NOVEL METHODS FOR HUMAN-ROBOT SHARED CONTROL IN COLLABORATIVE ROBOTICS

A Dissertation

by

ZONGYAO JIN

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2020

Major Subject: Mechanical Engineering

ABSTRACT

Blended shared control is a method to continuously combine control inputs from traditional automatic control systems and human operators for control of machines. An automatic control system generates control input based on feedback of measured signals, whereas a human operator generates control input based on experience, task knowledge, and awareness and sensing of the environment in which the machine is operating. Such active blending of inputs from the automatic control agent and the human agent to jointly control machines is expected to provide benefits in terms of utilizing the unique features of both agents, i.e., better task execution performance of automatic control systems based on sensed signals and maintaining situation awareness by having the human in the loop to handle safety concerns and environmental uncertainties. The shared control approach in this sense provides an alternative to full autonomy. Many existing and future applications of such an approach include automobiles, underwater vehicles, ships, airplanes, construction machines, space manipulators, surgery robots, and power wheelchairs, where machines are still mostly operated by human operators for safety concerns. Developing machines for full autonomy requires not only advances in machines but also the ability to sense the environment by placing sensors in it; the latter could be a very difficult task for many such applications due to perceived uncertainties and changing conditions. The notion of blended shared control, as a more practical alternative to full autonomy, enables keeping the human operator in the loop to initiate machine actions with real-time intelligent assistance provided by automatic control.

The problem of how to blend the two inputs and development of associated scientific tools to formalize and achieve blended shared control is the focus of this work. Specifically, the following essential aspects are investigated and studied. *Task learning:* modeling of a human-operated robotic task from demonstration into subgoals such that execution patterns are captured in a simple manner and provide reference for human intent prediction and automatic control generation. *Intent prediction:* prediction of human operator's intent in the framework of subgoal models such that it encodes the probability of a human operator seeking a particular subgoal. *Input blending:* generating automatic control input and dynamically combining it with human operator's input based on prediction probability; this will also account for situations where the human operator may take unexpected actions to avoid danger by yielding full control authority to the human operator. *Subgoal adjustment:* adjusting the learned, nominal task model dynamically to adapt to task changes, such as changes to target object, which will cause the nominal model learned from demonstration to lose its effectiveness. This dissertation formalizes these notions and develops novel tools and algorithms for enabling blended shared control. To evaluate the developed scientific tools and algorithms, a scaled hydraulic excavator for a typical trenching and truck-loading task is employed as a specific example. Experimental results are provided to corroborate the tools and methods.

To expand the developed methods and further explore shared control with different applications, this dissertation also studied the collaborative operation of robot manipulators. Specifically, various operational interfaces are systematically designed, a hybrid force-motion controller is integrated with shared control in a mixed world-robot frame to facilitate human-robot collaboration, and a method that utilizes vision-based feedback to predict the human operator's intent and provides shared control assistance is proposed. These methods provide ways for human operators to remotely control robotic manipulators effectively while receiving assistance by intelligent shared control in different applications. Several robotic manipulation experiments were conducted to corroborate the expanded shared control methods by utilizing different industrial robots.

DEDICATION

To my wife and parents!

ACKNOWLEDGMENTS

Doing this Ph.D. would be impossible for me without the support of so many awesome people. I would like to express my immense gratitude here. I am very grateful for being able to work with my advisor Dr. Pagilla on some very interesting problems. He taught me how to approach, formulate, and solve research problems. He has also been providing excellent academic guidance and lab resources more than I ever expected. I am also very grateful for the support of my family and friends. They made me understood that there is much more than research and work in this life. Thank you very much!

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This dissertation was supported by a committee consisting of Professors Prabhakar Pagilla, Won-Jong Kim, Sivakumar Rathinam of the Department of Mechanical Engineering, and Professor Theodora Chaspari of the Department of Computer Science & Engineering.

This research was conducted as part of a collaborative effort, with complementary research conducted by Dr. Girish Chowdhary and Harshal Maske of the Department of Agricultural and Biological Engineering, University of Illinois Urbana-Champaign, and Dr. Charles Abramson and Emily Kieson of the Department of Psychology, Oklahoma State University. A previous master student Mitchell A. Allain contributed to the customization of part of the excavator hardware platform. Their work and the guidance from my advisor Dr. Pagilla have led my research in the right direction.

All other work conducted for the dissertation was completed by the student independently.

Funding Sources

Graduate study was generously supported by the Mechanical Engineering Department of Texas A&M University in forms of Teaching Assistantship, Jame J. Cain Student Awards, Fellowship, and by my professor Dr. Pagilla in the form of Research Assistantship. The research project was funded by the US NSF NRI grant 1527828.

NOMENCLATURE

DOF	Degrees of Freedom
SC	Shared Control
BSC	Blended Shared Control
LfD	Learning from Demonstration
IRL	Inverse Reinforcement Learning
OPbS	Operator Primitives based Segmentation
MCV	Motion Command Variance
BNPC	Bayesian Non-Parametric Clustering
CRP	Chinese Restaurant Process
BNPC/TO	Bayesian Non-Parametric Clustering w/ Temporal Ordering
ESTM	Empirical Stochastic Transition Matrix
DADE	Dynamic Angle Difference Exponential
HSTF	Hyperbolic Slope Transition Function
AEHR	Adjustment Encoding Hyper-Rectangle
SWAI	Skill Weighted Action Integral
BSC/CA	Blended Shared Control with Conflict Awareness

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 General Concepts

Collaboration and cooperation among people in complex situations as a way to combine unique features and strengths from multiple agents and completing tasks which otherwise are not possible has been a key aspect for the development of human society [1, 2, 3]. Automatic control, traditionally considered as a powerful tool instead of a collaborator, is known to be able to execute well-defined and repetitive tasks with high precision and efficiency. Human operators, on the other hand, exhibit strong situational awareness for handling uncertainties in the environment and for quickly adapting to changes during task execution. With rapid advances in the development of scientific tools in robotics, artificial intelligence, machine learning, and automatic control, researchers have reconsidered the role of automatic control as an active collaborator in many applications [4, 5, 6]. Because of the potential for achieving a higher level of safety and performance, there has been a strong interest in the development of tools that address the challenges of collaboration between humans and robots [7, 8, 9]. Human-machine collaboration in the form of shared control has been explored in many studies in robotics and human-machine interaction with a focus on assisting human operators with intelligent robotics and control technologies during task execution. Results have shown that shared control architectures offer more situational awareness and robustness over full autonomy, and could provide additional benefits in terms of performance improvement, difficulty reduction, capability extension, and safety enhancement than human manual control alone [10, 11].

1.2 Classification and Applications of Shared Control

Shared control consists of methods which directly or indirectly combine human operator input with automatic control input for the control of dynamic systems [12], i.e., the actions of the human operator are closely integrated into the closed-loop control system. While collaborative robotics for complex applications sometimes consist of many shared control methods, the methods from existing literature can be generally classified into the following distinguishable forms, each of which has distinct characteristics and target applications: collaborative control, traded control, indirect shared control, coordinated control, virtual constraint control, blended shared control.

The simplest forms of shared control are collaborative control and traded control. In collaborative control, a certain subset of control inputs is handled by a human operator, while the rest of the inputs are processed by the machine. This is commonly observed in automotive applications, where for example, in the cruising mode, steering action is handled by a human and throttle control is achieved by the closed-loop controller. With traded control, the control authority can be transferred completely to either the operator or the automatic control algorithms. This has been explored and applied to aircraft autopilot systems where the control authority is given to the computer while cruising, whereas more complex situations such as taking-off and landing are handled by human operators. In such applications, switching of the control authority is decided by the human operators. Recent studies [13, 14] have suggested ways to adaptively change control authority between the human and the robot based on the confidence of the human operator's intent.

Indirect shared control, typically in the form of providing sensory information to human operators, has also been widely used. Instead of providing control input directly to the robot, the system offers feedback to the human operators to make adjustments or improvements. In [15], providing operational instructions to operators through graphical user interfaces has resulted in improved execution performance for excavator earth-moving tasks. Haptic feedback to human operators has been used for quadrotor collision avoidance [16], vehicle blind spot warning [17], and highway merging scenarios [18].

When controlling a multiple DOF robot, difficulties arise for the human operators when the inverse kinematics is not intuitive or when there are too many DOF for the human to control simultaneously. Coordinated control reduces such difficulty by handling the kinematics and having humans to give commands in a lower dimensional space which is typically more intuitive. For example, (1) in shared control of drones, human operators provide commands in the Cartesian space and the control algorithm coordinates the four rotors to achieve the desired motion [19], and

(2) in the control of robotic manipulators, human operators often specify the motion of the end-effector while the coordination of the individual joint actuators is handled in software via robot kinematics [20].

