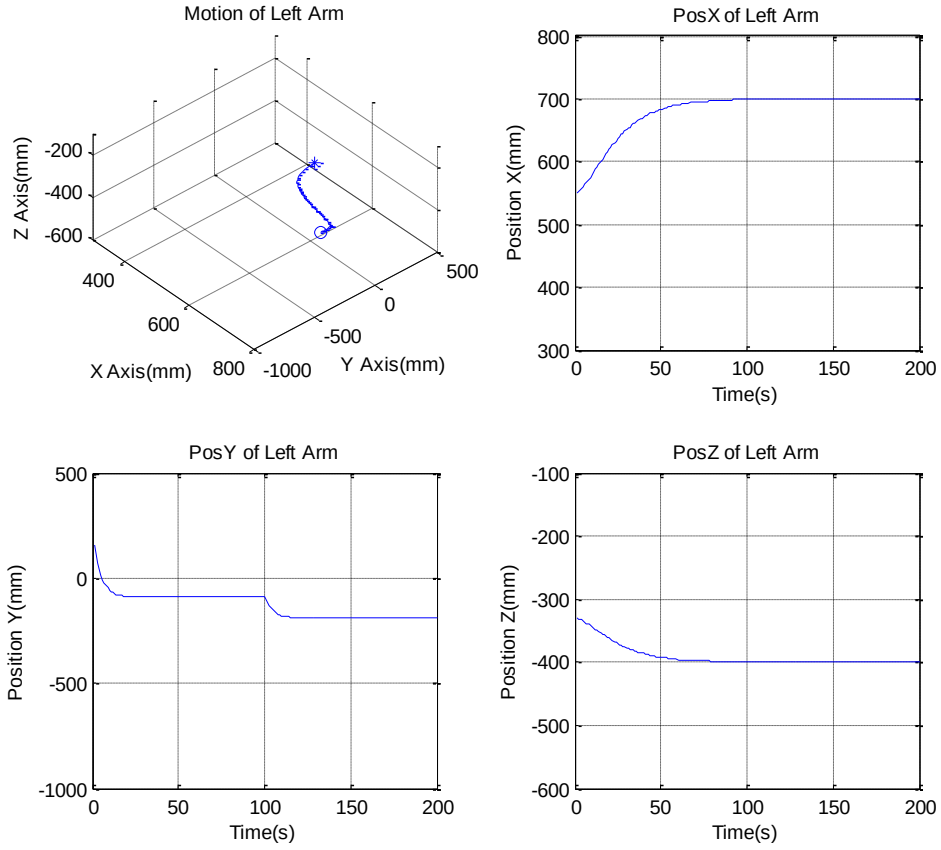


Figure 250 Generated Motion Trajectories of the Left Arm in Experiment 3B-7a-1

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-7a-2. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

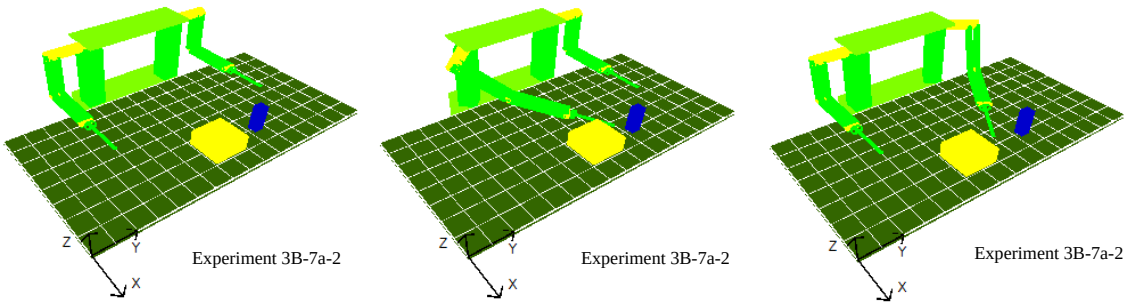


Figure 251 Simulation Results of Experiment 3B-7a-2

ISAC finds that it cannot push the box to the right using either of the arms in Experiment 3B-7a-3.

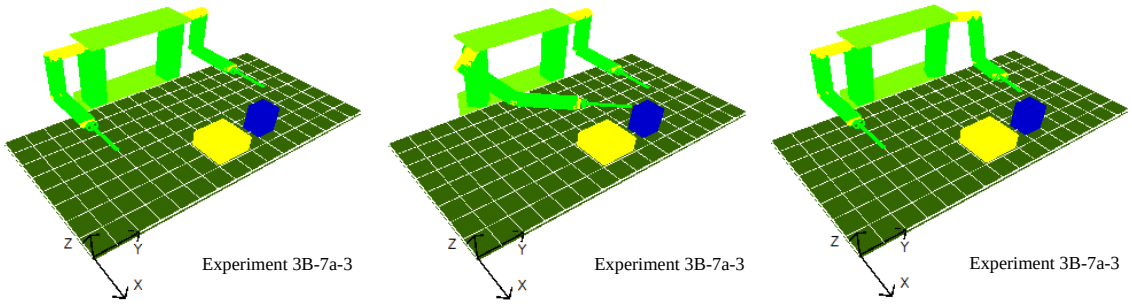


Figure 252 Simulation Results of Experiment 3B-7a-3

ISAC finds that it cannot push the box to the right using either of the arms in Experiment 3B-7a-4.

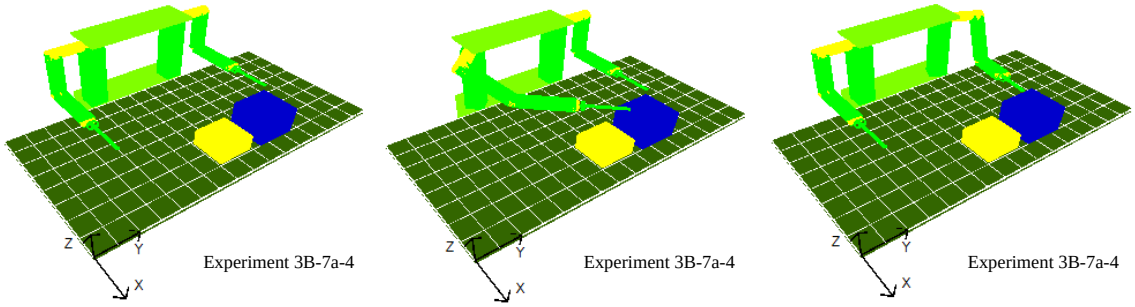


Figure 253 Simulation Results of Experiment 3B-7a-4

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-7b-1. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

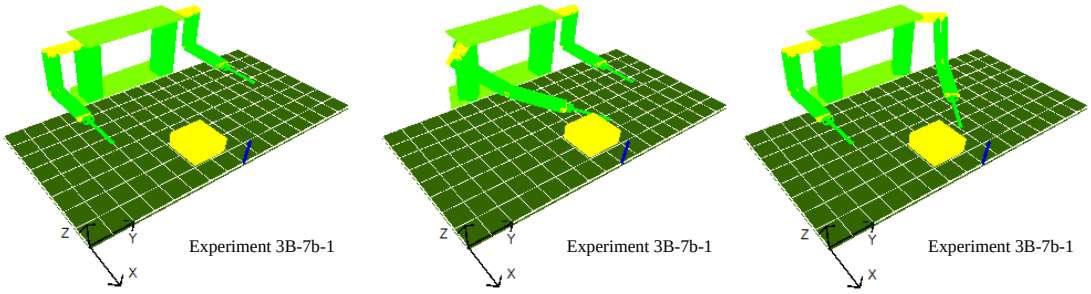


Figure 254 Simulation Results of Experiment 3B-7b-1

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-7b-2. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

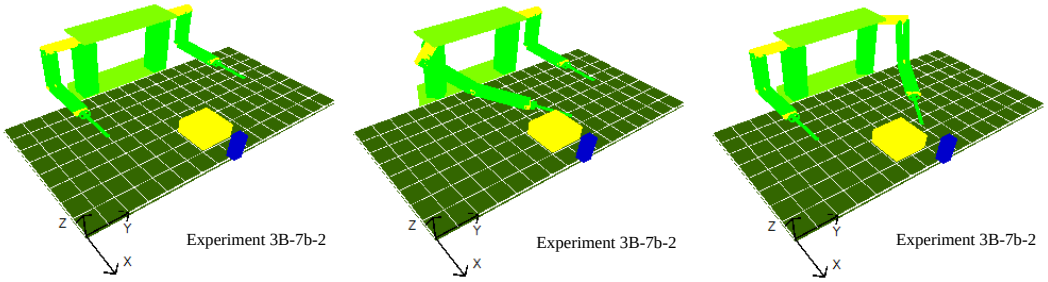


Figure 255 Simulation Results of Experiment 3B-7b-2

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-7b-3. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

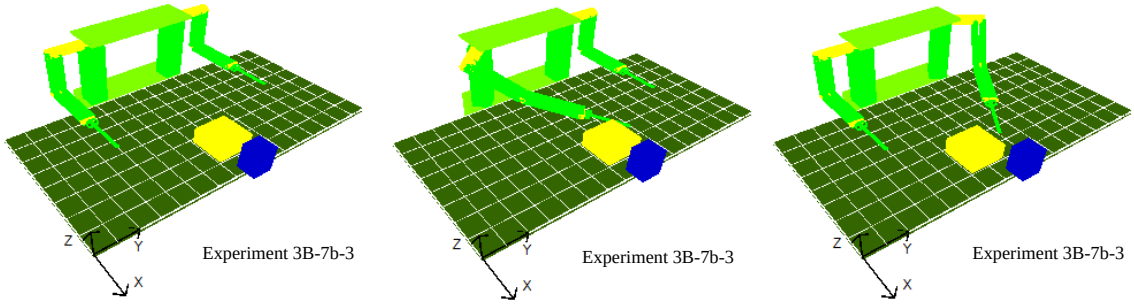


Figure 256 Simulation Results of Experiment 3B-7b-3

ISAC finds that it cannot push the box to the right using either of arms in Experiment 3B-7b-4.

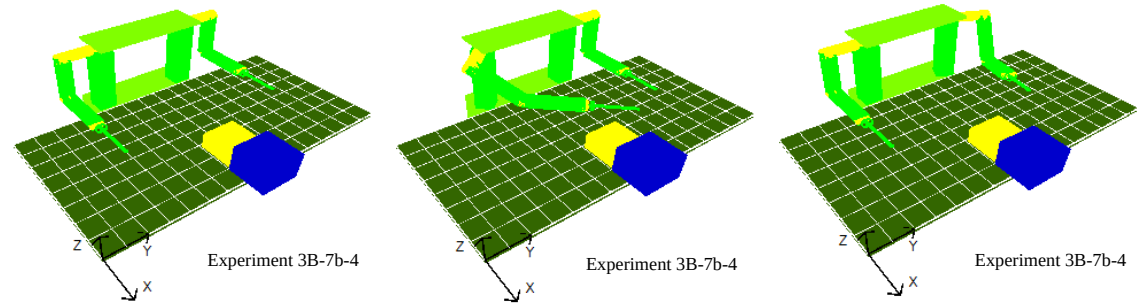


Figure 257 Simulation Results of Experiment 3B-7b-4

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-7c-1. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

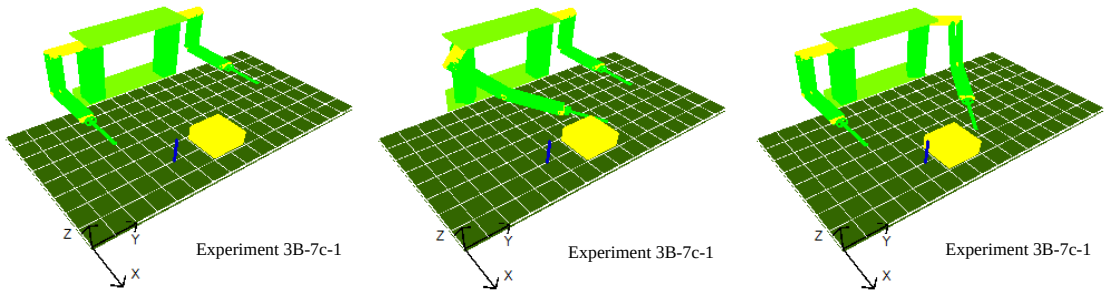


Figure 258 Simulation Results of Experiment 3B-7c-1

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-7c-2. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

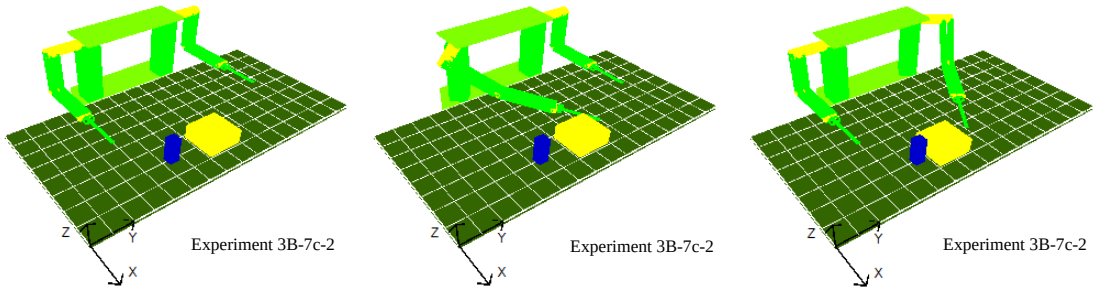


Figure 259 Simulation Results of Experiment 3B-7c-2

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-7c-3. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

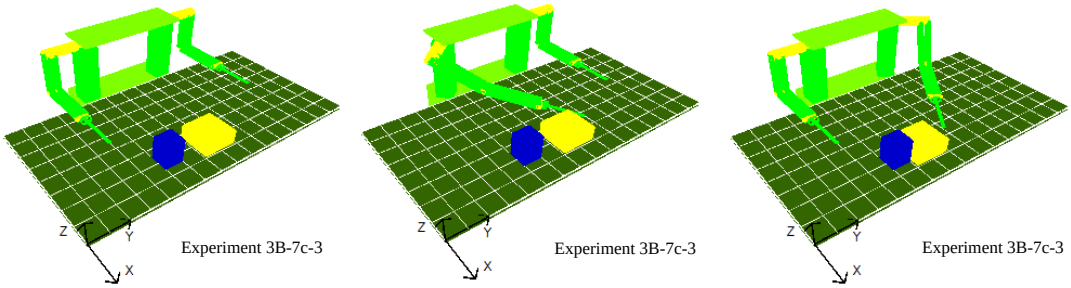


Figure 260 Simulation Results of Experiment 3B-7c-3

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-7c-4. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

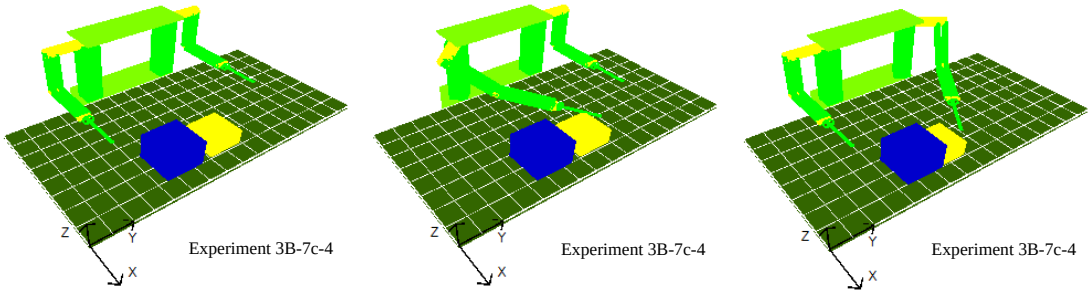


Figure 261 Simulation Results of Experiment 3B-7c-4

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-7d-1. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

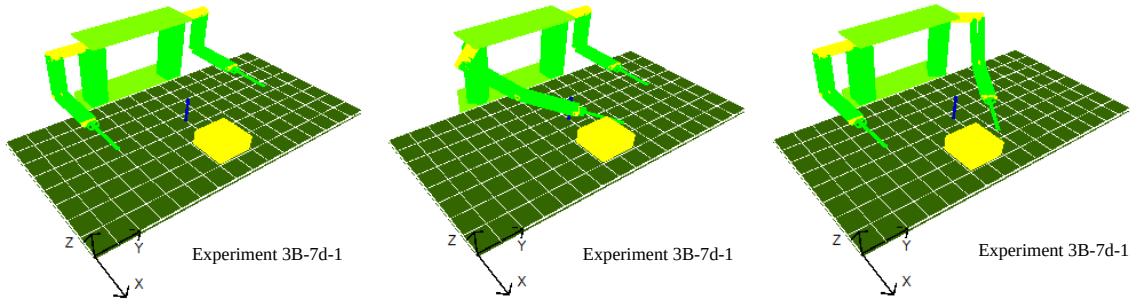


Figure 262 Simulation Results of Experiment 3B-7d-1

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-7d-2. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