In virtual constraint control methods, assistance is provided to human operators by disabling input commands leading to hazardous results. In [21], a large-scale manipulator with forwarder machine is designed with shared control capabilities that allows a human operator to give inputs in the Cartesian space. While executing the operator's command, the underlying software provides a protection mechanism such that the commands resulting in singularities or exceeding physical limits are ignored. Researchers working on the development of smart wheelchairs have also employed virtual constraint control to protect such systems from colliding with their surroundings, where inputs that may result in collisions are voided or handled by providing force feedback to the user [22, 23, 24].

1.3 Blended Shared Control

Blended shared control is yet another form of shared control, where the human operator and automatic control inputs are combined continuously to control a robot to collectively perform a task. The BSC scheme is more seamless in the sense that the human operator always has access to the control of the robot and that the actions are always initiated by the human operator; the automatic control, by predicting the operator's intent, keeps providing assistance in the form of closed-loop control input and is dynamically combined with the human input in real-time to jointly act on the robot. The BSC scheme is particularly relevant for applications where the environments are dynamic, unstructured, and uncertain. In such environments, it is difficult to efficiently execute tasks using full autonomy since comprehensive sensing of the environment could be impractical or prohibitively expensive. Furthermore, when safety is of paramount importance, the quick decision-making abilities and situational awareness of human operators provide a level of robustness that is difficult to encode in mathematical models and automatic algorithms. However, by providing some level of control assistance via BSC to the human operator, one can reduce the difficulty of operating robots in such environments, improve task performance, and enhance safety during task execution.

Studies on blended shared control have shown promising results towards applying such techniques to surgery robots, smart wheelchairs, construction excavators, etc. In [25, 26, 27], researchers applied BSC methods on wheelchair navigation problems for collision avoidance where the human and automatic control inputs were blended to drive the robots. The blending parameters in these studies were designed by optimizing various cost functions relating human input to obstacle collision or the probability distribution of velocity versus collision. In robotic surgery applications, researchers have demonstrated that by utilizing BSC on some surgical tasks, it is possible to reduce the probability of tissue crush injury [28, 29] caused by over-application of grasping force. In selecting the blending parameter, in one case, researchers related the parameter with a model dynamically predicting tissue type, in another case, the blending parameter was assigned to fixed values based on experiments.

1.4 Motivation and Problem Formulation

The motivation for studying BSC arises from the fact that many robotic tasks in dynamic environments with uncertainties are still executed by human operators. There is a recognition, which is is confirmed by several preliminary studies, that it is possible to provide a reasonable level of automated and intelligent assistance to human operators through the application of BSC to improve performance and reduce operational difficulty without compromising safety.

In this work, we first employ the operation of a hydraulic excavator for a trenching and truck-loading task in the heavy construction industry as a particular example for providing a simple formulation of BSC and the development of associated tools; this will be followed by more general formulations of the problem and associated tools and solutions. In the heavy construction industry, there are significant risks associated with operating heavy machinery in uncertain task environments without human involvement; thus, the consorted industry has always relied on human operators for controlling the excavators. However, operating machines such as hydraulic excavators efficiently requires a certain level of skill and experience. An operator of a simple excavator typically controls the four excavator degrees of freedom independently and simultaneously by two joysticks with four motion axes as shown in Fig. 1.1. It is a representative case of human-operated robots since similar joystick-based operational interfaces can be employed for the control of many other robots.

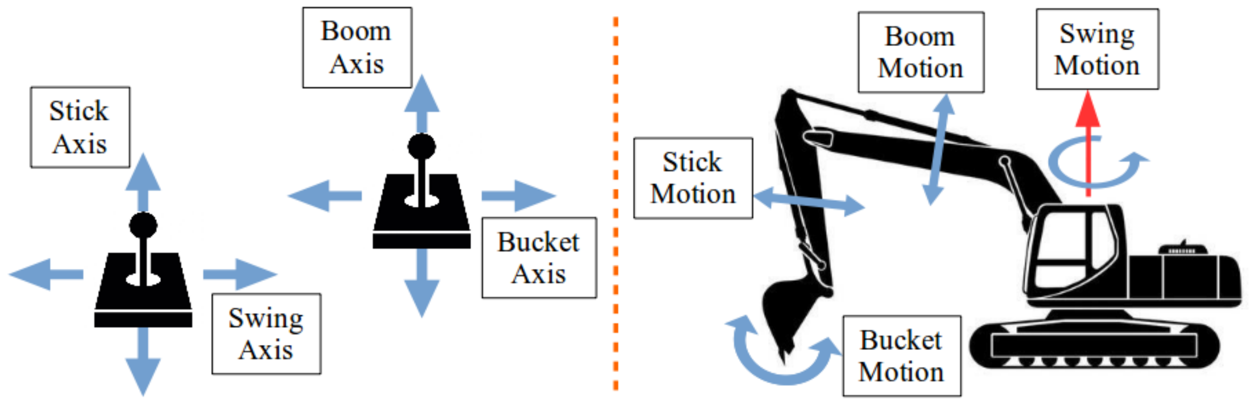


Figure 1.1: Excavator Operation Interface via Joysticks

The growth of the construction industry in recent decades has resulted in the shortage of skilled human operators, and the use of BSC scheme has the potential to overcome the skill-gap issue by providing intelligent assistance to novice operators to improve their task execution performance. Studies in [30, 31, 15, 32, 33] investigated some of these BSC solutions coupled with the concepts of task learning and intent prediction, and some promising experimental results were presented. In addition to the difficulty of operating robots with uncertainties in the task environment, another feature of excavator earth-moving, which is also representative in many other robotic applications, is that for completing a task, operations typically have similar patterns. They are often repeated over many cycles, but lack well-organized repetition. Thus, for providing intelligent assistance through BSC based on these features, one first needs to find an effective model to encode the task. Such model should not only provide references for autonomous task execution, but also be used to infer human operator's intent. The concept of subgoals as discussed in [34, 31, 15, 32] has been used for such interpretation in BSC, where specific tasks are learned through demonstration and divided into subgoals. Then, by predicting the human operator's intent for seeking a particular subgoal, automatic control is generated and blended to human operator's input for shared action. To investigate further and develop tools for BSC, we formulate the following key questions:

1. How does one learn the task and segment it into well-defined subgoals?

2. How does one ascertain and quantify the intent of the operator in terms of moving from one subgoal to another?

3. How does one update the blending parameter (the parameter that decides the extent of sharing between the human operator input and automatic control input in the total command input) in real-time to facilitate operator intent and subgoal achievement?

4. How does one adjust nominal subgoals based on changes in target objects during operation as detected by the human operator?

In this dissertation, novel strategies developed to answer these questions are described. Evaluation of the effectiveness of the proposed strategies is provided via experimental results on a scaled hydraulic excavator platform performing a typical trenching-and-truck-loading task where the human operator commands the robot with two joysticks as indicated in Fig. 1.1.

We have also extended the aforementioned strategies and methods to human-machine shared control in the operation of robotic manipulators for object inspection/handling and surface tracking applications where unknown geometry and uncertain environment present challenges for full autonomy. In this regard, we designed intuitive interfaces which can reduce the difficulty of operating a multiple DOF robotic manipulator, and integrated cameras into robotic end-effectors for facilitating sharing of the control authority between the human operator and a vision-based automatic control based on human intent prediction. To corroborate the effectiveness of the shared control methods developed for robotic manipulators, a number of experiments were conducted on two additional robotic platforms. One physical platform consists of a Universal Robot (UR5, 6-DOF), and a general-purpose joystick. A human operator is invited to execute a surface tracking task and a object handling task. The other physical platform consists of an ABB robot (IRB-4600, 6-DOF), a custom-designed end-effector with cameras, and a general-purpose game controller. Another human operator is invited to execute an object inspection and handling task by remotely controller the robot with the assistance of intent prediction and vision-based automatic control assistance.

1.5 Dissertation Outline

Blended shared control studies for collaborative robotic tasks described earlier involve many different aspects, which utilize ideas from various fields in order to formulate a framework for developing and synthesizing solutions. As a result, each of the following chapters starts with a specific literature review in the relevant fields, and then presents the development of the proposed technical tools and algorithms. The rest of this dissertation is organized as follows. From Chapter 2 to 6, we focus on developing and corroborating blended shared control methods where the excavator trenching-and-loading task is used as a specific example to motivate the developments which can be adapted to other robotic tasks. Specifically, Chapter 2 presents two methods for learning from demonstration with a focus on shared control tasks. Chapter 3 provides tools for predicting the intent of the human operator based on subgoals as the result of the learning algorithms. Input blending with intent and conflict awareness, as a method for combining different inputs and relating operator's input magnitude with the automatic controller gain, is provided in Chapter 4. During dynamic task execution, one needs to consider adjusting the nominal task model learned from the initial demonstration since task execution may result in changes in the environment or target object as perceived by the human operator. Chapter 5 is thus provided to address the problem of subgoal adjustment. The excavator hardware platform, experiment design and results are presented in Chapter 6. Chapter 7 presents the development of shared control methods for the collaborative operation of robotic manipulators, where two approaches each with a different focus are developed. Experimental results to corroborate the effectiveness of the collaborative operation methods of robotic manipulators are also provided in Chapter 7. A summary of this dissertation and topics for future work are given in Chapter 8.