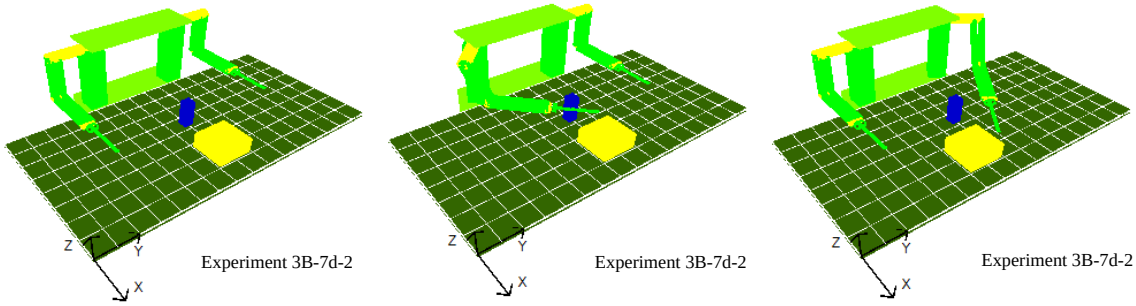


Figure 263 Simulation Results of Experiment 3B-7d-2

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-7d-3. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

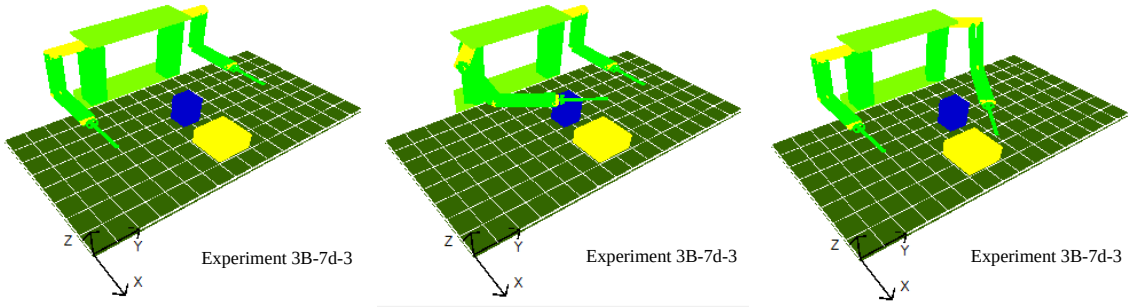


Figure 264 Simulation Results of Experiment 3B-7d-3

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-7d-4. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

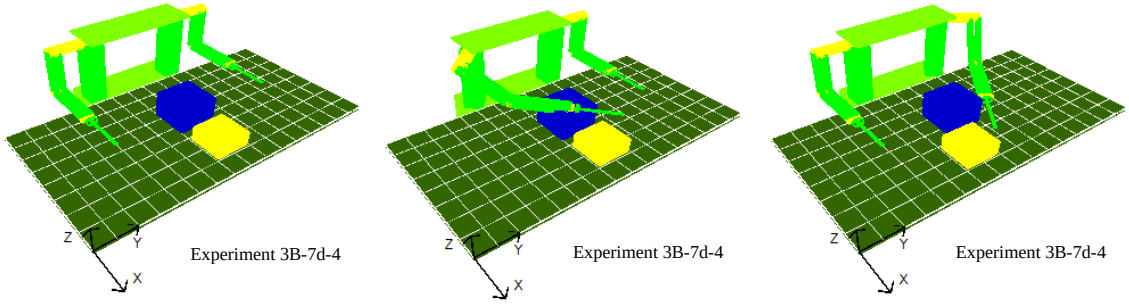


Figure 265 Simulation Results of Experiment 3B-7d-4

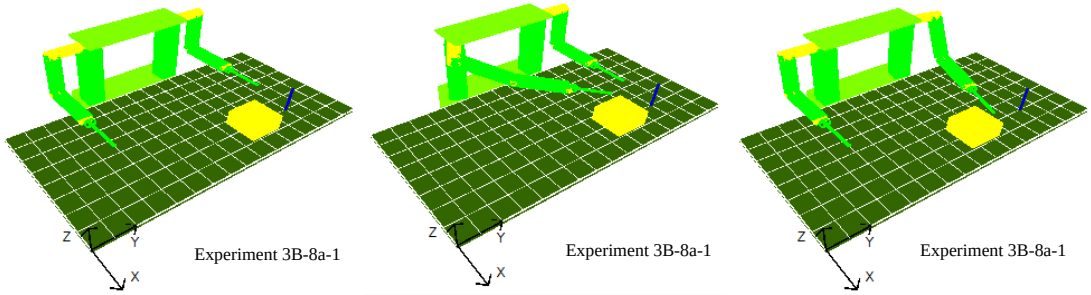


Figure 266 Simulation Results of Experiment 3B-8a-1

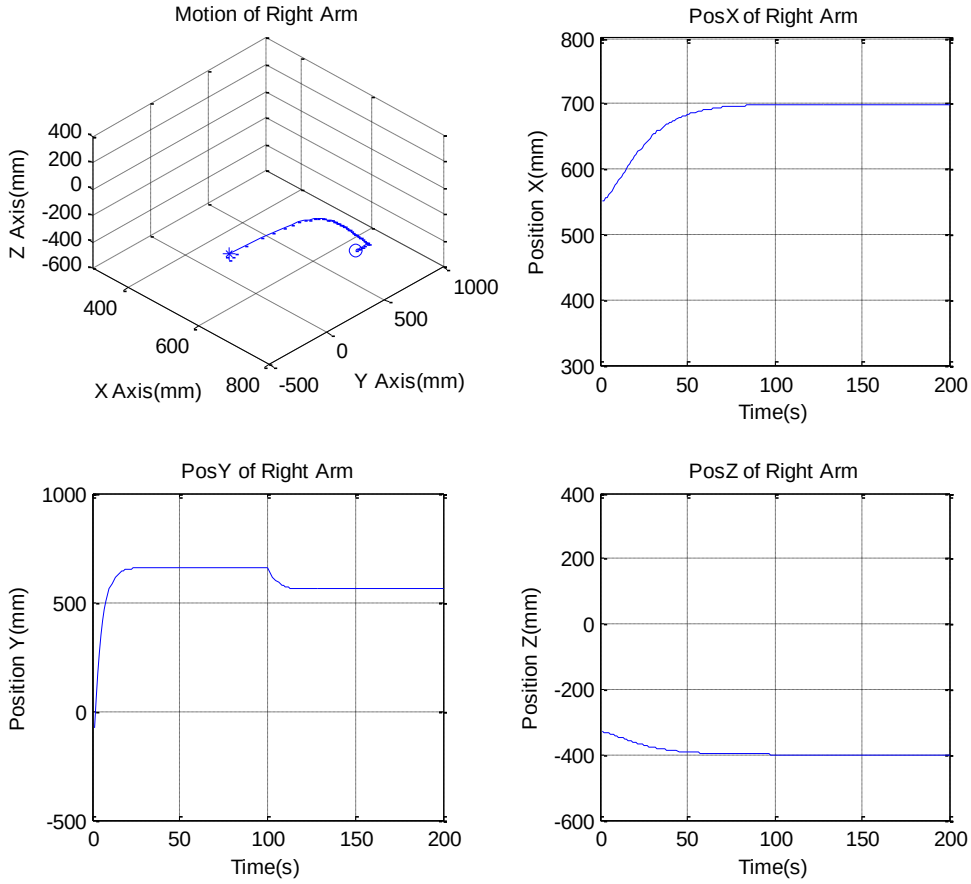


Figure 267 Generated Motion Trajectories of the Right Arm in Experiment 3B-8a-1

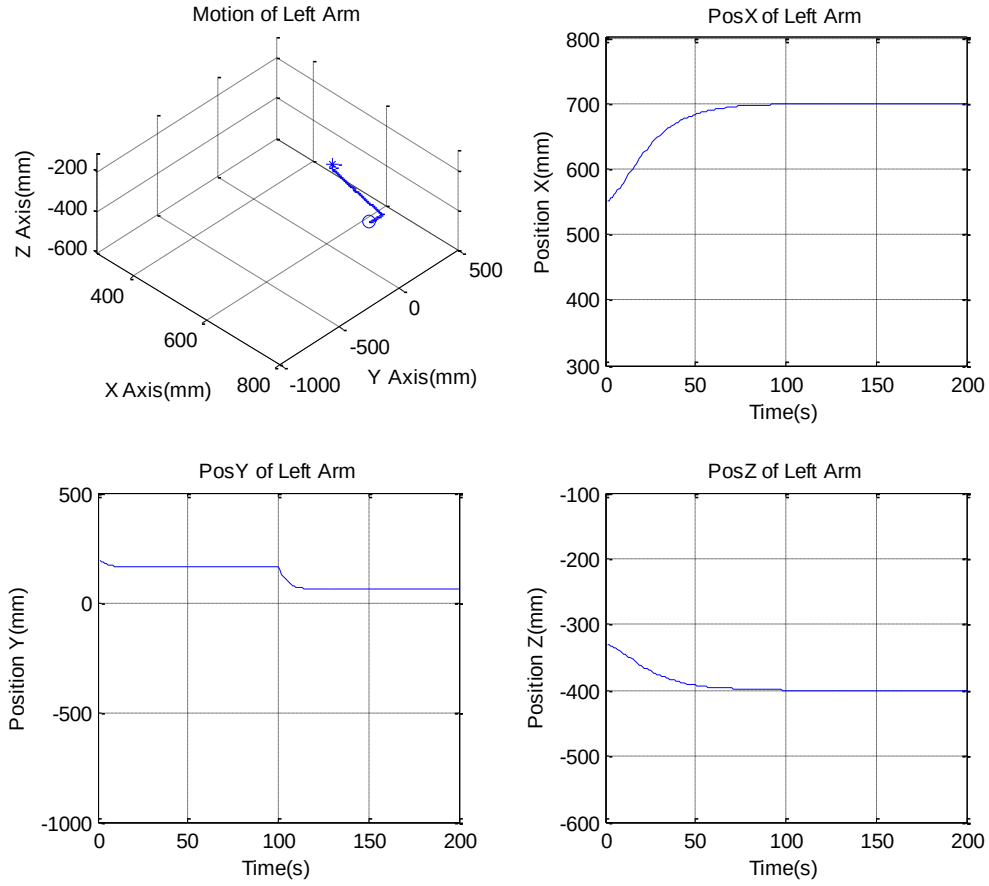


Figure 268 Generated Motion Trajectories of the Left Arm in Experiment 3B-8a-1

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-8a-2. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

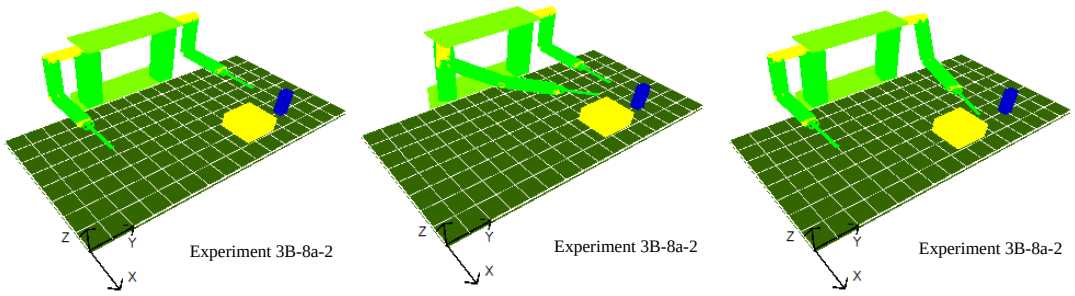


Figure 269 Simulation Results of Experiment 3B-8a-2

ISAC finds that it cannot push the box to the right using either of the arms in Experiment 3B-8a-3.

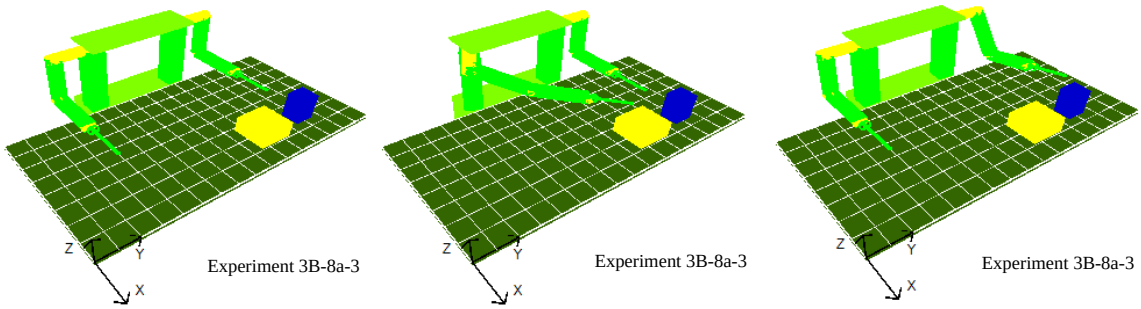


Figure 270 Simulation Results of Experiment 3B-8a-3

ISAC finds that it cannot push the box to the right using either of the arms in Experiment 3B-8a-4.

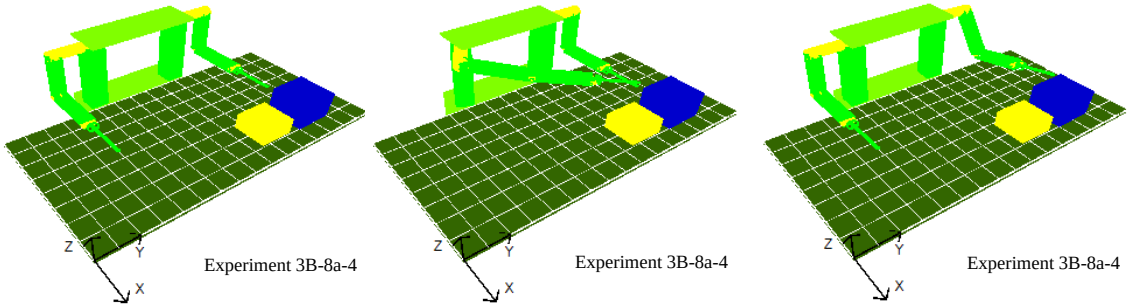


Figure 271 Simulation Results of Experiment 3B-8a-4

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-8b-1. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

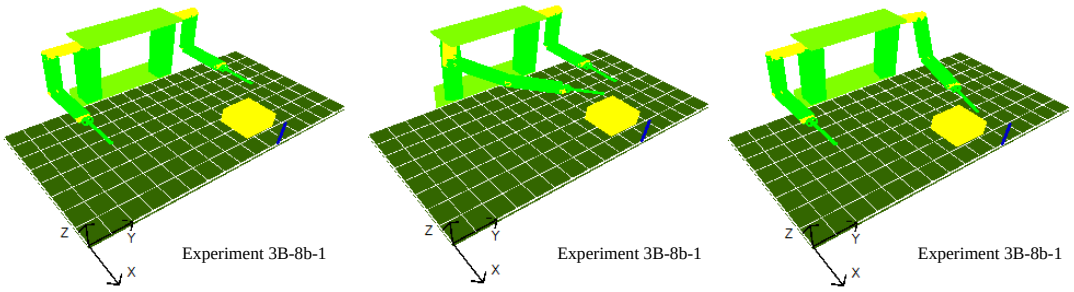


Figure 272 Simulation Results of Experiment 3B-8b-1

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-8b-2. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

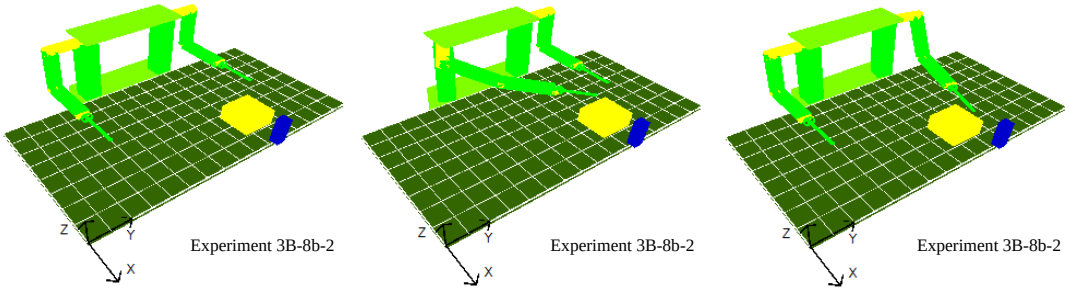


Figure 273 Simulation Results of Experiment 3B-8b-2

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-8b-3. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