2. TASK LEARNING

Learning from an operator's demonstration is the first key element in the BSC framework. Here we are looking for specific learning models which are not merely a description of a given task for automated execution. The model has to provide a mechanism for predicting the operator's intent for BSC applications where the human is actively operating the robot whereas automatic control provides assistance based on intent prediction. A survey of literature on learning from demonstration and task characteristics of SC applications are presented in Sec. 2.1. Based on this survey, subgoal based task model happens to be an appropriate model and is considered in this work. This model encodes a complicated task into a number of subtasks, each associated with a subgoal indicating the completion of a subtask. By completing subtasks in a specific order, the overall task is accomplished. The learning of subgoals involves capturing distinguishable patterns in operator's command input which will lead to subgoal distributions. Section 2.2 provides two methods developed for extracting the distributions of subgoals based on demonstration data and quantification of operator's behavior. In particular, one method was developed earlier based on a threshold mechanism. Another method was proposed with a unified metric quantifying operator's behavior to overcome situations where the threshold method is difficult to apply. Based on the extracted distributions of subgoals, Section 2.3 provides a method to identify the subgoals and their execution sequence which will lead to task completion that can be used for BSC.

2.1 Task Characteristics and Subgoal Learning

The majority of Learning from Demonstration (LfD) methods are developed for full autonomy as provided in several research surveys [35, 36]. Specifically, Inverse Reinforcement Learning (IRL) is used to learn reward functions describing desired task execution behaviors and some subsequent optimal policy which will lead to autonomous task completion [37, 38, 39, 40, 41, 42, 43]. These methods have been successfully applied to many autonomous robotics tasks. However, they have some drawbacks when dealing with SC applications. First, they can be computationally in-efficient when dealing with robots with continuous states and actions. Second, the policy model provides little guidance for predicting the operator's intent when the human is actively participating in task execution. When humans are executing a task, they typically do not associate each action taken with some mathematically constructed rewards. Subgoal-based IRL methods were also developed further such that the learning results are in the form of a set of subgoals [44, 45, 34]. These methods are also computationally expensive due to reward calculation through iterative calculation especially when dealing with robots that have many DOF, since calculation increases exponentially with the number of robot DOF. However, the concept of subgoals inspired the development of focused LfD methods for SC applications, and further assumptions are made based on human behaviors to facilitate the learning and intent prediction formulation [15, 32]. Humans are generally not good at multi-tasking and have the tendency of decomposing a given complicated task into a sequence of subtasks; by completing one subtask and achieving its subgoal at a time, the overall task is completed [46, 47, 48]. When generalized to robot operations, it can be interpreted as having a similar command pattern in the process of finishing the subtask. Once the subgoal is achieved, the operator then switches to the next subtask typically with a considerably different command pattern. These subgoals can be reciprocally used as references for intent prediction in that if an operator command is detected while knowing the robot states, it is possible to forecast which subtask the operator is executing and which subgoal the operator is seeking [31, 32, 33].

Subgoals, in the context of human operated robot tasks, can be points in the robot joint space which correspond to different robot configurations, or points in the Euclidean work space corresponding to different end-effector positions or robot body locations. To illustrate the concept of subgoals, consider the excavator shown in Fig. 2.1 performing a trenching-and-loading task. Seven nominal subgoals, visualized by wire-frame, are shown; by repeating such subgoals in a correct sequence, the operator can complete the task. Each subgoal in this case is a point in the joint space, but can be instantiated by a robot configuration via forward kinematics, as 1, 4, and 7 are shown on the right. Note that transitioning between subgoals is executable by either human operators (easy for skilled operators, yet may still be hard for novice operators) or some straight forward closed-loop control algorithms.

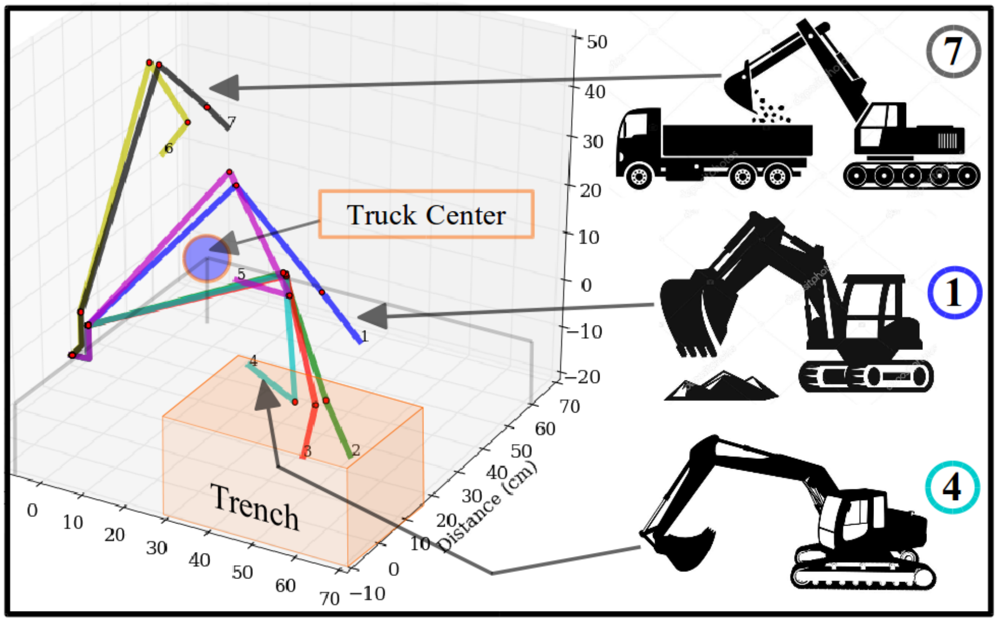


Figure 2.1: Illustration of Subgoals in Joint Space

With the inspiration of subgoals and assumptions on human behavior, the concepts of operator and action primitives were proposed in the SC-focused LfD methods to capture from demonstration significant changes in operator's command patterns, indicating completion of one subtask and start of another subtask. Thus, these command pattern changes can be employed to obtain distributions of subgoals. The primitive-based methods essentially set thresholds for the operator's input to each robot degree of freedom to classify input values into a number of broad categories. When an input crosses a given threshold, i.e., the general command changes its pattern at that time instance, the associated robot states and the corresponding configuration are recorded as a potential subgoal. Statistical inference tools are then employed to cluster the distributions as mixture models and identify subgoals. These methods are more suitable for SC applications in that (1) the learned model also provides reference for predicting the intent of the human operator and (2) they are computationally more efficient since reasonable assumptions on human behavior are made to simplify the learning process. However, they also have some limitations. Threshold methods are generally sensitive to noise which could cause the measured values to cross the threshold. Furthermore, the operator's input pattern in some applications can be quite complex such that it is hard to decide on how many thresholds will lead to meaningful categorizations.

2.2 Extracting Distributions of Subgoals

2.2.1 Operator Primitives Segmentation

To discover from demonstration data the distributions of points which potentially encode the subgoal-changing behavior, the concept of operator primitives is employed. It is a threshold mechanism and is similar to that of the action primitives [49, 15] in the sense that they both map the operator's joystick input into broader categories as depicted in Fig. 2.2. This is a reasonable generalization for human operated heavy construction machines since they typically have hydraulic actuation with slow dynamics, and input resolution is usually not critical. The operators tend to give commands in a broader sense, e.g., when we set $N = 3$ in Fig. 2.2, the categories could be positive input, negative input, and zero or near-zero perturbation, indicating forward, backward, and stop motions. The variable N hence controls the resolution of such a map.

With such generalization and considering that most construction machines are operated with joysticks to command the velocity of individual joints, we formulate our definition of operator primitives in the following manner. Let o_j be the operator primitive variable for joint j, then at any time, o_j can take N values given by $w \in \{1, 2, \cdots, N\}$ which encodes the categories. Thus, the primitive-categorized states of an m-DOF robot can be mathematically captured by the concept of operator primitives; at any given time, the operative primitives vector is given by $\mathbf{o} =$

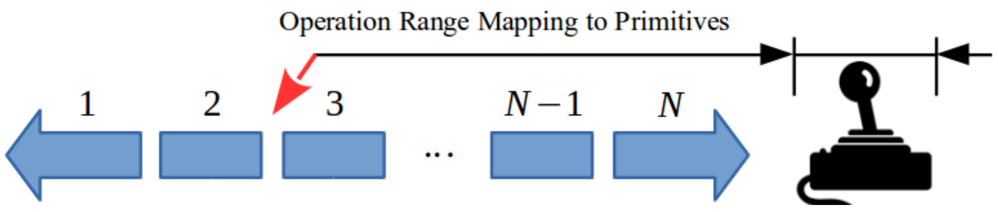


Figure 2.2: Mapping Between Joystick Input and Primitives

$[o_1, o_2, \cdots, o_m]$. In our case, we set $N = 3$ and use the K-Means algorithm [50] to cluster the data into three categories (forward, backward, near-zero). We initialize the cluster means $\mu_{jw}, w \in \{1, 2, 3\}$ for joint j by

$$\begin{cases} \mu_{j1} = \max_t x_j(t) \\ \mu_{j2} = \frac{1}{2}\{\max_t x_j(t) - \min_t x_j(t)\} \\ \mu_{j3} = \min_t x_j(t) \end{cases} \tag{2.1}$$

where $x_j(t)$ is the state of joint j at time t. We update the cluster assignment and the means by Lloyd's algorithm [51], i.e., finding the index w such that

$$\operatorname*{argmin}_w ||x_j(t) - u_{jw}||_2. \tag{2.2}$$

and updating u_{jw} with the new cluster assignment by

$$\mu_{jw} = \frac{1}{\eta} \sum_t \{x_j(t) \mathbb{1}(w)\} \tag{2.3}$$

where $\mathbb{1}(w)$ is the indicator function taking the value of 1 if $x_j(t)$ gets assigned to cluster w and 0 otherwise, and η is the total number of elements assigned to cluster w. We iterate on such process until the means converge. The rest of the segmentation process is carried out in a similar manner as described in details in [49, 15], i.e., we extract the data points where the the value of operator primitives vector $\mathbf{o}$ is different from its previous value, indicating a change of operation patter. The extracted data points, therefore, encode the potential distributions of subgoals. The implementation of such encoding is provided in Algorithm 1.

The difference in formulation between the operator primitives and the action primitives as in [49, 15] is that the action primitives are defined based on the robot joint velocities and operator primitives are based on the operator's joystick input. Hence, the operator primitives capture the operator's inputs directly from the joystick space. A significant advantage of the operator primitives in heavy construction setting is that they are not affected by drastic changes in workload or

Algorithm 1 Extraction of Subgoal Distributions via Operator Primitives Segmentation

```
Input: Robot states from demonstration
Output: Extracted subgoal distributions
 1: for each robot joint do
 2:     Take velocity states as training set and apply K-means clustering algorithm
 3:     Assign labels for robot states at each sample time
 4: end for
 5: Augment robot states with operator primitive vector o consisting of cluster labels
 6: for robot states augmented with operator primitive vectors do
 7:     if $o(t) \neq o(t+1)$ then
 8:         Encode robot position states at time $t$ and $t+1$ into the distributions of subgoals
 9:     end if
10: end for
```

contact force. For example, during a digging process, the operator may not change command input combination, hence the joint velocities ideally should not change drastically. However, the workload or contact force may change dramatically before and after the end-effector touches a hard surface, which will likely cause changes in robot joint velocities, thus, affecting the action primitives. Therefore, operator primitives provide better encoding of operator's behaviors and are less sensitive to perturbations due to the environment. One can observe this from the experimental data shown in Fig. 2.3, where the regions highlighted with yellow circles are the situations where the joint velocity for the bucket was significantly affected by hard contact during the digging process. But the operator's intent, as reflected from the joystick input, does not change.

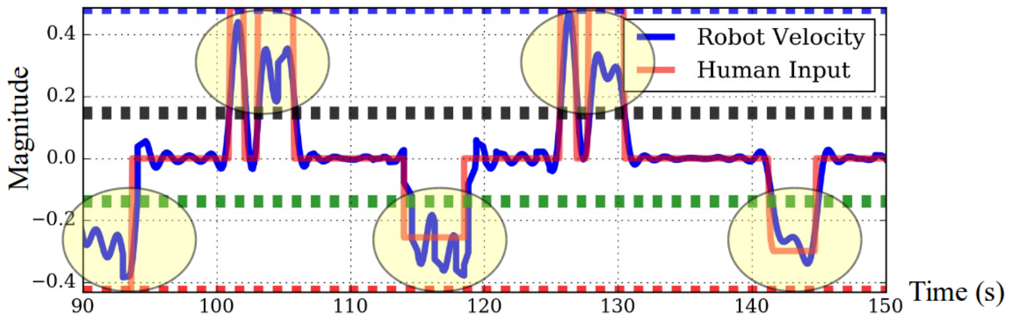


Figure 2.3: Operator Input overlapped with Joint Velocity of Excavator Bucket

2.2.2 Limitations of Threshold Method and Motivation for a Unified Metric

For illustrating the problem of threshold-crossing caused by noise and other signal related issues, consider the example shown in Fig. 2.4, where the ideal inputs and the real inputs of a two DOF robot are provided to the left and right, respectively. Note that all inputs are normalized to the range of [-1,1] with +1 indicating maximum forward speed, -1 indicating maximum backward speed and 0 indicating stop. By setting two thresholds marked with dashed lines in the figure,

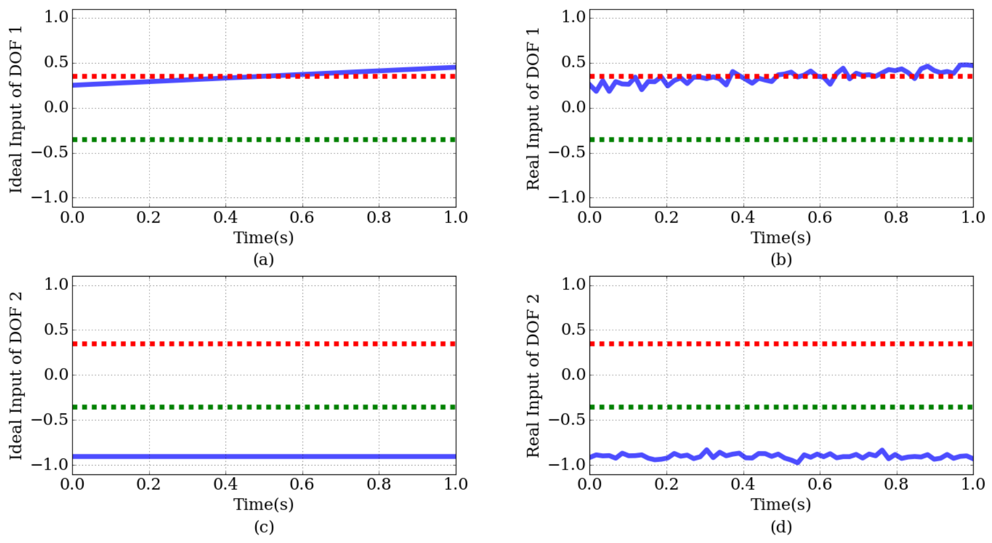


Figure 2.4: Illustration of Threshold-Crossing due to Noise

the method intends to characterize each input into three broad categories (forward, backward, near stop command) and capture command pattern changes when any input crosses a threshold. Ideally, only one set of robot states should be captured at the time when the input to DOF 1 crosses the threshold. However, input one may cross the threshold many times due to the presence of noise in the measured signal. The problem is more significant in that during the 1 s time interval, robot DOF 2 is commanded to move at nearly its maximum backward speed, which causes the robot to change its location or configuration rapidly. Thus, many recorded states where threshold-crossing occurs may consist of very different robot configurations or locations which was not intended to be captured by the threshold method.

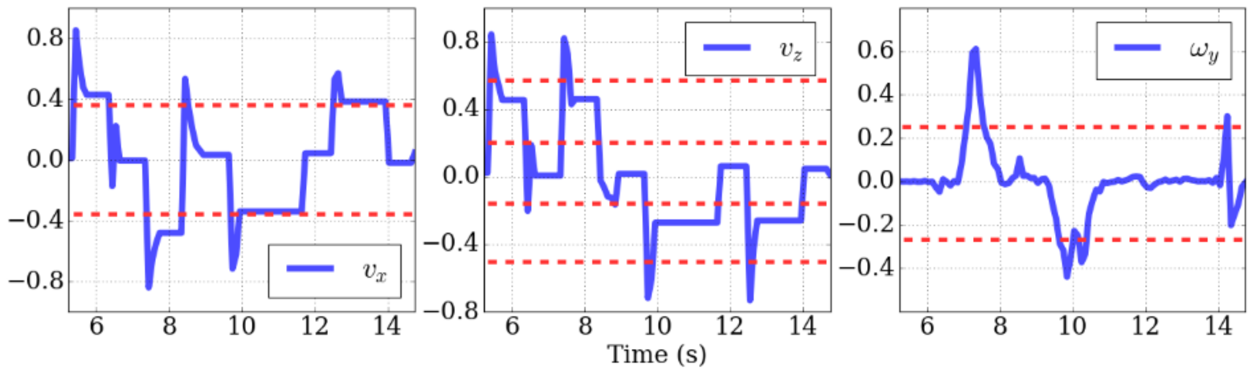


Figure 2.5: Illustration of Complex Overall Input

To illustrate another example of why the threshold method may not work when the overall command pattern is complex, consider the example provided in Fig. 2.5. This figure shows velocities (linearly mapped from inputs) of a quadrotor going through a series of acrobatic flight moves consisting of two translations (v_x, v_z) and one rotation (ω_y) in a plane (3 DOF). As one can observe from the figure, the overall input is difficult to categorize because it does not follow a specific pattern. Attempts of setting 2 and 4 thresholds are made and marked with dashed lines in different subfigures. However, none of them seem to lead to meaningful categorizations. In chapter 6 the same quadrotor example will be used to illustrate the effectiveness of the Motion Command Variance (MCV) method described in the following.