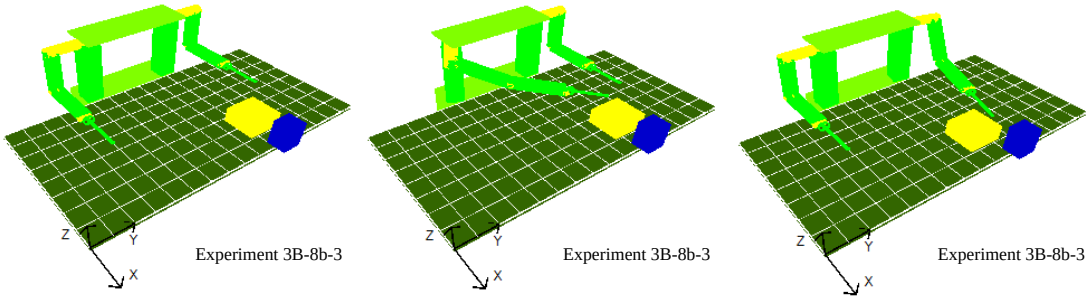


Figure 274 Simulation Results of Experiment 3B-8b-3

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-8b-4. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

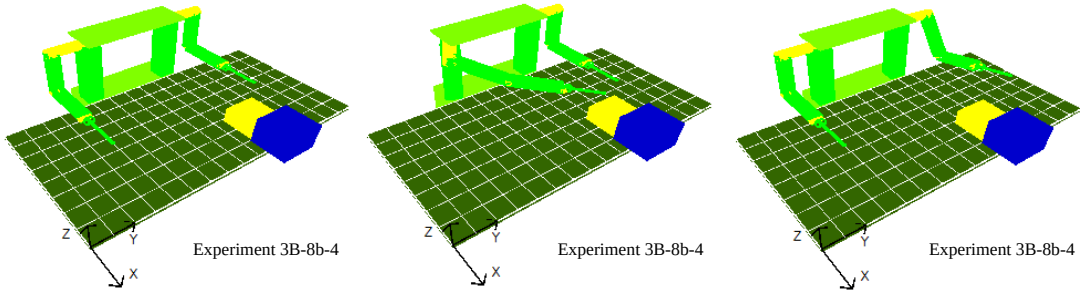


Figure 275 Simulation Results of Experiment 3B-8b-4

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-8c-1. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

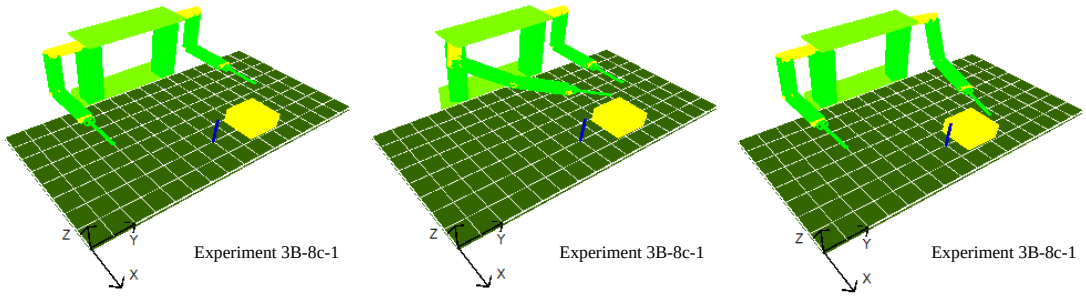


Figure 276 Simulation Results of Experiment 3B-8c-1

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-8c-2. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

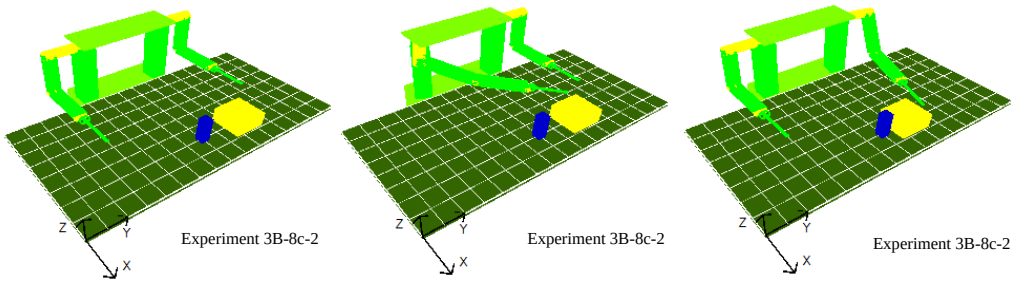


Figure 277 Simulation Results of Experiment 3B-8c-2

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-8c-3. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

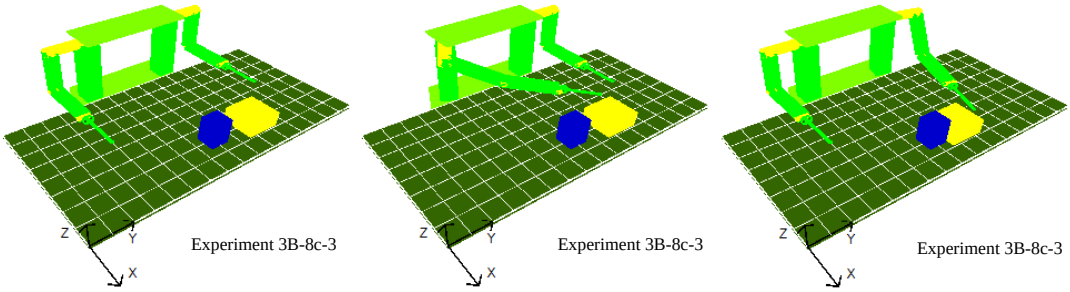


Figure 278 Simulation Results of Experiment 3B-8c-3

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-8c-3. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

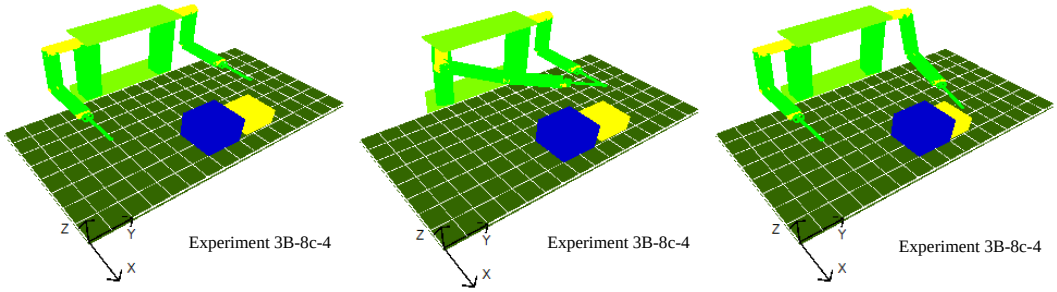


Figure 279 Simulation Results of Experiment 3B-8c-4

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-8d-1. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

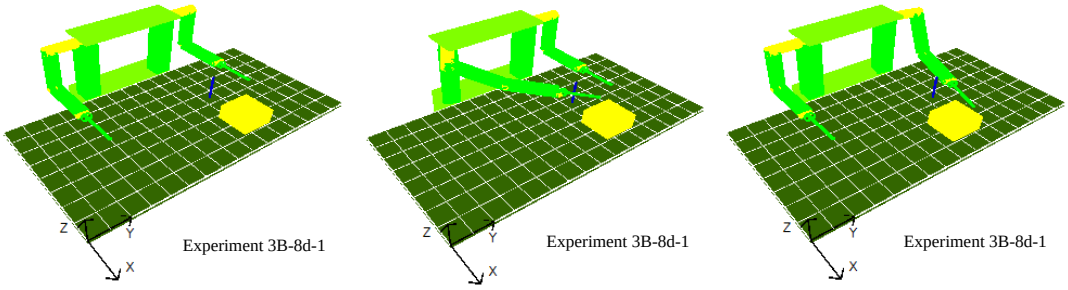


Figure 280 Simulation Results of Experiment 3B-8d-1

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-8d-2. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

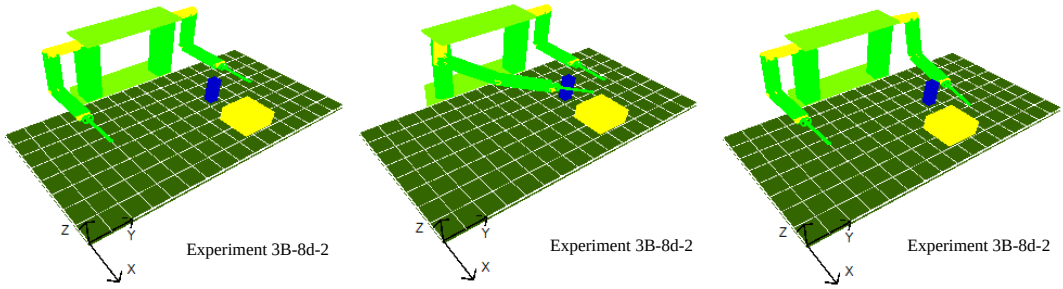


Figure 281 Simulation Results of Experiment 3B-8d-2

ISAC finds that it cannot push the box to the right using its right arms in Experiment 3B-8d-3. Then ISAC switches the generated behavior sequence to the left arm and pushes the box to the right.

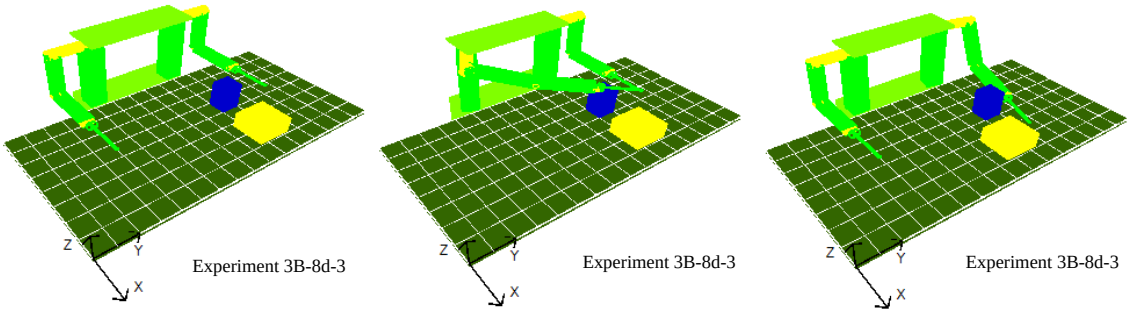


Figure 282 Simulation Results of Experiment 3B-8d-3

ISAC finds that it cannot push the box to the right using either of the arms in Experiment 3B-8d-4.

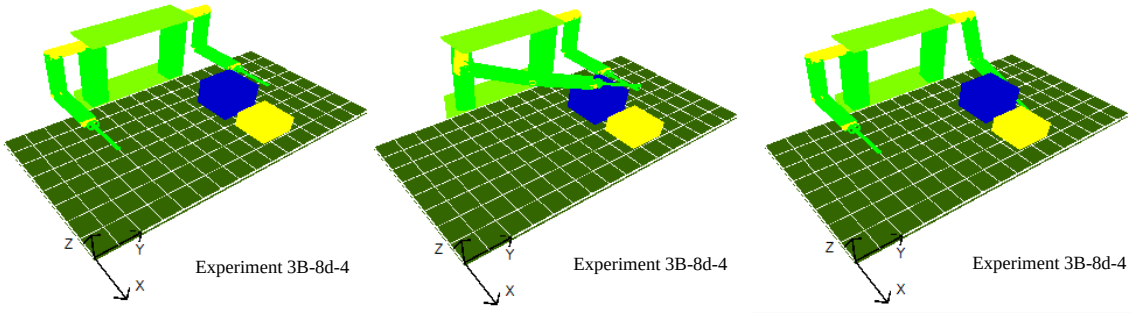


Figure 283 Simulation Results of Experiment 3B-8d-4

--*Experiment 3B-1*

In experiment 3B-1, the object was placed at location 1 and the obstacle was placed around it at 4 different locations. ISAC first tried to push the object using its right arm. If the evaluation failed, ISAC switched to the left arm. If the evaluation still failed, ISAC refused to complete the task and displayed a message on the screen.

The simulation results of 3B-1 are summarized in Table 28.

Table 28 Simulation Results of Experiment 3B-1

	Feasible/ Infeasible	Left/Right	Failure Reason	Evaluation
Experiment 3B-1a-1	Feasible	Right Arm	N/A	Correct
Experiment 3B-1a-2	Feasible	Right Arm	N/A	Correct
Experiment 3B-1a-3	Infeasible	N/A	Right Arm: Cannot Reach the Box Left Arm: Cannot Reach the Box	Wrong
Experiment 3B-1a-4	Infeasible	N/A	Right Arm: Cannot Reach the Box Left Arm: Cannot Reach the Box	Correct
Experiment 3B-1b-1	Feasible	Right Arm	N/A	Correct
Experiment 3B-1b-2	Feasible	Right Arm	N/A	Correct
Experiment 3B-1b-3	Feasible	Right Arm	N/A	Correct
Experiment 3B-1b-4	Infeasible	N/A	Right Arm: Cannot Reach the Box Left Arm: Cannot Reach the Box	Wrong
Experiment 3B-1c-1	Feasible	Right Arm	N/A	Correct
Experiment 3B-1c-2	Feasible	Right Arm	N/A	Correct
Experiment 3B-1c-3	Feasible	Right Arm	N/A	Correct
Experiment 3B-1c-4	Feasible	Right Arm	N/A	Correct
Experiment 3B-1d-1	Feasible	Right Arm	N/A	Correct
Experiment 3B-1d-2	Feasible	Right Arm	N/A	Correct
Experiment 3B-1d-3	Feasible	Right Arm	N/A	Correct
Experiment 3B-1d-4	Infeasible	N/A	Right Arm: Collision with the obstacle Left Arm: Cannot Reach the Box	Correct

In Experiment 3B-1a-3, 3B-1a-4, 3B-1b-4, and 3B-1d-4, ISAC found that it cannot push the box using either of its arms if it wanted to avoid the obstacle. So it displayed a message on the screen and refused to do. The rate of feasibility in Experiment 3B-1 is 75%.

In Experiment 3B-1a-3 and 3B-1b-4, ISAC made wrong decisions and refused to complete the task because it found that it cannot avoid the obstacle in order to push the box. The reason of causing the wrong decision comes from the obstacle avoidance module. A potential field-based method is used for ISAC to avoid the obstacle. As the size of the obstacle increases, the impedance potential field of the obstacle also increases. So ISAC cannot reach the box to push it because the larger impedance potential field of the obstacle. The success rate of the evaluation using the IRS in Experiment 3B-1 is 87.5%.

Table 29 displays the running time of the key components in Experiment 3B-1.

As shown in Table 29, the average running time for generating the behavior sequence in Experiment 3B-1 is 0.0050ms. The average running time for generating behaviors for the right arm and the left arm is 11.3215ms and 10.9597ms respectively.

The average running time of evaluating the generated behaviors for the right arm and for the left arm is 0.7758ms and 0.5004ms respectively. In Experiment 3B-1a-1, 3B-1a-2, 3B-1b-1, 3B-1b-2, 3B-1b-3, 3B-1c-1, 3B-1c-2, 3B-1c-3, 3B-1c-4, 3B-1d-1, 3B-1d-2, and 3B-1d-3, ISAC only needed to evaluate the behaviors for the right arm and did not evaluate the behaviors for the left arm, so the running time for evaluating the left arm is very small: 0.0004ms. That means ISAC evaluated the overall motion trajectory of the right arm The average running time for evaluating the behaviors for the right arm in these experiments is: 0.8227ms.

The running time of generating behavior sequence is 0.02% of the overall running time of the key components. 48.18% and 46.64% of the overall running time is used to

generate motion trajectories for the right arm and the left arm respectively. The evaluation time is 3.30% and 1.85% of the overall running time.

Table 29 Running Time of Key Components in Experiment 3B-1

	BSG	BGRA	BGLA	IRSRA	IRSLA
Experiment 3B-1a-1	0.0053	10.2442	10.7762	0.7960	0.0004
Experiment 3B-1a-2	0.0049	10.6272	10.6334	0.7976	0.0000
Experiment 3B-1a-3	0.0049	10.8399	10.4080	0.8653	0.1206
Experiment 3B-1a-4	0.0053	10.6009	11.3017	0.0114	1.1170
Experiment 3B-1b-1	0.0049	10.9897	10.6978	0.8276	0.0004
Experiment 3B-1b-2	0.0049	10.7249	11.1289	0.8296	0.0000
Experiment 3B-1b-3	0.0041	10.5398	10.5373	0.8362	0.0004
Experiment 3B-1b-4	0.0049	10.8218	10.7861	0.8300	0.2635
Experiment 3B-1c-1	0.0049	10.9179	10.4059	0.8288	0.0004
Experiment 3B-1c-2	0.0053	10.7512	11.1691	0.8296	0.0000
Experiment 3B-1c-3	0.0057	18.3274	11.0968	0.8337	0.0000
Experiment 3B-1c-4	0.0049	10.9072	10.8969	0.8313	0.0004
Experiment 3B-1d-1	0.0049	11.3284	11.5369	0.8292	0.0004
Experiment 3B-1d-2	0.0049	10.8949	11.5558	0.8341	0.0004
Experiment 3B-1d-3	0.0049	11.2077	11.1453	0.7992	0.0004
Experiment 3B-1d-4	0.0049	11.4216	11.2787	0.8325	0.0004
Average	0.0050	11.3215	10.9597	0.7758	0.5004
STD	0.0003	1.8915	0.3711	0.2045	0.5388
Percentage	0.02%	48.18%	46.64%	3.30%	1.85%

Unit: millisecond (ms)

--*Experiment 3B-2*

The simulation results of 3B-2 are summarized in Table 30.

Table 30 Simulation Results of Experiment 3B-2

	Feasible/ Infeasible	Left/Right	Failure Reason	Evaluation
Experiment 3B-2a-1	Infeasible	N/A	Right Arm: Cannot Reach the Box Left Arm: Collision with the obstacle	Wrong
Experiment 3B-2a-2	Infeasible	N/A	Right Arm: Cannot Reach the Box Left Arm: Collision with the obstacle	Wrong
Experiment 3B-2a-3	Infeasible	N/A	Right Arm: Cannot Reach the Box Left Arm: Collision with the obstacle	Correct
Experiment 3B-2a-4	Infeasible	N/A	Right Arm: Cannot Reach the Box Left Arm: Cannot Reach the Box	Correct
Experiment 3B-2b-1	Feasible	Left Arm	Right Arm: Cannot Reach the Box	Correct
Experiment 3B-2b-2	Feasible	Left Arm	Right Arm: Cannot Reach the Box	Correct
Experiment 3B-2b-3	Feasible	Left Arm	Right Arm: Cannot Reach the Box	Correct
Experiment 3B-2b-4	Feasible	Left Arm	Right Arm: Cannot Reach the Box	Correct
Experiment 3B-2c-1	Feasible	Left Arm	Right Arm: Cannot Reach the Box	Correct
Experiment 3B-2c-2	Feasible	Left Arm	Right Arm: Cannot Reach the Box	Correct
Experiment 3B-2c-3	Feasible	Left Arm	Right Arm: Cannot Reach the Box	Correct
Experiment 3B-2c-4	Feasible	Left Arm	Right Arm: Cannot Reach the Box	Correct
Experiment 3B-2d-1	Feasible	Left Arm	Right Arm: Cannot Reach the Box	Correct
Experiment 3B-2d-2	Feasible	Left Arm	Right Arm: Cannot Reach the Box	Correct
Experiment 3B-2d-3	Feasible	Left Arm	Right Arm: Cannot Reach the Box	Correct
Experiment 3B-2d-4	Feasible	Left Arm	Right Arm: Cannot Reach the Box	Correct

In Experiment 3B-2a-1, 3B-2a-2, 3B-2b-3, and 3B-2d-4, ISAC found that it cannot push the box using either of its arms if it wanted to avoid the obstacle. So it

displayed a message on the screen and refused to complete the task. The rate of feasibility in Experiment 3B-2 is 75%.

In Experiment 3B-2a-1 and 3B-2a-2, ISAC made wrong decisions, and refused to complete the task because it finds that it cannot avoid the obstacle in order to push the box. The reason of causing the wrong decision still comes from the obstacle avoidance module. A potential field-based method is used for ISAC to avoid the obstacle. However, this method usually cannot find a global solution. ISAC can avoid the obstacle in these methods by moving in the front or the back of the obstacle. However, in these experiments, ISAC moved in the front of the obstacle, so the arm collided with the obstacle. The success rate of the evaluation using the IRS in Experiment 3B-2 is 87.5%. Table 31 displays the running time of key components in Experiment 3B-2.

As shown in Table 31, the average running time for generating the behavior sequence in Experiment 3B-2 is 0.0050ms. The average running time for generating behaviors for the right arm and the left arm is 10.7149ms and 13.6738ms respectively.

The average running time of evaluating the generated behaviors for the right arm and for the left arm is 0.1618ms and 0.9363ms respectively. In the Experiments except 3B-2a-1, 3B-2a-2, 3B-2a-3, 3B-2a-4, ISAC needed to evaluate the overall motion trajectory of the left arm. The average time for evaluating the behaviors for the left arm in these experiments is: 1.1278ms.

The running time of generating behavior sequence is 0.02% of the overall running time of the key components. 42.03% and 53.64% of the overall running time is used to generate motion trajectories for the right arm and the left arm respectively. The evaluation time is 0.63% and 3.67% of the overall running time.

Table 31 Running Time of Key Components in Experiment 3B-2

	BSG	BGRA	BGLA	IRSRA	IRSLA
Experiment 3B-2a-1	0.0049	10.5295	10.5554	0.0784	0.0977
Experiment 3B-2a-2	0.0049	10.6506	10.7955	0.0784	0.0862
Experiment 3B-2a-3	0.0049	10.6358	20.5574	0.0874	0.0825
Experiment 3B-2a-4	0.0049	10.2434	10.2967	0.8805	1.1802
Experiment 3B-2b-1	0.0053	10.4527	10.1432	0.0849	1.1523
Experiment 3B-2b-2	0.0049	10.3411	49.4891	0.3830	1.0969
Experiment 3B-2b-3	0.0049	11.2574	10.8534	0.4027	1.1002
Experiment 3B-2b-4	0.0049	10.8813	10.8760	0.0233	1.1987
Experiment 3B-2c-1	0.0053	10.5414	10.8230	0.0796	1.0997
Experiment 3B-2c-2	0.0049	10.6769	10.6756	0.0788	1.0997
Experiment 3B-2c-3	0.0049	10.3702	10.5981	0.0821	1.1034
Experiment 3B-2c-4	0.0049	11.0722	10.5406	0.0127	1.1650
Experiment 3B-2d-1	0.0049	11.3345	10.4355	0.0853	1.1248
Experiment 3B-2d-2	0.0049	10.4745	10.6157	0.0841	1.1260
Experiment 3B-2d-3	0.0053	10.7413	10.8390	0.0903	1.1125
Experiment 3B-2d-4	0.0053	11.2352	10.6863	0.0570	1.1548
Average	0.0050	10.7149	13.6738	0.1618	0.9363
STD	0.0002	0.3453	9.8691	0.2218	0.4216
Percentage	0.02%	42.03%	53.64%	0.63%	3.67%

--*Experiment 3B-3*

In Experiment 3B-3a-1, 3B-3a-2, 3B-3b-3, 3B-3d-4, and 3B-3b-4, ISAC found that it cannot push the box using either of its arms if it wanted to avoid the obstacle. So it

displayed a message on the screen and refused to complete the task. The rate of feasibility in Experiment 3B-2 is 68.75%.

Table 32 Simulation Results of Experiment 3B-3

	Feasible/ Infeasible	Left/Right	Failure Reason	Evaluation
Experiment 3B-3a-1	Infeasible	N/A	Right Arm: Collision with the obstacle Left Arm: Collision with the obstacle	Wrong
Experiment 3B-3a-2	Infeasible	N/A	Right Arm: Collision with the obstacle Left Arm: Collision with the obstacle	Wrong
Experiment 3B-3a-3	Infeasible	N/A	Right Arm: Collision with the obstacle Left Arm: Collision with the obstacle	Correct
Experiment 3B-3a-4	Infeasible	N/A	Right Arm: Collision with the obstacle Left Arm: Collision with the obstacle	Correct
Experiment 3B-3b-1	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-3b-2	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-3b-3	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-3b-4	Infeasible	N/A	Right Arm: Collision with the object Left Arm: Cannot reach the box	Correct
Experiment 3B-3c-1	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-3c-2	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-3c-3	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-3c-4	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-3d-1	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-3d-2	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-3d-3	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-3d-4	Feasible	Left Arm	Right Arm: Collision with the object	Correct

Table 33 Running Time of Key Components in Experiment 3B-3

	BSG	BGRA	BGLA	IRSRA	IRSLA
Experiment 3B-3a-1	0.0053	10.9540	11.3805	0.0849	1.1613
Experiment 3B-3a-2	0.0049	21.0812	10.7795	0.0841	0.0242
Experiment 3B-3a-3	0.0057	11.2931	10.9162	0.0853	0.0127
Experiment 3B-3a-4	0.0057	10.9109	10.5114	0.8855	0.0119
Experiment 3B-3b-1	0.0061	11.2274	10.9675	0.0845	1.1572
Experiment 3B-3b-2	0.0049	11.0221	11.0028	0.3801	1.1556
Experiment 3B-3b-3	0.0053	11.2077	10.9581	0.3891	1.1593
Experiment 3B-3b-4	0.0053	10.9523	11.4322	0.8830	1.1761
Experiment 3B-3c-1	0.0053	11.3666	11.0972	0.0837	1.1568
Experiment 3B-3c-2	0.0049	11.0049	11.2134	0.0311	1.1593
Experiment 3B-3c-3	0.0065	10.8702	10.8756	0.0311	1.1638
Experiment 3B-3c-4	0.0061	10.4043	11.0303	0.0254	1.1630
Experiment 3B-3d-1	0.0057	11.2073	11.1687	0.0833	1.1572
Experiment 3B-3d-2	0.0053	11.1781	11.1728	0.0837	1.1556
Experiment 3B-3d-3	0.0057	10.7097	11.1030	0.0845	1.1568
Experiment 3B-3d-4	0.0053	10.9675	11.1383	0.0853	0.0106
Average	0.0055	11.6473	11.0467	0.2115	0.8738
STD	0.0005	2.5270	0.2241	0.2841	0.5122
Percentage	0.02%	48.97%	46.44%	0.89%	3.67%

Unit: millisecond (ms)

In Experiment 3B-3a-1 and 3B-3a-2, ISAC made wrong decisions, and refused to complete the task because it finds that it cannot avoid the obstacle in order to push the box. The reason of causing the wrong decision is the same as in the Experiment 3B-2a-1 and 3B-2a-2. In these experiments, ISAC moved in the front of the obstacle, so the arm

collided with the obstacle. The success rate of the evaluation using the IRS in Experiment 3B-3 is 87.5%.

As shown in Table 33, the average running time for generating the behavior sequence in Experiment 3B-3 is 0.0055ms. The average running time for generating behaviors for the right and the left arm is 11.6473ms and 11.0467ms respectively.

The average running time of evaluating the generated behaviors for the right arm and for the left arm is 0.2115ms and 0.8738ms respectively. In the Experiments except 3B-3a-1, 3B-3a-2, 3B-3a-3, 3B-3a-4, and Experiment 3B-3d-4, ISAC needed to evaluate the overall motion trajectory of the left arm. The average time for evaluating the behaviors for the left arm in these experiments is 1.1601ms.

The running time of generating behavior sequence is 0.02% of the overall running time of the key components. 48.97% and 46.44% of the overall running time is used to generate motion trajectories for the right arm and the left arm respectively. The evaluation time is 0.89% and 3.67% of the overall running time.

--*Experiment 3B-4*

In Experiment 3B-4a-3, 3B-4a-4, 3B-4b-4, and 3B-4d-4, ISAC found that it cannot push the box using either of its arms if it wanted to avoid the obstacle. So it displayed a message on the screen and refused to complete the task. The rate of feasibility in Experiment 3B-4 is 75%.

In Experiment 3B-4a-3 and 3B-4b-4, ISAC made wrong decisions, and refused to complete the task because it found that it cannot avoid the obstacle in order to push the box. The reason of causing the wrong decision is the same as in the Experiment 3B-1a-3 and 3B-1a-4. In these experiments, ISAC cannot reach the box because the impedance

potential field of the obstacle increases. The success rate of the evaluation using the IRS in Experiment 3B-4 is 87.5%.

Table 34 Simulation Results of Experiment 3B-4

	Feasible/ Infeasible	Left/Right	Failure Reason	Evaluation
Experiment 3B-4a-1	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-4a-2	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-4a-3	Infeasible	N/A	Right Arm: Collision with the object Left Arm: Cannot reach the box	Wrong
Experiment 3B-4a-4	Infeasible	N/A	Right Arm: Collision with the object Left Arm: Collision with the obstacle	Correct
Experiment 3B-4b-1	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-4b-2	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-4b-3	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-4b-4	Infeasible	N/A	Right Arm: Collision with the object Left Arm: Cannot reach the box	Wrong
Experiment 3B-4c-1	Feasible	Left Arm	Right Arm: Collision with the obstacle	Correct
Experiment 3B-4c-2	Feasible	Left Arm	Right Arm: Collision with the obstacle	Correct
Experiment 3B-4c-3	Feasible	Left Arm	Right Arm: Collision with the obstacle	Correct
Experiment 3B-4c-4	Feasible	Left Arm	Right Arm: Collision with the obstacle	Correct
Experiment 3B-4d-1	Feasible	Left Arm	Right Arm: Collision with the obstacle	Correct
Experiment 3B-4d-2	Feasible	Left Arm	Right Arm: Collision with the obstacle	Correct
Experiment 3B-4d-3	Feasible	Left Arm	Right Arm: Collision with the obstacle	Correct
Experiment 3B-4d-4	Infeasible	N/A	Right Arm: Collision with the obstacle Left Arm: Collision with the obstacle	Correct

Table 35 Running Time of Key Components in Experiment 3B-4

	BSG	BGRA	BGLA	IRSRA	IRSLA
Experiment 3B-4a-1	0.0049	12.5287	10.8912	0.2159	1.0870
Experiment 3B-4a-2	0.0049	11.0816	23.5251	0.2294	1.1634
Experiment 3B-4a-3	0.0057	10.4700	10.8575	0.2290	1.1519
Experiment 3B-4a-4	0.0049	10.5710	11.4142	0.2725	1.1507
Experiment 3B-4b-1	0.0049	10.6403	10.5722	0.2298	1.2089
Experiment 3B-4b-2	0.0049	10.7380	10.6789	0.2159	1.0883
Experiment 3B-4b-3	0.0049	10.6543	10.6921	0.2315	1.1416
Experiment 3B-4b-4	0.0057	10.8940	10.6325	0.2352	1.0899
Experiment 3B-4c-1	0.0049	11.0320	10.5861	0.2303	1.1412
Experiment 3B-4c-2	0.0057	10.3792	10.5295	0.2298	1.1416
Experiment 3B-4c-3	0.0049	10.2836	10.3784	0.2298	1.1429
Experiment 3B-4c-4	0.0049	11.1022	10.3682	0.2631	1.1429
Experiment 3B-4d-1	0.0053	10.7491	10.4322	0.2315	1.1424
Experiment 3B-4d-2	0.0049	10.6875	10.6149	0.2307	0.3431
Experiment 3B-4d-3	0.0053	10.6543	10.4384	0.2906	0.0935
Experiment 3B-4d-4	0.0061	10.8797	10.3887	0.2771	0.0127
Average	0.0052	10.8341	11.4375	0.2401	0.9526
Standard Deviation	0.0004	0.5100	3.2339	0.0225	0.4044
Percentage	0.02%	46.16%	48.73%	1.02%	4.06%

As shown in Table 35, the average generation time for generating behavior sequence in Experiment 3B-4 is 0.0052ms. The average running time for generating behaviors for the right and the left arm is 10.8341ms and 11.4375ms respectively.

The average running time of evaluating the generated behaviors for the right arm and for the left arm is 0.2401ms and 0.9526ms respectively. In the Experiments except 3B-4a-3, 3B-4a-4, 3B-4b-4, and 3B-4d-44, ISAC needed to evaluate the overall motion trajectory of the left arm. The average time for evaluating the behaviors for the left arm in these experiments is: 0.9864ms.

The running time of generating behavior sequence is 0.02% of the overall running time of the key components. 46.16% and 48.73% of the overall running time is used to generate motion trajectories for the right arm and the left arm respectively. The evaluation time is 1.02% and 4.06% of the overall running time.

--*Experiment 3B-5*

In Experiment 3B-5a-3, 3B-5a-4, 3B-5b-4, and 3B-5d-4, ISAC found that it cannot push the box using either of its arms if it wanted to avoid the obstacle. So it displayed a message on the screen and refused to complete the task. The rate of feasibility in Experiment 3B-5 is 75%.