Since the overall goal is to capture robot states at times when there are significant changes in command pattern, a better solution would be to have a unified metric characterizing the command complexity for the entire demonstration. The expectation is that the metric will be insensitive to small command/measurement perturbations and yet distinguish significant changes to reveal subgoals.

2.2.3 Quantification of Operator Command Pattern

In many industrial applications where the robot is operated by a human, the operator typically controls the velocity of each DOF independently, i.e., operator's inputs are mapped to velocities of different robot DOF. Since the commands initiate motions, we refer to those as motion commands [52]. To construct the unified metric for quantifying operational command complexity, we formulate the motion command mathematically in the following manner.

At each sample time t, the motion command initiated by the human operator for an m-DOF robot can be described by a vector of the form

$$\boldsymbol{o}(t) = \begin{bmatrix} o_1, o_2, \cdots, o_m \end{bmatrix}^T \tag{2.4}$$

where o_i, $i \in 1, 2, \cdots, m$ denotes the operator's input, or the velocity it gets mapped to, for each robot DOF. For the time interval starting from t and containing n samples of motion command with a sampling period of k, the motion commands initiated by the operator is given by the following matrix:

$$\boldsymbol{O}(t) = \begin{bmatrix} \boldsymbol{O}_1, \boldsymbol{O}_2, \cdots, \boldsymbol{O}_n \end{bmatrix}^T, \tag{2.5}$$

where $\boldsymbol{O}_i = \boldsymbol{o}[t + (i-1)k]^T$, $i \in 1, 2, \cdots, n$. The covariance matrix of the motion commands captured in such a time interval is then given by

$$\Sigma(t) = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} & \cdots & \Sigma_{1m} \\ \Sigma_{21} & \Sigma_{22} & \cdots & \Sigma_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{m1} & \Sigma_{m2} & \cdots & \Sigma_{mm} \end{bmatrix}, \tag{2.6}$$

where each entry can be computed by

$$\Sigma_{ij} = E[(\boldsymbol{O}_i - \mu_i)(\boldsymbol{O}_j - \mu_j)], \tag{2.7}$$

where E denotes the expectation operator and $\mu_i = E[\boldsymbol{O}_i]$. The covariance matrix not only encodes the correlation of commands, but also contains information about subtask switching, hence indication of subgoals. That is, during a given time interval, if the operator focuses on one subtask, the commands should be similar; if the operator has just finished one subtask and switched to the next, the commands initiated should be observably different.

We use the following metric to quantify such information which we refer to as Motion Command Variance,

$$\mathcal{M}(t) = \sum_{i=1}^{m} \Sigma_{ii}, \tag{2.8}$$

where $\Sigma_{ii}, i \in 1, 2, \cdots, m$ are the diagonal elements of the covariance matrix. The MCV metric $\mathcal{M}(t)$ is the summation of the principal components of the hyper-ellipsoid formed by samples of operator's command each considered as a point in an m-dimensional space. If the operator initiated subtask switching with different command patterns during the time interval, geometrically the hyper-ellipsoid must be stretched along one or more axes which corresponds to a large value of $\mathcal{M}(t)$ [53, 54]. We choose summation of principal components because although multiplication directly relates to the volume of the hyper-ellipsoid, it does not provide the refined differentiation in that if one principal component is close to zero, the overall result will be close to zero.

We introduce a design parameter $\gamma \in [0, 1]$ which we refer to as the MCV ratio, and capture the set of desired robot states from the demonstration data set χ that satisfies the following:

$$\{x(t) \in \chi \,|\, \mathcal{M}(t) > \gamma \cdot \max_{\tau \in \Gamma} \mathcal{M}(\tau)\} \tag{2.9}$$

where $x(t)$ denotes the robot states at any sample time t, $\mathcal{M}(t)$ is the associated MCV, and Γ is the set of all sampled times for the entire demonstration. The set of robot states in (2.9) thus represents the subgoal distributions, i.e., the MCV at those points is large, thus, indicating that the operator's command changed significantly as a result of subtask-switching during the associated time intervals.

The implementation of extracting distributions of subgoals via MCV is given in Algorithm 2, where χ is the demonstration data set with ν data points, and each data point has an m-dimensional robot joint information associated with one dimensional time information. We assume that the total number of samples, denoted by $K \in \mathbb{N}^+$, collected during the entire demonstration is much greater than the number of samples $n \in \mathbb{N}^+$ that are utilized to calculate the MCV for a given time interval, namely $n \ll K$, which implies that the operator's command pattern changes for subtask-switching occur in a short time interval relative to the time duration of the entire task demonstration. Note that practical implementation requires appropriate scaling of the measurement data from different robot DOF in that different types of robot joints (prismatic, revolute) may have measurement values that belong to different ranges. In order to have the MCV weigh the input of each DOF uniformly, we normalize each input or robot state to the range of [-1,1] if they are not already in the range.

Algorithm 2 Extracting Distributions of Subgoals via MCV

```
Input: Data set $\chi \in \mathbb{R}^{(m+1)\times\nu}$, MCV ratio $\gamma \in [0,1]$, number of samples $n \in \mathbb{N}^+$ for the MCV
    time interval
Output: Subgoal distributions in the form of a set of data points
1: for $i$ in $1, 2, \cdots, K-n$ do
2:     Collect $n$ samples starting from index $i$ in time sequence each in the form of (2.4)
3:     Construct matrix in (2.5) with the $n$ collected samples
4:     Compute MCV value using (2.6) and (2.8) and attach the result to each data point
5: end for
6: Store and output all the data points with MCV values satisfying (2.9)
```

2.3 Subgoal Identification with Execution Sequence

Subgoal distributions of a given shared control task can be captured with the methods developed in Section 2.2. However, the number of subgoals, namely the number of clusters in the data set, may vary significantly due to tasks with different complexity or the fact that different operators may choose to divide the same task differently. Bayesian Non-Parametric Clustering (BNPC) is an appropriate statistical model where the Chinese Restaurant Process (CRP) prior handles the uncertain number of clusters by assuming a potential infinite number of clusters with only a finite number of them active. With the exchangeable property [55] of CRP, Gibbs sampler can be applied as the inference tool [34, 56] which will converge to the true distributions after a sufficient number of iterations.

Algorithm 3 BNPC/TO

```
Input: Data set $\chi \in \mathbb{R}^{(m+1)\times\nu}$, concentration hyper parameter $\alpha \in \mathbb{R}^+$, number of iterations $k \in \mathbb{N}^+$
Output: Cluster assignment for each data point, mean (subgoal) and covariance for each cluster
1: Initialization Assign each point to a distinct cluster
2: while Number of iterations $\leq k$ do
3:     Unassign observation from its original cluster
4:     Calculate $\operatorname{argmax}\{P(z_i|z_{-i})P(x_i|\phi_j)\}$ and assign new cluster label accordingly
5: end while
6: Finalization Sort clusters and re-assign labels increasingly according to $\overline{\tau}_J$ in (2.10)
```

Because of the dynamical nature of the SC tasks, subgoals have to be finished in a temporal sequence in order to complete the overall task. To encode the temporal ordering to the mixture model where each cluster represents a subgoal, we utilize the fact that data points are sampled with time stamps, and we label clusters sequentially in increasing order according to the data point in that cluster associated with the earliest time stamp denoted by $\overline{\tau}_J$. We define τ_i as the time stamp associated with the data point x_i, then

$$\overline{\tau}_J = \min_i \tau_i, \ \ \forall z_i = j \tag{2.10}$$

where z_i is the cluster assignment of data point i. The inference of z_i is a result of the BNPC formulation. Let $P(z_i = j|z_{-i}, \phi_j)$ be the probability of x_i having label $z_i = j$, then

$$P(z_i = j|z_{-i}, \phi_j) \propto P(z_i|z_{-i})P(x_i|\phi_j) \tag{2.11}$$

where z_{-i} is the cluster assignments of all other data points except i, x_i is the value of the data point, j is the assigned cluster label for x_i, and ϕ_j is a parameter for the base distribution. The CRP prior $p(z_i|z_{-i})$ in (2.11) is given by

$$P(z_i = j|z_{-i}) = \begin{cases} \frac{n}{n-1+\alpha}, & \text{from an existing cluster} \\ \frac{\alpha}{n-1+\alpha}, & \text{starting a new cluster} \end{cases} \tag{2.12}$$

where n is the number of data points assigned to cluster j, α is the concentration hyper parameter controlling the number of clusters that will be generated. Since all the measured values are normalized to the range of [-1,1], we use the Gaussian distribution with zero mean and unit variance [56] as the base distribution. The likelihood $P(x_i|\phi_j)$ in (2.11) is then given by

$$P(x_i|\phi_j) = \begin{cases} \mathcal{N}(x_i, \frac{n\overline{\mathbf{x}}_j}{n+1}, \mathbb{I}) & \text{from an existing cluster} \\ \mathcal{N}(x_i, 0, \mathbb{I}), & \text{starting a new cluster} \end{cases} \tag{2.13}$$

where $\overline{\mathbf{x}}_j$ is the mean of the cluster j and $\mathbb{I}$ is the identity covariance matrix. The implementation of the BNPC/TO algorithm is provided in Algorithm 3.