In Experiment 3B-5b-4, ISAC made wrong decisions, and refused to complete the task because it finds that it cannot avoid the obstacle to push the box. The reason of causing the wrong decision in Experiment 5b-4 is the same as in the Experiment 3B-1a-3 and 3B-1a-4. The success rate of the evaluation using the IRS in Experiment 3B-5 is 93.75%.

Table 36 Simulation Results of Experiment 3B-5

	Feasible/ Infeasible	Left/Right	Failure Reason	Evaluation
Experiment 3B-5a-1	Feasible	Right Arm	N/A	Correct
Experiment 3B-5a-2	Feasible	Right Arm	N/A	Correct
Experiment 3B-5a-3	Infeasible	N/A	Right Arm: Cannot Reach the Box Left Arm: Cannot Reach the Box	Correct
Experiment 3B-5a-4	Infeasible	N/A	Right Arm: Cannot Reach the Box Left Arm: Cannot Reach the Box	Correct
Experiment 3B-5b-1	Feasible	Right Arm	N/A	Correct
Experiment 3B-5b-2	Feasible	Right Arm	N/A	Correct
Experiment 3B-5b-3	Feasible	Right Arm	N/A	Correct
Experiment 3B-5b-4	Infeasible	Left Arm	Right Arm: Cannot Reach the Box Left Arm: Cannot Reach the Box	Wrong
Experiment 3B-5c-1	Feasible	Right Arm	N/A	Correct
Experiment 3B-5c-2	Feasible	Right Arm	N/A	Correct
Experiment 3B-5c-3	Feasible	Right Arm	N/A	Correct
Experiment 3B-5c-4	Feasible	Right Arm	N/A	Correct
Experiment 3B-5d-1	Feasible	Right Arm	N/A	Correct
Experiment 3B-5d-2	Feasible	Right Arm	N/A	Correct
Experiment 3B-5d-3	Feasible	Right Arm	N/A	Correct
Experiment 3B-5d-4	Infeasible	Right Arm	Right Arm: Collision with the obstacle Left Arm: Cannot Reach the Box	Correct

As shown in Table 37, the average running time for generating behavior sequence in Experiment 3B-5 is 0.0049ms. The average running time for generating behaviors for the right and the left arm is 11.6536ms and 10.7647ms respectively.

Table 37 Running Time of Key Components in Experiment 3B-5

	BSG	BGRA	BGLA	IRSRA	IRSLA
Experiment 3B-5a-1	0.0061	13.3260	10.8148	0.7997	0.0000
Experiment 3B-5a-2	0.0045	10.7914	10.1132	0.7738	0.0004
Experiment 3B-5a-3	0.0049	10.1916	10.2397	0.7964	0.2348
Experiment 3B-5a-4	0.0049	10.4371	11.0381	0.7984	0.2672
Experiment 3B-5b-1	0.0045	11.0533	10.7787	0.8300	0.0004
Experiment 3B-5b-2	0.0045	10.8756	10.8197	0.8140	0.0000
Experiment 3B-5b-3	0.0049	11.1030	10.6465	0.8046	0.0004
Experiment 3B-5b-4	0.0049	10.8243	11.3046	0.7988	0.2348
Experiment 3B-5c-1	0.0053	10.9215	10.8403	0.7951	0.0004
Experiment 3B-5c-2	0.0045	22.1186	10.8321	0.7960	0.0004
Experiment 3B-5c-3	0.0049	10.8649	11.1687	0.8321	0.0000
Experiment 3B-5c-4	0.0049	10.9671	10.8132	0.7951	0.0000
Experiment 3B-5d-1	0.0045	10.5143	10.8468	0.7726	0.0000
Experiment 3B-5d-2	0.0049	11.1075	10.6440	0.8387	0.0000
Experiment 3B-5d-3	0.0053	10.5123	10.8813	0.8300	0.0000
Experiment 3B-5d-4	0.0049	10.8489	10.4540	0.8350	0.2422
Average	0.0049	11.6536	10.7647	0.8069	0.0613
STD	0.0004	2.8729	0.3048	0.0209	0.1096
Percentage	0.02%	49.64%	45.85%	3.44%	1.05%

Unit: millisecond (ms)

The average running time of evaluating the generated behaviors for the right arm and for the left arm is 0.8069ms and 0.0613ms respectively. In Experiment 3B-1a-1, 3B-1a-2, 3B-1b-1, 3B-1b-2, 3B-1b-3, 3B-1c-1, 3B-1c-2, 3B-1c-3, 3B-1c-4, 3B-1d-1, 3B-1d-2, and 3B-1d-3, ISAC evaluated the overall motion trajectory of the right arm The

average time for evaluating the behaviors for the right arm in these experiments is: 0.8068ms.

The running time of generating behavior sequence is 0.02% of the overall running time of the key components. 49.64% and 45.85% of the overall running time is used to generate motion trajectories for the right arm and the left arm respectively. The evaluation time is 3.44% and 1.05% of the overall running time.

--*Experiment 3B-6*

In Experiment 3B-6a-4, ISAC found that it cannot push the box using either of its arms if it wanted to avoid the obstacle. So it displayed a message on the screen and refused to complete the task. The rate of the feasibility in Experiment 3B-6 is 93.75%.

In all experiments of Experiment 3B-6, ISAC made correct decisions. The success rate of the IRS evaluation is 100%.

As shown in Table 39, the average running time for generating the behavior sequence in Experiment 3B-6 is 0.0054ms. The average running time for generating behaviors for the right and left arm is 15.9670ms and 11.9227ms respectively. In Experiment 3B-3c-3, the running time for generating behaviors for the right arm is 79.1158. This is due to the interrupt generated by the operating system kernel.

The average running time of evaluating the generated behaviors for the right arm and for the left arm is 0.8134ms and 1.0514ms respectively. In Experiment 3B-6a-1, 3B-6a-2, 3B-6b-1, 3B-6b-2, 3B-6b-3, 3B-6c-1, 3B-6c-2, 3B-6c-3, 3B-6c-4, 3B-6d-1, and 3B-6d-2, ISAC evaluated the overall motion trajectory of the right arm The average time for evaluating the behaviors for the right arm in these experiments is: 0.9124ms. In Experiment 3B-6a-3, 3B-6b-4, 3B-6d-3, and 3B-6d-4, ISAC evaluated the overall motion

trajectory of the right arm. The average time for evaluating the behaviors for the right arm in these experiments is: 1.1295ms.

Table 38 Simulation Results of Experiment 3B-6

	Feasible/ Infeasible	Left/Right	Failure Reason	Evaluation
Experiment 3B-6a-1	Feasible	Right Arm	N/A	Correct
Experiment 3B-6a-2	Feasible	Right Arm	N/A	Correct
Experiment 3B-6a-3	Feasible	Left Arm	N/A	Correct
Experiment 3B-6a-4	Infeasible	N/A	Right Arm: Cannot Reach the Box Left Arm: Cannot Reach the Box	Correct
Experiment 3B-6b-1	Feasible	Right Arm	N/A	Correct
Experiment 3B-6b-2	Feasible	Right Arm	N/A	Correct
Experiment 3B-6b-3	Feasible	Right Arm	N/A	Correct
Experiment 3B-6b-4	Feasible	Left Arm	Right Arm: Cannot Reach the Box	Correct
Experiment 3B-6c-1	Feasible	Right Arm	N/A	Correct
Experiment 3B-6c-2	Feasible	Right Arm	N/A	Correct
Experiment 3B-6c-3	Feasible	Right Arm	N/A	Correct
Experiment 3B-6c-4	Feasible	Right Arm	N/A	Correct
Experiment 3B-6d-1	Feasible	Right Arm	N/A	Correct
Experiment 3B-6d-2	Feasible	Right Arm	N/A	Correct
Experiment 3B-6d-3	Feasible	Left Arm	Right Arm: Cannot reach the Box	Correct
Experiment 3B-6d-4	Feasible	Left Arm	Right Arm: Cannot reach the Box	Correct

Table 39 Running Time of Key Components in Experiment 3B-6

	BSG	BGRA	BGLA	IRSRA	IRSLA
Experiment 3B-6a-1	0.0049	11.1079	11.0504	0.8231	0.0004
Experiment 3B-6a-2	0.0053	11.3571	11.0981	0.8616	0.0004
Experiment 3B-6a-3	0.0057	11.2147	12.3863	0.9655	1.1970
Experiment 3B-6a-4	0.0049	12.2537	12.1227	0.8941	1.1461
Experiment 3B-6b-1	0.0053	13.4459	13.2205	0.9971	0.0004
Experiment 3B-6b-2	0.0049	13.5325	11.4606	0.8555	0.0004
Experiment 3B-6b-3	0.0053	11.1404	12.1285	1.3108	0.0004
Experiment 3B-6b-4	0.0065	13.9360	12.9672	0.8670	1.1400
Experiment 3B-6c-1	0.0057	11.0217	11.5845	0.8526	0.0000
Experiment 3B-6c-2	0.0053	11.1941	11.5804	0.8543	0.0004
Experiment 3B-6c-3	0.0049	79.1158	17.1488	1.0242	0.0004
Experiment 3B-6c-4	0.0057	12.9738	11.0492	0.8009	0.0004
Experiment 3B-6d-1	0.0053	10.8091	10.8616	0.8345	0.0000
Experiment 3B-6d-2	0.0061	11.4084	10.8128	0.8214	0.0000
Experiment 3B-6d-3	0.0049	10.4601	10.7758	0.1588	1.0969
Experiment 3B-6d-4	0.0053	10.5012	10.5159	0.0931	0.6769
Average	0.0054	15.9670	11.9227	0.8134	1.0514
STD	0.0005	16.8770	1.6059	0.2959	0.2123
Percentage	0.02%	53.65%	40.06%	2.73%	3.53%

The running time of generating behavior sequence is 0.02% of the overall running time of the key components. 53.65% and 40.06% of the overall running time is used to generate motion trajectories for the right arm and the left arm respectively. The evaluation time is 2.73% and 3.53% of the overall running time.

--*Experiment 3B-7*

Table 40 Simulation Results of Experiment 3B-7

	Feasible/ Infeasible	Left/Right	Failure Reason	Evaluation
Experiment 3B-7a-1	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-7a-2	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-7a-3	Infeasible	N/A	Right Arm: Cannot Reach the Box Left Arm: Cannot Reach the Box	Wrong
Experiment 3B-7a-4	Infeasible	N/A	Right Arm: Cannot Reach the Box Left Arm: Cannot Reach the Box	Correct
Experiment 3B-7b-1	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-7b-2	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-7b-3	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-7b-4	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-7d-1	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-7d-2	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-7d-3	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-7d-4	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-7d-1	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-7d-2	Feasible	Left Arm	Right Arm: Collision with the obstacle	Correct
Experiment 3B-7d-3	Feasible	Left Arm	Right Arm: Collision with the obstacle	Correct
Experiment 3B-7d-4	Infeasible	N/A	Right Arm: Collision with the obstacle Left Arm: Collision with the object	Correct

In Experiment 3B-7a-3, 3B-7a-4, and 3B-7d-4, ISAC found that it cannot push the box using either of its arms if it wanted to avoid the obstacle. So it displayed a

message on the screen and refused to complete the task. The rate of the feasibility in Experiment 3B-7 is 81.25%.

In Experiment 3B-7a-3, ISAC made wrong decisions, and refused to complete the task because it finds that it cannot avoid the obstacle to push the box. The reason of causing the wrong decision in Experiment 3B-7a-3 is the same as in the Experiment 3B-1a-3 and 3B-1a-4. The success rate of the evaluation using the IRS in Experiment 3B-7 is 93.75%.

As shown in Table 41, the average running time for generating the behavior sequence in Experiment 3B-7 is 0.0053ms. The average running time for generating behaviors for the right and the left arm is 11.0320 and 11.9550ms respectively.

The average running time of evaluating the generated behaviors for the right arm and for the left arm is 0.8134ms and 1.0514ms respectively. In the experiments except Experiment 3B-7a-3, 3B-7a-4, and 3B-7d-4, ISAC evaluated the overall motion trajectory of the left arm The average time for evaluating the behaviors for the right arm in these experiments is: 1.1329ms.

The running time of generating behavior sequence is 0.02% of the overall running time of the key components. 44.84% and 48.60% of the overall running time is used to generate motion trajectories for the right arm and the left arm respectively. The evaluation time is 1.94% and 4.60% of the overall running time.

Table 41 Running Time of Key Components in Experiment 3B-7

	BSG	BGRA	BGLA	IRSRA	IRSLA
Experiment 3B-7a-1	0.0049	10.8419	10.4819	0.4351	1.1383
Experiment 3B-7a-2	0.0041	10.5935	10.7196	0.4384	1.1392
Experiment 3B-7a-3	0.0053	11.1502	21.1058	0.8210	1.1047
Experiment 3B-7a-4	0.0053	10.4798	11.2036	0.8834	1.1231
Experiment 3B-7b-1	0.0057	10.8173	10.3977	0.4536	1.1757
Experiment 3B-7b-2	0.0053	10.4150	10.5258	0.4273	1.1026
Experiment 3B-7b-3	0.0053	10.6120	10.4843	0.4298	1.0850
Experiment 3B-7b-4	0.0057	10.6440	10.8567	0.8284	1.0887
Experiment 3B-7c-1	0.0053	10.7898	10.4889	0.4372	1.1227
Experiment 3B-7c-2	0.0053	11.0217	12.5411	0.4367	1.1457
Experiment 3B-7c-3	0.0053	13.2944	16.5819	0.4372	1.1396
Experiment 3B-7c-4	0.0061	11.5082	11.1313	0.4515	1.1461
Experiment 3B-7d-1	0.0049	11.2750	11.2496	0.4363	1.1445
Experiment 3B-7d-2	0.0049	10.9150	11.4577	0.0989	1.1572
Experiment 3B-7d-3	0.0053	10.9638	11.1843	0.1420	1.1420
Experiment 3B-7d-4	0.0061	11.1904	10.8694	0.4782	1.1494
Average	0.0053	11.0320	11.9550	0.4772	1.1315
Standard Deviation	0.0005	0.6737	2.8628	0.2126	0.0252
Percentage	0.02%	44.84%	48.60%	1.94%	4.60%

--Experiment 3B-8

Table 42 Simulation Results of Experiment 3B-8

	Feasible/ Infeasible	Left/Right	Failure Reason	Evaluation
Experiment 3B-8a-1	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-8a-2	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-8a-3	Infeasible	N/A	Right Arm: Collision with the object Left Arm: Cannot reach the box	Wrong
Experiment 3B-8a-4	Infeasible	N/A	Right Arm: Collision with the object Left Arm: Cannot reach the box	Wrong
Experiment 3B-8b-1	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-8b-2	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-8b-3	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-8b-4	Infeasible	N/A	Right Arm: Collision with the object Left Arm: Cannot reach the box	Wrong
Experiment 3B-8c-1	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-8c-2	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-8c-3	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-8c-4	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-8d-1	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-8d-2	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-8d-3	Feasible	Left Arm	Right Arm: Collision with the object	Correct
Experiment 3B-8d-4	Infeasible	N/A	Right Arm: Collision with the object Left Arm: Collision with the obstacle	Correct

In Experiment 3B-8a-3, 3B-8a-4, 3B-8b-4, and 3B-8d-4, ISAC found that it cannot push the box using either of its arms if it wanted to avoid the obstacle. So it

displayed a message on the screen and refused to do. The rate of feasibility in Experiment 3B-8 is 75%.

In Experiment 3B-8a-3, 3B-8a-4, and 3B-8b-4, ISAC made wrong decisions, and refused to complete the task because it finds that it cannot avoid the obstacle to push the box. The reason of causing the wrong decision in Experiment 3B-7a-3 is the same as in the Experiment 3B-2 The success rate of the evaluation using the IRS in Experiment 3B-7 is 81.25%.

As shown in Table 43, the average running time for generating the behavior sequence in Experiment 3B-8 is 0.0053ms. The average running time for generating behaviors for the right and the left arm is 10.8341ms and 11.4375ms respectively.

The average running time of evaluating the generated behaviors for the right arm and for the left arm is 0.2401ms and 0.9526ms respectively. In the experiments except Experiment 3B-8a-3, 3B-8a-4, 3B-8b-4, and 3B-8d-4, ISAC evaluated the overall motion trajectory of the left arm. The average time for evaluating the behaviors for the right arm in these experiments is: 0.9864ms.

The running time of generating behavior sequence is 0.02% of the overall running time of the key components. 46.16% and 48.73% of the overall running time is used to generate motion trajectories for the right arm and the left arm respectively. The evaluation time is 1.02% and 4.06% of the overall running time.

Table 43 Running Time of Key Components in Experiment 3B-8

	BSG	BGRA	BGLA	IRSRA	IRSLA
Experiment 3B-8a-1	0.0049	12.5287	10.8912	0.2159	1.0870
Experiment 3B-8a-2	0.0049	11.0816	23.5251	0.2294	1.1634
Experiment 3B-8a-3	0.0057	10.4700	10.8575	0.2290	1.1519
Experiment 3B-8a-4	0.0049	10.5710	11.4142	0.2725	1.1507
Experiment 3B-8b-1	0.0049	10.6403	10.5722	0.2298	1.2089
Experiment 3B-8b-2	0.0049	10.7380	10.6789	0.2159	1.0883
Experiment 3B-8b-3	0.0049	10.6543	10.6921	0.2315	1.1416
Experiment 3B-8b-4	0.0057	10.8940	10.6325	0.2352	1.0899
Experiment 3B-8c-1	0.0049	11.0320	10.5861	0.2303	1.1412
Experiment 3B-8c-2	0.0057	10.3792	10.5295	0.2298	1.1416
Experiment 3B-8c-3	0.0049	10.2836	10.3784	0.2298	1.1429
Experiment 3B-8c-4	0.0049	11.1022	10.3682	0.2631	1.1429
Experiment 3B-8d-1	0.0053	10.7491	10.4322	0.2315	1.1424
Experiment 3B-8d-2	0.0049	10.6875	10.6149	0.2307	0.3431
Experiment 3B-8d-3	0.0053	10.6543	10.4384	0.2906	0.0935
Experiment 3B-8d-4	0.0061	10.8797	10.3887	0.2771	0.0127
Average	0.0052	10.8341	11.4375	0.2401	0.9526
Standard Deviation	0.0004	0.5100	3.2339	0.0225	0.4044
Percentage	0.02%	46.16%	48.73%	1.02%	4.06%

D. Major Software Program Files and Functions

Speech

File: DeliverationVoiceNov.cs

Function: *void rec_SpeechRecognized(object sender, SpeechRecognizedEventArgs e)*

This function is triggered by the event of recognized sentences which obey the rules of the grammar predefined. According to different grammars, this function chose different actions (e.g., learning, generalization, generation, etc.) for ISAC.

Vision

File: Perception.cs

Function: void myKinectSensor_AllFramesReady(object sender, AllFramesReadyEventArgs e)

This Function is triggered by the event of all frames from the Kinect Sensor is ready for processing. Two types of information is generated and stored in two global shared arrays respectively:

double [4] objectPosition

double [3] handRightPosition

objectPosition stores the position values of the detected target object:

objectPosition [0] is the value on the X-Axis

objectPosition [1] is the value on the Y-Axis

objectPosition [2] is the value on the Z-Axis

handRightPosition stores the position values of the detected right hand of the human body:

handRightPosition [0] is the value on the X-Axis

handRightPosition [1] is the value on the Y-Axis

handRightPosition [2] is the value on the Z-Axis

Behavior Generalization

File: FeatureAnalysis.m

Function: *function [FeaturePreIndex,FeatureInternalIndex,FeaturePostIndex] = FeatureAnalysis(behaviorName,varargin)*

This function generalizes the common features of a basic behavior.

The input of this function is a behavior name and related parameters. The output is a 3-dimensional vector. The first element of the vector is the number related the most common feature of the Pre-Condition, the second element of the vector is the number related the most common feature of the Internal-Constraint, The third element of the vector is the number related the most common feature of the Post-Result.

Behavior Sequence Generation

File: BehaviorGraph.cs

Function: *public ArrayList FindPath(string destination)*

This function finds a path in the behavior graph to generate a behavior sequence, the destination of which is the required behavior.

The input parameter is the name of a required behavior. The output is a *ArrayList* which is a behavior sequence. Each element in this *ArrayList* is a behavior.

Motion Trajectory Generation

File: BasicBehaviors.cs

Function: *public void GenerateBehaviors(ArrayList behaviorList, ArrayList parameterList)*

This function generated via points of the motion trajectories of all the behaviors in a behavior sequence.

The input is two *ArrayList*: the first is the *behaviorList* which stores the name of the behaviors in the behavior sequence; the second is *parameterList* which stores the task-related parameters. Three global shared *ArrayList*: generatedTrajectoryX, generatedTrajectoryY, and generatedTrajectoryZ store X, Y, and Z value of the generated via points for the behavior sequence.

Internal Rehearsal

File: InternalRehearsalSystem.cs

Function: *public bool WorkspaceChecking(double px,double py,double pz)*

This function checks whether the via point (the input) is within the working space of ISAC.

The input is three *double* values which are the X, Y, and Z values of the via point respectively. The output is a Boolean value. If it is true, the point is within the working space of ISAC; If it is false, the point is out of the working space of ISAC.

Function: *public bool CollisionChecking(double px, double py, double pz)*

This Function whether the arm of ISAC collides with the obstacle given the via point (the input).

The input is three *double* values which are the X, Y, and Z values of the via point respectively. The output is a Boolean value. If it is true, collision does not happen; If it is false, collision Happens.

Arm Control

File: ArmControl.cpp

Function: *void InverseKinematicsRight()*

This function computes the joint angles of the right arm given the position and orientation of the right end-effector of ISAC.

Function: *void InverseKinematicsLeft()*

This function computes the joint angles of the left arm given the position and orientation of the left end-effector of ISAC.

Function: *void PIDRight()*

This function computes the required voltage of the regulators to control the right arm of ISAC.

Function: *void PIDLeft()*

This function computes the required voltage of the regulators to control the left arm of ISAC.

Function: *void PressureOutputRight()*

This function changes the voltages of the regulators which are used control the air muscles of the right arm of ISAC.

Function: *void PressureOutputLeft()*

This function changes the voltages of the regulators which are used control the air muscles of the left arm of ISAC.

ISAC Initialization

The following figure displays how to initialize ISAC.

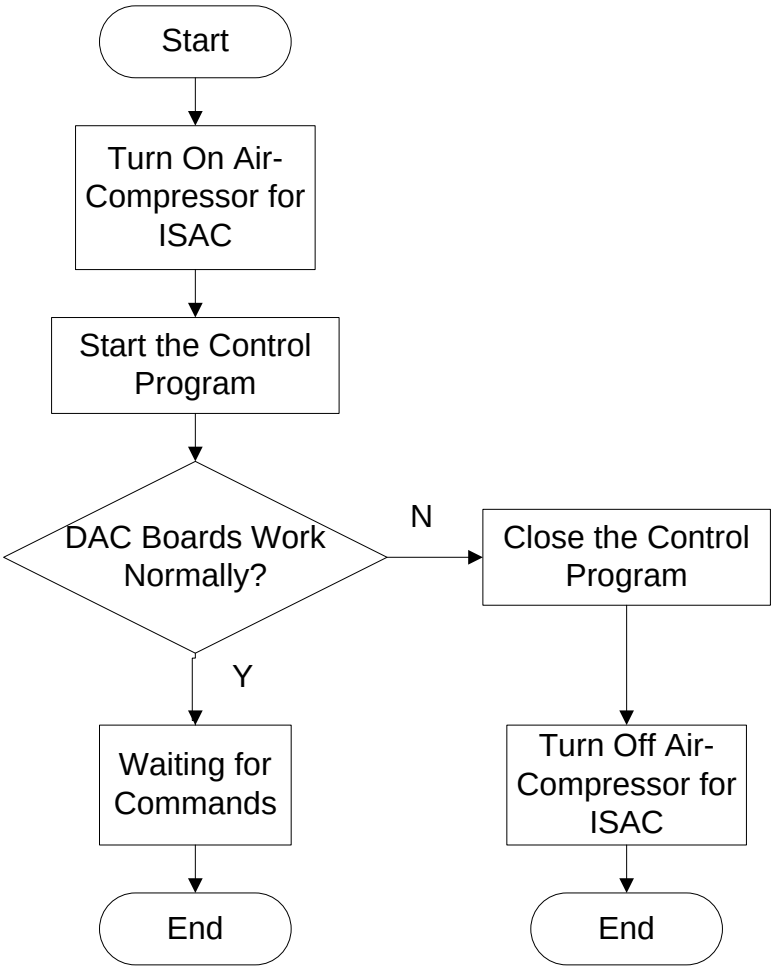


Figure 284 ISAC Starting Up

1. Turn on the Freezer

Figure 285 Freezer

2. Turn on the air valve on the air-tank by rotating counter-clockwise.

Figure 286 Air-Tank

3. Flip the switch on the left electrical control cabinet to turn on it

4. Flip the switch on the right electrical control cabinet

5. Press the Reset Button on the right electrical control cabinet to turn on it

6. Rotate the key on the right electrical control cabinet 90 degrees clockwise to turn on it

Figure 287 Electrical Control Cabinet

7. Turn on the regulators

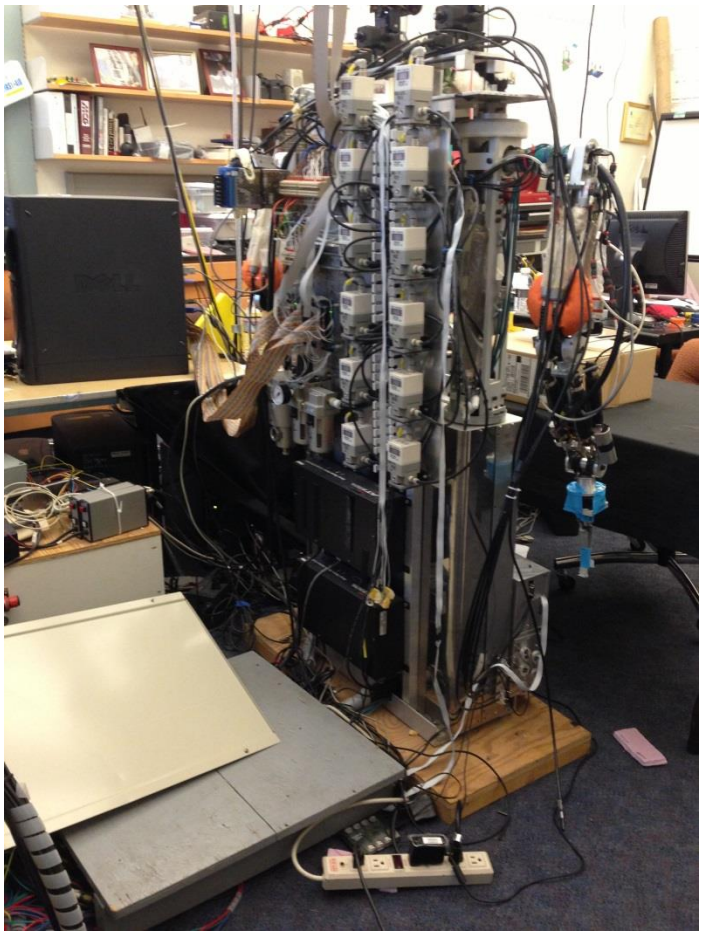


Figure 288 Regulators

8. Start Computer “Octavia”

9. Login on “Octavia” using your own VUNetID and corresponding Password

10. Open Visual C++ 6.0 by selecting “Microsoft Visual C++ 6.0” from the “Start” menu of Windows

11. Press “File->Open Workspace…”

12. Select the file “C:\Documents and Settings\User\Desktop\ArmControl\ArmControl.dsw” and open it by double clicking it.

13. Press Ctrl+F5 to run the program, program will check the DAC boards automatically. If it does not work normally, the program will terminate.

14. If the DAC boards works normally, after 20 seconds, the control program moves the arms to the home position and waiting for the future commands from other programs and from the manual input

Deliberation Program

1. Double click DeliberationVoiceNov.sln to open it
2. Press F5 to run the program

Perception Program

1. Connect the USB cable of the Kinect to the Computer Sally
2. Insert the power cable of the Kinect to a power cord outlet
3. Double click Perception.sln to open it
4. Press F5 to run the program

Usage

Learning

1. Give ISAC a speech command: "I will show you how to use the "××" behavior." ×× is the behavior the human teacher wants to demonstrate, e.g., *Reaching the Object*, *Push the Object*, etc.
2. Give ISAC a speech command: "That's the ×× demonstration."

×× is the index number of the demonstration, e.g., first, second, etc.

3. Give ISAC the speech command: “I am ready”.

   ISAC starts to record the motion of the hand of the human teacher when hearing this command. (Before giving this command, make sure the skeleton of the human teacher is displayed on the dialog of the perception program.)

4. Give ISAC the speech command: “Stop Recording”.

   ISAC stop recording the motion of the hand of the human teacher when hearing this command.

5. Given ISAC the speech command: “That’s the end of the demonstrations”.

   ISAC start generalizing the demonstrations.

   The human teacher needs to repeat step 2-4 to give ISAC several demonstrations for it to generalize.

Generation

1. Put a box on the table in front of ISAC.

   Make sure the position values of the box are displayed on the dialog of the perception program. If there are no position values on the dialog, the box is outside of the working space of ISAC.

2. Given ISAC a speech command: “Could you push the box to the right” or “Could you push the box to the left”.

   ISAC starts to push the box.

REFERENCES

Aiyama, Y., M. Inaba, et al. (1993) Pivoting: A New Method of Graspless Manipulation of Object by Robot Fingers. *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo, Japan, 136-143.

Albus, J. and A. Barbera (2005) Rcs: A Cognitive Architecture for Intelligent Multi-Agent Systems. *Annual Reviews in Control*, **29**, 87-99.

Albus, J. S. (1991) Outline for a Theory of Intelligence. *IEEE Transactions on Systems, Man and Cybernetics*, **21**, 473-509.

Ambrose, R. O., H. Aldridge, et al. (2000) Robonaut: Nasa's Space Humanoid. *IEEE Trasaction on Intelligent Systems and their Applications*, **15**, 57-63.

Amit, R. and M. Matari (2002) Learning Movement Sequences from Demonstration. *Proceedings of the Second International Conference on Development and Learning*, Cambridge, Massachusetts, USA, 203-208.

An, M., T. Taura, et al. (2007) A Study on Acquiring Underlying Behavioral Criteria for Manipulator Motion by Focusing on Learning Efficiency. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, **37**, 445-455.

Anderson, J. and C. Lebiere (1998) The Atomic Components of Thought. Lawrence Erlbaum.

Argall, B., Chernova, S, Veloso, M, and Browning, B (2009) A Survey of Robot Learning from Demonstration. *Robotics and Autonomous Systems*, **57**, 469-483.

Arikan, O. and D. Forsyth (2002) Interactive Motion Generation from Examples. *ACM Transactions on Graphics*, **21**, 483-490.

Asada, M., K. MacDorman, et al. (2001) Cognitive Developmental Robotics as a New Paradigm for the Design of Humanoid Robots. *Robotics and Autonomous Systems*, **37**, 185-193.

Atkeson, C., J. Hale, et al. (2000) Using Humanoid Robots to Study Human Behavior. *IEEE Intelligent Systems and their applications*, **15**, 46-56.

Atkeson, C. and J. McIntyre (1986) Robot Trajectory Learning through Practice. *Proceedings of the 1986 IEEE Conference on Robotics and Automation*, San Francisco, California, USA, 1737-1742.

Atkeson, C., A. Moore, et al. (1997) Locally Weighted Learning. *Artificial intelligence review*, **11**, 11-73.

Atkeson, C. and S. Schaal (1997) Learning Tasks from a Single Demonstration. *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, Albuquerque, New Mexico, USA, 1706-1712.

Atkeson, C. and S. Schaal (1997) Robot Learning from Demonstration. *Proceedings of the Fourteenth International Conference on Machine Learning*, San Francisco, California, USA, 11-73.

Badre, D. (2008) Cognitive Control, Hierarchy, and the Rostro-Caudal Organization of the Frontal Lobes. *Trends in cognitive sciences*, **12**, 193-200.

Banich, M. T., K. L. Mackiewicz, et al. (2009) Cognitive Control Mechanisms, Emotion and Memory: A Neural Perspective with Implications for Psychopathology. *Neuroscience & Biobehavioral Reviews*, **33**, 613-630.