3. INTENT PREDICTION

Predicting a human operator's intent based on the task model learned from demonstration is another key element in the BSC framework. To assist human operators, one first needs to predict, through measurement and probabilistic inference, which subgoal the operator is seeking to achieve. Based on the prediction, automatic control input can be generated where the predicted subgoal can serve as its reference. Such automatic control input can then be blended with the operator's input to jointly control the robot and facilitate task execution.

Literature review of various methods for intent prediction in different research fields and their applications are presented in Sec. 3.1. Inspired by the existing work, Sec. 3.2 provides a specific framework for intent prediction based on subgoal models. A novel method considering both the past (empirical knowledge) and present (operator's command in real-time) information to formulate the final intent prediction algorithm is described in Sec. 3.3 and 3.4. To further improve the intent prediction, we developed another method in 3.5 to dynamically updates the subgoal transition probabilities based on observed transitions executed by the human operator.

3.1 Review of Human Intent Prediction in Various Research Fields

Interpreting human behaviors and predicting operator's intent have been studied in many research fields related to human-robot interactions. In particular, these are important topics in many fields, such as advanced driver assistance systems in vehicles, aviation flight traffic control, brain-machine interface, gaze-based interaction, intelligent power wheelchairs, etc.

In the field of intelligent driver assistance systems, some surveys on motion prediction and intent inference can be found in [57, 58]. The prediction targets (estimated maneuvers) typically include acceleration, stop, brake, turning, lane keeping/changing, passing, following, etc., [59, 60]. To collect dynamic information from the driver, vehicle, and surrounding environment, researchers employ sensors to measure vehicle parameters, GPS location, presence of surrounding objects, orientation, state of nearby traffic light, distance to intersections, driver motions and inputs to the vehicle, etc., [61, 62]. Mathematical tools for processing data acquired generally involves clustering (support vector machine, logistic regression, etc.) [60, 63] and Bayesian inference (Markov chain, hidden Markov model, etc.) [64, 65]. While most methods relate current states of the vehicle, human, and surroundings, some research suggests short term state memory with recurrent neural network for driver intent prediction [66].

In the aviation industry, the notion of a free flight system is a promising direction of development since it has many advantages over the traditional fixed-path routing system. Technological advancements in aircraft manufacturing has led to aircraft being more versatile and cheaper to produce, which is also spurring the need of a free flight system. A safe free flight system will inevitably involve understanding and predicting interactions between the human operator (pilot) and the airplane, and also between airplanes. Such studies can be found in many research papers, such as [67, 68, 69, 70, 71]. In the literature, some methods consider the actions taken by the pilot and encode them as sets of discrete values, and others consider the continuous motion of an aircraft measured by sensors. Operator function models take a set of discrete actions from the pilot and infer its intent through some hierarchical network representing different inter-related control functions. Plan detection and comparison mechanism observes pilot's discrete actions and performs intent prediction by comparing the pilot's admissible actions with some database relating actions to goals. Continuous motion inference includes methods that take aircraft states as input, and train some machine learning networks to perform classification such that different sets of states can be recognized as maneuvers with different intent. Whereas conformance monitoring systems build fault-detection mechanisms to test if the continuous motion of an aircraft matches any high-level flight plans.

Research in the areas of gaze-based interaction, brain-machine interface and intelligent power wheelchairs shares the similarity, in many cases, that they are all looking for enhancing human capabilities through understanding and predicting human behavior via the measurement of biological signals, and then providing assistance through various devices accordingly. The results have been applied to facilitate the control of different types of machines. Gaze-based interaction researchers attempt to discover and explain connections between eye movement and human intent. It has applications including monitoring distraction for driver assistance, facilitating hands-free reading on mobile devices, and operation in virtual reality environment. Pupil responses, eye movement fixation and saccade duration have been used with machine learning classification models to predict mental workload, physical effort, task types such as searching and reading [72, 73, 74, 75, 76]. In the domain of rehabilitation and restoration, brain-machine interface has been studied to help people with disability and enhance the capabilities of people without disability. Electroencephalogram, electrooculogram, and many other subtle electrical signals generated by the human body, together with the orientation and motion of human body, have been utilized to better control prosthetic limbs, exo-skeletons, and power wheelchairs [77, 78, 79, 80]. Researchers typically study the patterns of these signals by applying physical laws, kinematics, or machine learning to relate the meaning of different signals and to control various devices [81, 82, 83].

3.2 Intent Prediction based on Subgoal Model

The previous chapter on task learning provided a review of different methods developed for learning from demonstration, and developed the subgoal model that will be used for the development of shared control in this work. The subgoal approach was selected because subgoals are a compact description of a given task and they can provide references for predicting human operator intent and generating automatic control input. Although there are no existing methods that can directly relate the prediction of human operator intent to task subgoals, some work does provide clues for such a development. Drawing inspiration from the existing work and our own prior work in this area, we provide a novel method to predict intent based on the subgoal model approach.

Based on the set of learned subgoals, the action of the human operator can be used to predict which subgoal is sought. In predicting the intent of the operator's input to seek a particular subgoal, we want to incorporate both the empirical knowledge of the task process and the operator's input in real-time. In order to encode such information, we construct a factor graph representing their dependencies as shown in Fig. 3.1(a), where L denotes the last visited subgoal, X is the current joint position, V denotes the current joint velocity, G is the predicted next subgoal that the operator

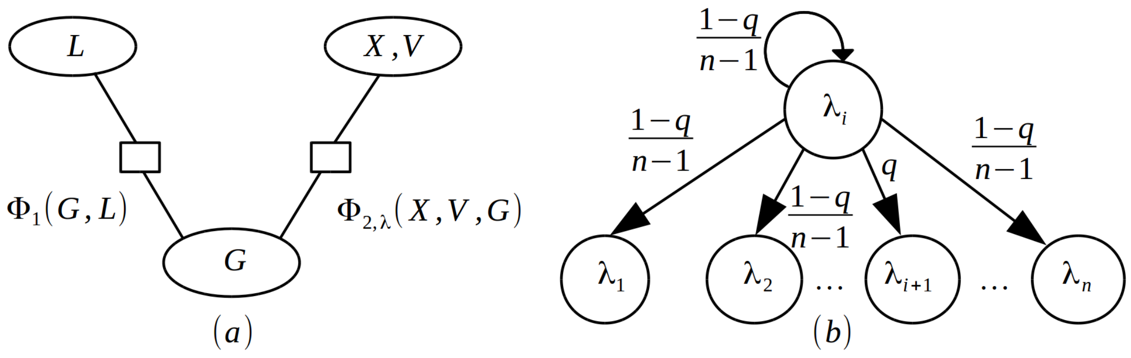


Figure 3.1: Factor Graph and Markov Chain for Prediction

wishes to reach, Φ_1 denotes the function encoding the dependency between the last visited and the predicted subgoal, and $\Phi_{2,\lambda}$ denotes the dependency between the current position, velocity and the predicted subgoal. The parameter λ contains information of all the subgoals which is known. Let $P(G|X, V, L)$ denote the probability of going to subgoal G given current position X, velocity V, and last visited subgoal L. $\Phi_1(G|L)$ be the probability of going to subgoal G given last visited subgoal L and $\Phi_{2,\lambda}(G|X, V)$ be the probability of going to subgoal G given current position X and velocity V. Then,

$$P(G|X, V, L) \propto \Phi_1(G, L)\, \Phi_{2,\lambda}(G, X, V). \tag{3.1}$$

In the following we describe a method to compute $\Phi_1(G|L)$ and $\Phi_{2,\lambda}(G|X, V)$ by employing the empirical stochastic transition matrix (ESTM) and the dynamic angle difference exponential (DADE), respectively.