Bartholomew, D. (1984) The Foundations of Factor Analysis. *Biometrika*, **71**, 221-232.
Beer, R. (2000) Dynamical Approaches to Cognitive Science. *Trends in cognitive sciences*, **4**, 91-99.
Begley, S. M. (2008) Gesture Recognition and Mimicking in a Humanoid Robot. Vanderbilt University.
Begum, M. and F. Karray (2011) Visual Attention for Robotic Cognition: A Survey. *IEEE Transactions on Autonomous Mental Development*, **3**, 92-105.
Belkin, M. and P. Niyogi (2003) Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Computation*, **15**, 1373-1396.
Benner, P., V. Mehrmann, et al. (2005) Dimension Reduction of Large-Scale Systems. Springer Berlin, Heidelberg.
Bentivegna, D. and C. Atkeson (2001) Learning from Observation Using Primitives. *Proceedings of the 2001 International Conference on Robotics and Automation*, Seoul, Korea, 1988-1993.
Berndt, D. and J. Clifford (1994) Using Dynamic Time Warping to Find Patterns in Time Series. *Proceedings of the 1994 AAAI Workshop on Knowledge Discovery in Databases*, Seattle, Washington, USA.
Bicho, E. (2012) Towards More Natural Human-Robot Interaction: From Cognitive Sciences and Neurobiology to Socially Aware/Assistive Robotics. *Course Notes*, 29.
Bien, Z. and H. Lee (2007) Effective Learning System Techniques for Human-Robot Interaction in Service Environment. *Knowledge-Based Systems*, **20**, 439-456.
Billard, A. (2001) Learning Motor Skills by Imitation: A Biologically Inspired Robotic Model. *Cybernetics and Systems*, **32**, 155-193.
Billard, A., S. Calinon, et al. (2007) Robot Programming by Demonstration. *Handbook of robotics*. B. Siciliano and O. Khatib. New York, NY, USA, Springer.
Billard, A., S. Calinon, et al. (2006) Discriminative and Adaptive Imitation in Uni-Manual and Bi-Manual Tasks. *Robotics and Autonomous Systems*, **54**, 370-384.
Bishop, C., M. Svensén, et al. (1998) Gtm: The Generative Topographic Mapping. *Neural Computation*, **10**, 215-234.
Bitzer, S., I. Havoutis, et al. (2008) Synthesising Novel Movements through Latent Space Modulation of Scalable Control Policies. *Proceedings of the Tenth International Conference on Simulation of Adaptive Behaviour: From Animals to Animats*, Osaka, Japan, 199-209.
Bitzer, S., S. Klanke, et al. (2009) Does Dimensionality Reduction Improve the Quality of Motion Interpolation? *Proceedings of the Seventeenth European Symposium on Artificual Neural Networks*, Bruges, Belgium, 71–76.
Bitzer, S. and S. Vijayakumar (2009) Latent Spaces for Dynamic Movement Primitives. *Proceedings of the 2009 IEEE-RAS International Conference on Humanoid Robots*, Paris, France, 574-581.
Brock, O. and L. Kavraki (2001) Decomposition-Based Motion Planning: A Framework for Real-Time Motion Planning in High-Dimensional Configuration Spaces. *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, Seoul, Korea, 1469-1474.
Brooks, R. (1986) Achieving Artificial Intelligence through Building Robots. Cambridge, MA, USA, Massachusetts Institute of Technology Artificial Intelligence Laboratory.

Brooks, R. (1986) A Robust Layered Control System for a Mobile Robot. *IEEE journal of robotics and automation*, **2**, 14-23.

Brooks, R. (1991) How to Build Complete Creatures Rather Than Isolated Cognitive Simulators. *Architectures for Intelligence*. K. VanLehn. Hillsdale, NJ, Erlbaum, 225–239.

Brooks, R., C. Breazeal, et al. (1998) Alternative Essences of Intelligence. *Proceedings of the AAAI 1998*, Madison, Wisconsin, USA, 961-968.

Brooks, R., C. Breazeal, et al. (1999) The Cog Project: Building a Humanoid Robot. *Computation for Metaphors, Analogy, and Agents*. C. Nehaniv. Heidelberg, Berlin, Germany, Springer-Verlag, 52-87.

Burattini, E., A. Finzi, et al. (2011) Cognitive Control in Cognitive Robotics: Attentional Executive Control. *Proceedings of the 201115th International Conference on Advanced Robotics (ICAR)*, Tallinn, Estonia, 359-364.

Calinon, S. and A. Billard (2007) Incremental Learning of Gestures by Imitation in a Humanoid Robot. *Proceedings of the 2007 ACM/IEEE International Conference on Human-robot interaction*, New York, NY, USA, 255-262.

Calinon, S., F. Guenter, et al. (2007) On Learning, Representing, and Generalizing a Task in a Humanoid Robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, **37**, 286-298.

Carreira-Perpinán, M. (1997) A Review of Dimension Reduction Techniques. *University of Sheffield, Sheffield, UK, Tech. Rep. CS-96-09*.

Cassimatis, N., J. Trafton, et al. (2004) Integrating Cognition, Perception and Action through Mental Simulation in Robots. *Robotics and Autonomous Systems*, **49**, 13-23.

Chella, A., M. Frixione, et al. (1997) A Cognitive Architecture for Artificial Vision* 1. *Artificial Intelligence*, **89**, 73-111.

Chen, J. and A. Zelinsky (2003) Programing by Demonstration: Coping with Suboptimal Teaching Actions. *The International Journal of Robotics Research*, **22**, 299.

Chernova, S. and M. Veloso (2007) Confidence-Based Policy Learning from Demonstration Using Gaussian Mixture Models. *Proceedings of the Sixth international joint conference on Autonomous agents and multiagent systems*, New York, New York, USA, 1315-1322.

Conforth, M. and M. Yan (2011) Charisma: A Context Hierarchy-Based Cognitive Architecture for Self-Motivated Social Agents. *Proceedings of the 2011 International Joint Conference on Neural Networks (IJCNN)*, San Jose, California, USA, 1894-1901.

Crick, C., S. Osentoski, et al. (2011) Human and Robot Perception in Large-Scale Learning from Demonstration. *Proceedings of the Sixth ACM international conference on Human-robot interaction*, New York, New York, USA, 339-346.

Daerden, F. and D. Lefeber (2002) Pneumatic Artificial Muscles: Actuators for Robotics and Automation. *European journal of mechanical and environmental engineering*, **47**, 11-21.

Delson, N. and H. West (2002) Robot Programming by Human Demonstration: Adaptation and Inconsistency in Constrained Motion. *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, Washington, DC, USA, 30-36.

Demiris, J. and G. Hayes (1996) Imitative Learning Mechanisms in Robots and Humans. *DAI Research Paper*.

Deutsch, J. and D. Deutsch (1963) Attention: Some Theoretical Considerations. *Psychological review*, **70**, 80-90.

Dijkstra, E. W. (1959) A Note on Two Problems in Connexion with Graphs. *Numerische mathematik*, **1**, 269-271.

Dillmann, R., M. Kaiser, et al. (1995) Acquisition of Elementary Robot Skills from Human Demonstration. *Proceedings of the 1995 International Symposium on Intelligent Robotic System*, Pisa, Italy, 185-192.

Dillmann, R., O. Rogalla, et al. (2000) Learning Robot Behaviour and Skills Based on Human Demonstration and Advice: The Machine Learning Paradigm. *Proceedings of the Ninth International Symposium of Robotics Research,*, Snowbird, UT, USA, 229-238.

Dodd, W. (2005) The Design of Procedural, Semantic, and Episodic Memory Systems for a Cognitive Robot. EECS Department. Nashville, Vanderbilt University. **Master**.

Driscoll, J. A., R. Peters, et al. (1998) A Visual Attention Network for a Humanoid Robot. *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Victoria, Canada, 1968-1974.

Edelman, G. (1987) Neural Darwinism: The Theory of Neuronal Group Selection. New York, NY, USA, Basic Books.

Erdemir, E., C. Frankel, et al. (2008) A Robot Rehearses Internally and Learns an Affordance Relation. *Proceedings of the 2008 IEEE International Conference on Development and Learning*, Monterey, California, USA, 298-303.

Erlhagen, W. and E. Bicho (2006) The Dynamic Neural Field Approach to Cognitive Robotics. *Journal of neural engineering*, **3**, R36.

Estlin, T., R. Volpe, et al. (2001) Decision-Making in a Robotic Architecture for Autonomy. *Proceedings of the International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, Montreal, Canada.

Fong, T., I. Nourbakhsh, et al. (2003) A Survey of Socially Interactive Robots. *Robotics and Autonomous Systems*, **42**, 143-166.

Forte, D., A. Ude, et al. (2010) Robot Learning by Gaussian Process Regression. *Proceedings of the 2010 IEEE 19th International Workshop on Robotics in Alpe-Adria-Danube Region*, Balatonfüred, Hungary, 303-308.

Gams, A., A. J. Ijspeert, et al. (2009) On-Line Learning and Modulation of Periodic Movements with Nonlinear Dynamical Systems. *Autonomous Robots*, **27**, 3-23.

Glas, D., S. Satake, et al. (2011) An Interaction Design Framework for Social Robots. *Robotics: Science and Systems*, **7**, 89-96.

Gobet, F., P. Lane, et al. (2001) Chunking Mechanisms in Human Learning. *Trends in Cognitive Sciences*, **5**, 236-243.

Goodrich, M. and A. Schultz (2007) Human-Robot Interaction: A Survey. *Foundations and Trends in Human-Computer Interaction*, **1**, 203-275.

Gorbenko, A., V. Popov, et al. (2012) Robot Self-Awareness: Exploration of Internal States. *Applied Mathematical Sciences*, **6**, 675-688.

Gouaillier, D., V. Hugel, et al. (2009) Mechatronic Design of Nao Humanoid. *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, Kobe, Japan, 769-774.

Haazebroek, P., S. van Dantzig, et al. (2011) A Computational Model of Perception and Action for Cognitive Robotics. *Cognitive Processing*, **12**, 355-365.

Hall III, J. F. (2007) Internal Rehearsal for a Cognitive Robot Using Collision Detection. EECS Department. Nashville, Vanderbilt University. **Master**.

Hambuchen, K. A. (2004) Multi-Modal Attention and Event Binding in Humanoid Robots Using a Sensory Ego-Sphere. Vanderbilt University. **PhD**.

Hamker, F. (2012) Neural Learning of Cognitive Control. *KI - Künstliche Intelligenz*, **26**, 397-401.

Herzog, D., V. Krüger, et al. (2008) Parametric Hidden Markov Models for Recognition and Synthesis of Movements. *Proceedings of the 2008 British Machine Vision Conference*, Leeds, UK, 163–172.

Ho, E. S. L., T. Komura, et al. (2010) Controlling Humanoid Robots in Topology Coordinates. *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, China, 178-182.

Horowitz, R. (1993) Learning Control of Robot Manipulators. *American Society of Mechanical Engineers Journal of Dynamic Systems, Measure, and Control*, **115**, 1-32.

Hovland, G., P. Sikka, et al. (1996) Skill Acquisition from Human Demonstration Using a Hidden Markov Model. *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, Minneapolis, Minnesota, USA, 2706-2711.

Huntsberger, T. (2011) Cognitive Architecture for Mixed Human-Machine Team Interactions for Space Exploration. *Proceedings of the 2011 IEEE Aerospace Conference*, Big Sky, Montana, USA, 1-11.

Ijspeert, A., J. Nakanishi, et al. (2002) Learning Rhythmic Movements by Demonstration Using Nonlinear Oscillators. *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, 958–963.

Ijspeert, A., J. Nakanishi, et al. (2002) Movement Imitation with Nonlinear Dynamical Systems in Humanoid Robots. *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, Washington, DC, USA, 1398-1403.

Ijspeert, A., J. Nakanishi, et al. (2003) Learning Attractor Landscapes for Learning Motor Primitives. *Advances in Neural Information Processing Systems*, **15**, 1523-1530.

Inamura, T., M. Inaba, et al. (1999) Acquisition of Probabilistic Behavior Decision Model Based on the Interactive Teaching Method. *Proceedings of the Ninth International Conference on Advanced Robotics*, Tokyo, Japan, 523–528.

Inamura, T., H. Tanie, et al. (2003) Keyframe Compression and Decompression for Time Series Data Based on the Continuous Hidden Markov Model. *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, Nevada, USA, 1487-1492.

Inamura, T., I. Toshima, et al. (2003) Acquiring Motion Elements for Bidirectional Computation of Motion Recognition and Generation. *Experimental Robotics viii*. . Springer Berlin Heidelberg, 372-381.

Insaurralde, C. C., J. J. Cartwright, et al. (2012) Cognitive Control Architecture for Autonomous Marine Vehicles. *Proceedings of the 2012 IEEE International Systems Conference*, Vancouver, British Columbia, Canada, 1-8.

Jain, A. K., J. Mao, et al. (1996) Artificial Neural Networks: A Tutorial. *Computer*, **29**, 31-44.

Jenkins, O. C. and M. J. Matari (2004) A Spatio-Temporal Extension to Isomap Nonlinear Dimension Reduction. *Proceedings of the Twenty-First International Conference on Machine Learning*, Banff, Canada.

Jolliffe, I. (2002) Principal Component Analysis. New York, NY, Springer.
Kaiser, M. and R. Dillmann (1996) Building Elementary Robot Skills from Human Demonstration. *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, Minneapolis, Minnesota, USA, 2700-2705.
Kalman, R. E. (1960) A New Approach to Linear Filtering and Prediction Problems. *Journal of basic Engineering*, **82**, 35-45.
Kambhatla, N. and T. Leen (1993) Fast Nonlinear Dimension Reduction. *Proceedings of the 1993 IEEE International Conference on Neural Networks*, San Francisco, California, USA, 1213-1218.
Kambhatla, N. and T. Leen (1997) Dimension Reduction by Local Principal Component Analysis. *Neural Computation*, **9**, 1493-1516.
KangGeon, K., C. Dongkyu, et al. (2011) Controlling a Humanoid Robot in Home Environment with a Cognitive Architecture. *Proceedings of the 2011 IEEE International Conference on Robotics and Biomimetics*, Phuket Island, Thailand, 1754-1759.
Karaoguz, C., T. Rodemann, et al. (2011) Optimisation of Gaze Movement for Multitasking Using Rewards. *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, California, USA, 1187-1193.
Kawamura, K. and W. N. Browne (2009) Cognitive Robotics. *Encyclopedia of Complexity and System Science*. R. A. Meyers. Heidelberg, Germany, Springer Science, 1109-1126.
Kawamura, K. and S. Gordon (2006) From Intelligent Control to Cognitive Control. *Proceedings of the 11th International Symposim on Robotics and Applications (ISORA)*, Budapest, Hungary, 24-27.
Kawamura, K., S. Gordon, et al. (2008) Implementation of Cognitive Control for a Humanoid Robot. *International Journal of Humanoid Robotics*, **5**, 547-586.
Kawamura, K., R. Peters II, et al. (2004) Multiagent-Based Cognitive Robot Architecture and Its Realization. *International Journal of Humanoid Robotics*, **1**, 65-93.
Kawamura, K., R. A. Peters, et al. (2000) Isac: Foundations in Human-Humanoid Interaction. *IEEE Intelligent Systems and their Applications*, **15**, 38-45.
Keiras, D. E. and D. E. Meyer (1997) An Overview of the Epic Architecture for Cognition and Performance with Application to Human-Computer Interaction. *Human-Computer Interaction*, **12**, 391-438.
Khamassi, M., S. Lallée, et al. (2011) Robot Cognitive Control with a Neurophysiologically Inspired Reinforcement Learning Model. *Frontiers in neurorobotics*, **5**, 1-14.
Khatib, O. (1985) Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, St. Louis, Missouri, 500-505.
Khatib, O., K. Yokoi, et al. (2001) Robots in Human Environments. *ARCHIVES OF CONTROL SCIENCE*, **11**, 123-138.
Kohonen, T. (1982) Self-Organized Formation of Topologically Correct Feature Maps. *Biological cybernetics*, **43**, 59-69.
Kohonen, T. (1990) The Self-Organizing Map. *Proceedings of the IEEE*, **78**, 1464-1480.

Kohonen, T., S. Kaski, et al. (2002) Self Organization of a Massive Document Collection. *IEEE Transactions on Neural Networks*, **11**, 574-585.

Kuffner Jr, J. J. and S. M. LaValle (2000) Rrt-Connect: An Efficient Approach to Single-Query Path Planning. *Proceedings of the the 2000 IEEE Conference on Robotics and Automation*, San Francisco, CA, USA, 995-1001.

Kulic, D., D. Lee, et al. (2008) Incremental Learning of Full Body Motion Primitives for Humanoid Robots. *Proceedings of the Eighth IEEE-RAS International Conference on Humanoid Robots*, Daejeon, Korea, 326-332.

Kulic, D., W. Takano, et al. (2008) Combining Automated on-Line Segmentation and Incremental Clustering for Whole Body Motions. *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, Pasadena, CA, USA, 2591-2598.

Kuniyoshi, Y., M. Inaba, et al. (1994) Learning by Watching: Extracting Reusable Task Knowledge from Visual Observation of Human Performance. *IEEE Transactions on Robotics and Automation*, **10**, 799.

Kuniyoshi, Y., Y. Yorozu, et al. (2003) From Visuo-Motor Self Learning to Early Imitation-a Neural Architecture for Humanoid Learning. *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, Taipei, China, 3132-3139.

Lawrence, N. (2005) Probabilistic Non-Linear Principal Component Analysis with Gaussian Process Latent Variable Models. *The Journal of Machine Learning Research*, **6**, 1816.

Lee, D. and Y. Nakamura (2006) Stochastic Model of Imitating a New Observed Motion Based on the Acquired Motion Primitives. *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, 4994-5000.

Lee, D., C. Ott, et al. (2011) Physical Human Robot Interaction in Imitation Learning. *Proceedings of the 2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, 3439-3440.

Lee, D. and J. Thompson (1982) Vision in Action: The Control of Locomotion. *Analysis of Visual Behavior*. D. J. Ingle, M. A. Goodale and R. J. W. Mansfield. Cambridge, MA, MIT Press, 411–433.

Lehman, J., J. Laird, et al. (1998) A Gentle Introduction to Soar, an Architecture for Human Cognition. *An Invitation to Cognitive Science: Methods, Models, and Conceptual Issues*, **4**, 211-253.

Li, R. and X. Wang (2002) Dimension Reduction of Process Dynamic Trends Using Independent Component Analysis. *Computers & Chemical Engineering*, **26**, 467-473.

Liu, H. and M. Schneider (2012) Similarity Measurement of Moving Object Trajectories. *Proceedings of the Third ACM SIGSPATIAL International Workshop on GeoStreaming*, Redondo Beach, California, USA, 19-22.

Mahadevan, S. and J. Connell (1992) Automatic Programming of Behavior-Based Robots Using Reinforcement Learning. *Artificial Intelligence*, **55**, 311-365.

Malfaz, M., A. Castro-Gonzalez, et al. (2011) A Biologically Inspired Architecture for an Autonomous and Social Robot. *Autonomous Mental Development, IEEE Transactions on*, **3**, 232-246.

Mardia, K. V., J. T. Kent, et al. (1980) Multivariate Analysis. New York, New York, USA, Academic Press.
Marsland, S. (2009) Machine Learning: An Algorithmic Perspective. Chapman & Hall/CRC.
Mataric, M. (2002) Sensory-Motor Primitives as a Basis for Imitation: Linking Perception to Action and Biology to Robotics. *Imitation in Animals and Artifacts*, 391–422.
Mataric, M., M. Williamson, et al. (1998) Behavior-Based Primitives for Articulated Control. *Proceedings of the 1998 International Conference on Simulation of Adaptive Behavior*, Cambridge, Massachusetts, USA, 165–170.
Mataric, M. J. (1992) Integration of Representation into Goal-Driven Behavior-Based Robots. *IEEE Transactions on Robotics and Automation*, **8**, 304-312.
Maye, A. and A. K. Engel (2011) A Discrete Computational Model of Sensorimotor Contingencies for Object Perception and Control of Behavior. *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 3810-3815.
McCarthy, J. (1996) From Here to Human-Level Ai. *Artificial Intelligence*, **171**, 1174-1182.
Microsoft. from http://www.xbox.com/en-US/kinect.
Mirza, N. A., C. L. Nehaniv, et al. (2007) Grounded Sensorimotor Interaction Histories in an Information Theoretic Metric Space for Robot Ontogeny. *Adaptive Behavior*, **15**, 167-187.
Mitchell, T. M. (1997) Machine Learning. *Burr Ridge, IL: McGraw Hill*.
Miyamoto, H. and M. Kawato (1998) A Tennis Serve and Upswing Learning Robot Based on Bi-Directional Theory. *Neural Networks*, **11**, 1331-1344.
Morita, M. (1993) Associative Memory with Nonmonotone Dynamics. *Neural Networks*, **6**, 115-126.
Morita, M. (1994) Smooth Recollection of a Pattern Sequence by Nonmonotone Analog Neural Networks. *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, 1032-1037.
Morita, M. (1996) Memory and Learning of Sequential Patterns by Nonmonotone Neural Networks. *Neural Networks*, **9**, 1477-1489.
Muench, S., J. Kreuziger, et al. (1994) Robot Programming by Demonstration (Rpd)-Using Machine Learning and User Interaction Methods for the Development of Easy and Comfortable Robot Programming Systems. *Proceedings of the 24th International Symposium on Industrial Robots*, Tokyo, Japan, 685-692.
Munoz, P., M. D. R-Moreno, et al. (2011) A Cognitive Architecture and Simulation Environment for the Ptinto Robot. *Proceedings of the 2011 IEEE Fourth International Conference on Space Mission Challenges for Information Technology*, Palo Alto, California, USA, 129-136.
Nabeshima, C., Y. Kuniyoshi, et al. (2006) Adaptive Body Schema for Robotic Tool-Use. *Advanced Robotics*, **20**, 1105-1126.
Nakauchi, Y. and R. Simmons (2002) A Social Robot That Stands in Line. *Autonomous Robots*, **12**, 313-324.

Naksuk, N., C. Lee, et al. (2005) Whole-Body Human-to-Humanoid Motion Transfer. *Proceedings of the 2005 IEEE International Conference on Humanoid Robots*, Tsukuba, Japan, 104-109.

Nehaniv, C. and K. Dautenhahn (2002) The Correspondence Problem. *Imitation in Animals and Artifacts*. Chrystopher L. Nehaniv and K. Dautenhahn, 41-61.

Nehaniv, C. and K. Dautenhahn (2007) Imitation and Social Learning in Robots, Humans and Animals: Behavioural, Social and Communicative Dimensions. Cambridge Univ Press.

Nicolescu, M. and M. Mataric (2003) Natural Methods for Robot Task Learning: Instructive Demonstrations, Generalization and Practice. *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, Melbourne, VIC, Australia 241-248.

Ogawara, K., J. Takamatsu, et al. (2003) Extraction of Essential Interactions through Multiple Observations of Human Demonstrations. *IEEE Transactions on Industrial Electronics*, **50**, 667-675.

Pardowitz, M., S. Knoop, et al. (2007) Incremental Learning of Tasks from User Demonstrations, Past Experiences, and Vocal Comments. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, **37**, 322-332.

Peters II, R. A., K. Hambuchen, et al. (2001) The Sensory Ego-Sphere as a Short-Term Memory for Humanoids. *Proceedings of the 2nd IEEE-RAS International Conference on Humanoid Robots*, Tokyo, Japan, 22-24.

Pfeifer, R., J. Bongard, et al. (2007) How the Body Shapes the Way We Think : A New View of Intelligence. Cambridge, Mass., MIT Press.

Pook, P. and D. Ballard (1993) Recognizing Teleoperated Manipulations. *Proceedings of the 1993 IEEE International Conference and Robotics and Automation*, Atlanta, GA, USA, 578-585.

Purdy, J. and K. Olmstead (1984) New Estimate for Storage Time in Sensory Memory. *Perceptual and motor skills*, **59**, 683-686.

Rabiner, L. R. (1989) A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, **77**, 257-286.

Ragland, J., J. Yoon, et al. (2007) Neuroimaging of Cognitive Disability in Schizophrenia: Search for a Pathophysiological Mechanism. *International Review of Psychiatry*, **19**, 417-427.

Rahimi, M. and W. Karwowski (1992) Human-Robot Interaction. Taylor & Francis, Inc. Bristol, PA, USA.

Rahman, S. and H. Xu (2004) A Univariate Dimension-Reduction Method for Multi-Dimensional Integration in Stochastic Mechanics. *Probabilistic Engineering Mechanics*, **19**, 393-408.

Roweis, S. and L. Saul (2000) Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, **290**, 2323-2326.

Russell, S. J. and P. Norvig (2003) Artificial Intelligence : A Modern Approach. Upper Saddle River, N.J., Prentice Hall/Pearson Education.

Rybski, P. and R. Voyles (1999) Interactive Task Training of a Mobile Robot through Human Gesture Recognition. *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, Detroit, Michigan, USA, 664-669.

Salem, M., K. Rohlfing, et al. (2011) A Friendly Gesture: Investigating the Effect of Multimodal Robot Behavior in Human-Robot Interaction. *Proceedings of the 20th IEEE International Symposium on Robot and Human Interactive Communication*, Atlanta, Georgia, USA, 247-252.

Sammon, J., JW (1969) A Nonlinear Mapping for Data Structure Analysis. *IEEE Transactions on Computers*, **100**, 401-409.

Sánchez Boza, A., R. H. Guerra, et al. (2011) Artificial Cognitive Control System Based on the Shared Circuits Model of Sociocognitive Capacities. A First Approach. *Engineering Applications of Artificial Intelligence*, **24**, 209-219.

Šarić, M., C. H. Ek, et al. (2011) Dimensionality Reduction Via Euclidean Distance Embeddings.

Sauser, E. L., B. D. Argall, et al. (2011) Iterative Learning of Grasp Adaptation through Human Corrections. *Robotics and Autonomous Systems*, **60**, 55-71.

Schaal, S. (1999) Is Imitation Learning the Route to Humanoid Robots. *Trends in Cognitive Sciences*, **3**, 233-242.

Schaal, S., A. Ijspeert, et al. (2003) Computational Approaches to Motor Learning by Imitation. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, **358**, 537.

Schneider, W. (1999) Working Memory in a Multilevel Hybrid Connectionist Control Architecture (Cap2). *Models of Working Memory: Mechanisms of Active Maintenance and Executive Control*. A.Miyake and P. Shah. New York, NY, USA, Cambridge University Press, 340–374.

Schölkopf, B., A. Smola, et al. (1998) Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, **10**, 1299-1319.

Shon, A., K. Grochow, et al. (2005) Robotic Imitation from Human Motion Capture Using Gaussian Processes. *Proceedings of the Fifth IEEE-RAS International Conference on Humanoid Robots*, Tsukuba, Japan, 129-134.

Shrobe, H. and P. Wilson (2006) Chip: A Cognitive Architecture for Comprehensive Human Intelligence and Performance. *Electronic Resource: http://www.darpa.mil/ipto/programs/bica/phase1.htm*.

Shukla, A. and A. Billard (2012) Coupled Dynamical System Based Arm–Hand Grasping Model for Learning Fast Adaptation Strategies. *Robotics and Autonomous Systems*, **60**, 424-440.

Simmons, R. and D. Apfelbaum (1998) A Task Description Language for Robot Control. *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Victoria, Canada, 1931-1937.

Skubic, M. and R. Volz (2000) Acquiring Robust, Force-Based Assembly Skills from Human Demonstration. *IEEE transactions on robotics and automation*, **16**, 772-781.

Sloman, A. (2009) Some Requirements for Human-Like Robots: Why the Recent over-Emphasis on Embodiment Has Held up Progress. *Creating Brain-Like Intelligence*. B. Sendhoff, 248-277.

Sloman, A., J. Wyatt, et al. (2006) Long Term Requirements for Cognitive Robotics. *Proceedings of the AAAI 2006 Workshop on Cognitive Robotics*, Boston, Massachusetts, USA, 143–150.

Steil, J., F. Röthling, et al. (2004) Situated Robot Learning for Multi-Modal Instruction and Imitation of Grasping. *Robotics and Autonomous Systems*, **47**, 129-141.

Sumtak. from http://www.sumtak.com/en/products/rotary-encoder/.
Sun, R. (2003) A Tutorial on Clarion. Technical Report. Cognitive Science Department, Rensselaer Polytechnic Institute. **15**, 2003.
Sutton, R. and A. Barto (1998) Reinforcement Learning: An Introduction. The MIT press.
Tan, H. (2011) A Framework for Skill Learning for Cognitive Robots through Semantic Teaching and Imitation Using a Cognitive Architecture. *Journal of Robotics and Autonomous Systems*.
Tan, H. (2012) Imitation Learning and Behavior Generation in a Robot Team. *Proceedings of the 11th International Symposium on Distributed Autonomous Robotics Systems*, Baltimore, Maryland.
Tan, H. (2012) Implementation of a Framework for Imitation Learning on a Humanoid Robot Using a Cognitive Architecture. *The Future of Humanoid Robots: Research and Applications*. R. Zaier, InTech, 189-210.
Tan, H., Q. Du, et al. (2012) Robots Learn Writing. *Journal of Robotics*, **2012**, 15.
Tan, H., E. Erdemir, et al. (2011) A Potential Field Method-Based Extension of the Dynamic Movement Primitive Algorithm for Imitation Learning with Obstacle Avoidance. *Proceedings of the 2011 IEEE International Conference on Mechatronics and Automation*, Beijing, China, 525-530.
Tan, H. and K. Kawamura (2011) A Computational Framework for Integrating Robotic Exploration and Human Demonstration in Imitation Learning. *Proceedings of the the 2011 IEEE International Conference on System, Man and Cybernetics*, Anchorage, AK, USA, 2501-2506.
Tan, H. and C. Liang (2011) A Conceptual Cognitive Architecture for Robots to Learn Behaviors from Demonstrations in Robotic Aid Area. *Proceedings of the 33rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society* Boston, MA, USA, 1248-1262.
Tani, J. (2003) Learning to Generate Articulated Behavior through the Bottom-up and the Top-Down Interaction Processes. *Neural Networks*, **16**, 11-23.
Tani, J., M. Ito, et al. (2004) Self-Organization of Distributedly Represented Multiple Behavior Schemata in a Mirror System: Reviews of Robot Experiments Using Rnnpb. *Neural Networks*, **17**, 1273-1289.
Tenenbaum, J., V. Silva, et al. (2000) A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, **290**, 2319-2323.
Tenorth, M. and M. Beetz (2009) Knowrob—Knowledge Processing for Autonomous Personal Robots. *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, St.Louis Missouri, 4261-4266.
Theodorou, E., J. Buchli, et al. (2010) A Generalized Path Integral Control Approach to Reinforcement Learning. *The Journal of Machine Learning Research*, **11**, 3137-3181.
Theodorou, E., J. Buchli, et al. (2010) Reinforcement Learning of Motor Skills in High Dimensions: A Path Integral Approach. *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*, Anchorage, Alaska, USA, 2397-2403.
Thornton, S. R. (2009) Real-Time Gesture Imitation in a Soft-Arm Control Robot. EECS Department. Nashville, Vanderbilt University. **Master**.
Tibshirani, R. (1992) Principal Curves Revisited. *Statistics and Computing*, **2**, 183-190.
Tipping, M. and C. Bishop (1999) Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **61**, 611-622.

Trafton, J., A. Schultz, et al. (2005) Perspective-Taking with Robots: Experiments and Models. *Proceedings of the 2005 IEEE International Workshop on Robot and Human Interactive Communication*, Nashville, Tennessee, USA, 580-584.

Uchiyama, M. (1978) Formation of High Speed Motion Pattern of Mechanical Arm by Trial. *Transactions, Society of Instrument and Control Engineers*, **19**, 706-712.

Ude, A., C. Atkeson, et al. (2004) Programming Full-Body Movements for Humanoid Robots by Observation. *Robotics and Autonomous Systems*, **47**, 93-108.

Vakanski, A., I. Mantegh, et al. (2012) Trajectory Learning for Robot Programming by Demonstration Using Hidden Markov Model and Dynamic Time Warping. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, **42**, 1039-1052.

Verbeek, J., N. Vlassis, et al. (2002) Coordinating Principal Component Analyzers. *Artificial Neural Networks—ICANN 2002*, Springer Berlin Heidelberg, 914-919.

Vijayakumar, S. and S. Schaal (2000) Locally Weighted Projection Regression: An O (N) Algorithm for Incremental Real Time Learning in High Dimensional Space. *Proceedings of the 17th International Conference on Machine Learning*, Stanford, CA, USA, 288–293.

Voyles, R. and P. Khosla (2001) A Multi-Agent System for Programming Robots by Human Demonstration. *Integrated Computer-Aided Engineering*, **8**, 59-67.

Wang, J., D. Fleet, et al. (2008) Gaussian Process Dynamical Models for Human Motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **30**, 283-298.

Williams, C. (2002) On a Connection between Kernel Pca and Metric Multidimensional Scaling. *Machine Learning*, **46**, 11-19.

Wilson, A. D. and A. F. Bobick (1999) Parametric Hidden Markov Models for Gesture Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **21**, 884-900.

Winikoff, M. (2005) Jack™ Intelligent Agents: An Industrial Strength Platform. *Multi-Agent Programming*. Bordini, Kluwer, 175-193.

Wood, F., K. Esbensen, et al. (1987) Principal Component Analysis. *Chemometr. Intel. Lab. Syst*, **2**, 37–52.

Yang, J., Y. Xu, et al. (1997) Human Action Learning Via Hidden Markov Model. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, **27**, 34-44.

Yang, T., J. Zhang, et al. (2006) Motion Planning and Error Analysis in Robot Assistant Micro-Surgery System. *Proceedings of the the 6th World Congress on Intelligent Control and Automation*, Dalian, Liaoning, China, 8819-8823.

Yeasin, M. and S. Chaudhuri (2000) Toward Automatic Robot Programming: Learning Human Skill Fromvisual Data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, **30**, 180-185.