3.3 Encoding Empirical Knowledge

In order to encode the empirical knowledge of subgoal transitions, we construct a Markov Chain indicated in Fig. 3.1(b). It shows the transitions happening at state i in the Markov Chain, where $q \in [0, 1]$ denotes the probability that the task will be completed in the ordered sequence. It depicts that, at any given state (each subgoal λ_i is a state in terms of the Markov Chain), the probability of going to the next subgoal is given by $q \in [0, 1]$ which reflects the confidence level that one has that the task will be completed in the ordered sequence. While jumping to any other subgoal or staying in the same subgoal all have the same probability of $\frac{1-q}{n-1}$ for n-state Markov Chain. Based on such construction, to encode the information we define ESTM as

$$T = \begin{bmatrix} \frac{1-q}{n-1} & q & \frac{1-q}{n-1} & \cdots & \frac{1-q}{n-1} \\ \frac{1-q}{n-1} & \frac{1-q}{n-1} & q & \cdots & \frac{1-q}{n-1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \frac{1-q}{n-1} & \frac{1-q}{n-1} & \ddots & \ddots & q \\ q & \frac{1-q}{n-1} & \cdots & \cdots & \frac{1-q}{n-1} \end{bmatrix}. \tag{3.2}$$

Let T_{ij} denote the element in the i-th row and j-th column of T. Then, the probability of going to subgoal j, namely G, given the last visited subgoal i, namely L, is given by

$$\Phi_1(G, L) \propto P(G|L) \propto T_{ij}. \tag{3.3}$$

3.4 Integrating Dynamic Prediction

To incorporate into our prediction the dynamic measurements and operator input in real-time, we introduce the idea of dynamic angle difference. Figure 3.2 shows the angles between current action, which is the velocity commanded by the operator in real-time, and closed-loop actions of going to each of the subgoals. In Fig. 3.2(a), $\lambda_1, \lambda_2, \cdots, \lambda_n$ denote the positions of subgoal

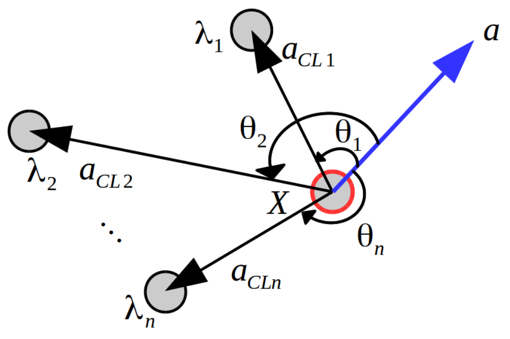


Figure 3.2: Dynamic Angle Difference

$1, 2, \cdots, n$ in joint space, respectively; a denotes the operator's current action; $a_{CL1}, a_{CL2}, \cdots, a_{CLn}$ denote the closed-loop actions of a proportional controller for going to subgoal $1, 2, \cdots, n$, respectively; $\theta_1, \theta_2, \cdots, \theta_n \in [0, \pi]$ denote the angles between current action and closed-loop subgoal-tracking actions. We say that the dynamic probability of operator's action a of going to subgoal j, is inversely proportional to the dynamic angle difference θ_j. In other words, the subgoal with the smallest angle between closed-loop action and operator's current action is our predicted target. We can construct such a probability is by defining

$$\Phi_{2,\lambda}(G, X, V) \propto P(G|X, V) \propto \frac{1}{\theta_j}. \tag{3.4}$$

However, such a construction has two drawbacks. It approaches singularity when θ_j is close to zero. The output of such a function also would overweight the probability from the stochastic matrix when the angle is small. To overcome these issues we propose the dynamic probability in such context as given by the following dynamic angle difference exponential:

$$P(G|X, V) \propto e^{-\theta_j} \tag{3.5}$$

where

$$\theta_j = \arccos\left(\frac{a \cdot a_{CLj}}{||a|| \cdot ||a_{CLj}||}\right) \tag{3.6}$$

Therefore, DADE solves the problem of singularity and normalizes its output such that the probabilities of ESTM and DADE are in the same scale.

The net equation for subgoal prediction is then given by

$$P(G|X, V, L) \propto T_{ij} \cdot e^{-\theta_j} \tag{3.7}$$

and the predicted subgoal is the one with the maximum probability among all possible scenarios, and is given by

$$\underset{j}{\text{argmax}}\,\{T_{ij} \cdot e^{-\theta_j}\} \tag{3.8}$$

where i is the index of the last visited subgoal, which is known, j is the index of a possible target subgoal in the set of all possible subgoals λ. To get a valid probability for the subgoal predictions, we normalize (3.7) by

$$P(G = \iota | P, V, L) = \frac{T_{\iota k} \cdot e^{-\theta_\iota}}{\sum\limits_{j,\lambda} T_{ij} \cdot e^{-\theta_j}} \tag{3.9}$$

where $\iota = 1, 2, \cdots, n$; this renders $P(G = \iota | P, V, L) \in [0, 1]$ such that the predictions made by (3.9) sum to 1. The implementation of such prediction is given in Algorithm 4.

Algorithm 4 Prediction of Target Subgoal

```
Input: Set of all subgoals, design parameter $q$ for ESTM, Operator's action in real-time
Output: Predicted subgoal and its probability
1: Construct ESTM by (3.2)
2: for each subgoal do
3:     Compute its probability using (3.9)
4: end for
5: Compare subgoal prediction probabilities using (3.8)
6: Output the the index of the maximum as prediction result, and its value as the probability
```

3.5 Learning Subgoal Transition Probabilities

In some applications, the subgoal transition probabilities from empirical experience may not be objective, or the change of task environment may require the operator to execute the subgoals in an order that is different from its nominal order. For these situations, instead of assigning fixed transition probabilities based on task-specific knowledge, we propose a procedure to continuously update the transition probabilities of the Markov Chain based on observed subgoal transitions during task execution. In the following, we discuss the details of the procedure. To start with, we initialize the stochastic transition matrix as

$$T = \begin{bmatrix} 1/n & 1/n & \cdots & 1/n \\ 1/n & 1/n & \cdots & 1/n \\ \vdots & \vdots & \ddots & \vdots \\ 1/n & 1/n & \cdots & 1/n \end{bmatrix}. \tag{3.10}$$

This initialization indicates that there is no prior knowledge or observed evidence of subgoal transitions, i.e., all subgoal transitions are equally likely. Without loss of generality, we assume that the operator is at the i^{th} subgoal to begin with.

If the operator takes an action A, the prediction model will generate a predicted subgoal with index J given by (3.8) and normalized probability P given by (3.9). We denote such an action, prediction and probability by the triplet $\mathcal{A} = (A, J, P)$. Assuming the index of the last visited subgoal is i, and the operator takes a sequence of actions generating

$$\mathcal{A}_x = (A_x, J_x, P_x) \tag{3.11}$$

where $x \in \{1, 2, \cdots, m\}$; if the robot ends up in subgoal j and $J_x = j$ for a set of $\mathcal{A}_x$'s before the robot getting to the subgoal, we say that one transition update cycle is observed and completed. In other words, if the robot visited a subgoal which happens to be the result of intent prediction based on a set of actions taken by the human operator, it is an observed and correctly predicted transition. Once such a cycle is observed and completed, we update the i^{th} row, encoding transition probabilities from subgoal i, of the stochastic matrix

$$T_i = \begin{bmatrix} T_{i1} & T_{i2} & \cdots & T_{ij} & \cdots & T_{in} \end{bmatrix} \tag{3.12}$$

according to the following rules.

Let the K^{th} update ($K = 1, 2, 3, \ldots$) of the transition matrix element T_{ij} be given by $T_{ij,K}$; note that T_{ij} is the transition probability from subgoal i to j. We consider the following update law for making adjustment to T_{ij} at the K^{th} update cycle:

$$T_{ij,K} = T_{ij,K-1} + \beta_K(Q - T_{ij,K-1}) \tag{3.13}$$

where $\beta_K \in (0, 1]$ is the update gain or rate at the K^{th} cycle and $Q \in (\frac{1}{n}, 1)$ is a design parameter indicating the upper bound to which T_{ij} is expected to converge if repeatedly the human operator chooses to visit subgoal j given that the last visited subgoal is i. Intuitively, one could choose the update rate β_K to be a constant in the range of $(0, 1]$. The inference is that once a transition from subgoal i to j is observed, the corresponding term T_{ij} in the stochastic matrix is increased towards its specified upper bound Q by the difference between Q and its current value multiplied by a constant. However, we would like to relate β_K dynamically with the prediction probability. Specifically, if a transition is predicted and observed, we increase T_{ij} by the difference between Q and its current value multiplied with the prediction probability which is given by

$$\beta_K = \frac{e^{-\theta_j} T_{ij,K-1}}{\sum\limits_j e^{-\theta_j} T_{ij,K-1}} \tag{3.14}$$

where $e^{-\theta_j}$ is the result of DADE generated by the first action in the set of $\mathcal{A}_x$'s of (3.11) within the K^{th} transition update cycle. One could interpret the selection of β_K in this manner as follows: If a subgoal transition is observed and predicted, instead of increasing by a fixed amount towards its upper bound, we increase it by the amount reflecting our confidence. In other words, the transition probability changes by a larger amount towards the specified upper bound Q if the observed transition is associated with a higher prediction probability and vice versa.

The remainder $n - 1$ entries in (3.12) are updated as follows:

$$T_{ik,K} = T_{ik,K-1} - \frac{T_{im,K-1}}{\sum\limits_{m \neq j} T_{im,K-1}} \beta_K(Q - T_{ij,K-1}) \tag{3.15}$$

where $k \neq j, k = 1, 2, \ldots, n$. Note that the update laws, (3.13) and (3.15), guarantee that after each transition update cycle, T is still a valid stochastic transition matrix since for $k \neq j$

$$\sum_{k\neq j} \frac{T_{ik,K-1}}{\sum\limits_{k\neq j} T_{ik,K-1}} \beta_K(Q - T_{ij,K-1}) = \beta_K(Q - T_{ij,K-1}).$$

Following these update laws for the elements of the stochastic transition matrix, if the operator visits subgoal j a number of times consecutively after visiting subgoal i, the i^{th} row will evolve into the following form

$$T_i = \begin{bmatrix} T_{i1} & T_{i2} & \cdots & T_{ij} = Q & \cdots & T_{in} \end{bmatrix} \tag{3.16}$$

where the j^{th} entry converges to Q. The remainder of the elements in that row are proportional to the number of times the corresponding subgoals are visited until T_{ij} converges to Q given that i was the previous visited subgoal. We refer to this property resulting from the update of the transition matrix in this fashion as subgoal-visit memory. The following theorem formalizes this notion of subgoal-visit memory and provides the convergence of the update laws. The convergence of the transition probability T_{ij} to Q with repeated continuous transitions from subgoal i to j is clear when $\beta_K \in (0, 1]$ since the update law resembles a stable first-order linear filter. However, the following theorem shows even with a dynamically changing β given in (3.14), which renders the evolution equation for the transition probability T_{ij} non-linear, T_{ij} still converges to Q.

Theorem 1. *Suppose the last visited subgoal is i. For the transition from subgoal i to subgoal j, let the elements of the i^{th} row of the transition matrix be updated according to the update laws given by* (3.13) *and* (3.15). *Let the update rate be given by* (3.14). *If the j^{th} subgoal is visited from the i^{th} subgoal consecutively for a large number of times, then the ij^{th} element of the transition matrix converges to Q and the sum of the remaining elements of the i^{th} row converges to $1 - Q$.*

Proof. To simplify the notation, we will denote $T_{ij,K}$ as T_K. Thus, we can write (3.13) in the following form:

$$T_K = T_{K-1} + \beta_K(Q - T_{K-1}) \tag{3.17}$$

Note that at the previous transition update cycle, one has

$$T_{K-1} = T_{K-2} + \beta_{K-1}(Q - T_{K-2}). \tag{3.18}$$

Substitution of (3.18) into (3.17) and rearranging terms, we obtain

$$T_K = T_{K-2} + \beta_{K_1}(Q - T_{K-2}) \tag{3.19}$$

where

$$\begin{aligned} \beta_{K_1} &= \beta_K + \beta_{K-1} - \beta_K\beta_{K-1} \\ &= 1 - (1 - \beta_K)(1 - \beta_{K-1}) \end{aligned} \tag{3.20}$$

Applying the same reasoning further, one has

$$T_K = T_{K-3} + \beta_{K_2}(Q - T_{K-3}) \tag{3.21}$$

where

$$\begin{aligned} \beta_{K_2} &= \beta_{K_1} + \beta_{K-2} - \beta_{K_1}\beta_{K-2} \\ &= \beta_K + \beta_{K-1} + \beta_{K-2} \\ &\quad - \beta_K\beta_{K-1} - \beta_{K-1}\beta_{K-2} - \beta_K\beta_{K-2} \\ &\quad + \beta_K\beta_{K-1}\beta_{K-2} \\ &= 1 - (1 - \beta_K)(1 - \beta_{K-1})(1 - \beta_{K-2}) \end{aligned} \tag{3.22}$$

Thus, given T_0 as the initial value of T_{ij} when the consecutive transitions from subgoal i to subgoal

j was initiated, the K^{th} update is given by

$$T_K = T_0 + \beta_{K_K}(Q - T_0) \tag{3.23}$$

where

$$\beta_{K_K} = \sum_i \beta_i - \sum_{i \neq j} \beta_i \beta_j + \sum_{i \neq j \neq k} \beta_i \beta_j \beta_k - \cdots \tag{3.24}$$

where $i, j, k \in \{1, 2, \cdots, K\}$. Note that (3.24) can also be written in the form

$$\beta_{K_K} = 1 - \prod_{m=1}^{K} (1 - \beta_m) \tag{3.25}$$

where β_m, from (3.14), is given by

$$\beta_m = \frac{e^{-\theta_j} T_{ij,m-1}}{\sum\limits_j e^{-\theta_j} T_{ij,m-1}}.$$

Since $\beta_m \in (0, 1]$, we have

$$\lim_{K \to \infty} \prod_{m=1}^{K} (1 - \beta_m) = 0. \tag{3.26}$$

Thus,

$$\lim_{K \to \infty} \beta_{K_K} = 1. \tag{3.27}$$

In other words, the probability of the j^{th} entry of the i^{th} row of the stochastic transition matrix will converge to the specified upper bound Q if the operator visits subgoal j enough times consecutively after visiting subgoal i. Further, from (3.15), the sum of the remaining elements of the i^{th} row converges to $1 - Q$. This completes the proof. $\square$

Remark 1. *The subgoal-visit memory property after T_{ij} converges to Q may explained in the following manner with* (3.15). *If the operator visited subgoal k for example at the K^{th} update cycle, it must lead to a higher probability from the dynamic prediction encoded in β_K. This would* *result in a larger value of the term* $\beta_K(Q - T_{ik,K-1})$ *in* (3.13). *Also, observe that the term*

$$\frac{T_{ik,K-1}}{\sum\limits_{k \neq j} T_{ik,K-1}}$$

in (3.15) *is proportional to the probability of visiting subgoal* k *over the summation of all other transition probabilities except* T_{ij}. *This means that if the operator visited a subgoal more than others before convergence, the resulting large probability will reduce, due to* T_{ij} *converging to* Q*, but remain greater than the rest.*

Remark 2. *The selection of the parameter* $Q \in (\frac{1}{n}, 1)$ *depends on various factors of the shared control application. If the environmental uncertainties associated with the task are critical, then this value must be chosen smaller than unity with a reasonable margin; that is, the value of* $(1-Q)$ *is distributed to all other* $n-1$ *probabilities, and this by design should not be too small to allow for non-nominal transitions to subgoals other than* j.

Remark 3. *The proposed update law for transition probabilities also has a distinct computational advantage in the sense that the update is computed based only on current values without the need to keep track of past transition probabilities which makes the implementation of the update law straightforward.*

Thus, this update method provides a way to dynamically adjust the subgoal transition probabilities used in the ESTM.

4. INPUT BLENDING

This chapter provides methods for blending of inputs from the human operator and the automatic control system. A literature review on different blending schemes and parameter selections is provided in Sec. 4.1. A method for blending inputs with intent and conflict awareness based on subgoals is presented in Sec. 4.2. This method offers a mechanism to: (1) generate and blend automatic control to assist the human operator if intent prediction indicates that the operator is seeking a specific subgoal and (2) yield control authority to the human operator in situations when the human operator may take unexpected actions to avoid danger instead of seeking any of the subgoals. In practical implementation, human and automatic control inputs may have values in very different ranges or have very different magnitude, they must be properly processed and scaled accordingly before getting blended such that the blended input is an appropriate combination of both. A method for such processing and scaling is provided in Sec. 4.3, which not only considers normalization of control inputs, but also the changing of controller gain according to operator's input magnitude for a more responsive assistance.

4.1 Methods for Blending Inputs

Based on specific applications (smart wheelchairs, surgical robots, manufacturing and construction machines, etc.) and various design concerns, researchers have developed different methods for selecting the blending parameter. For example, in research related to construction robots, blending control inputs with a parameter optimizing various cost functions internal to the robot are proposed in [12, 30]. Researchers in [31] suggested a blending parameter that depends on the probability calculated from a finite state machine quantifying the subgoal transition probabilities of a given construction task. The work in [49] proposed a method for choosing the blending parameter based on a function measuring the difference between automatic control and human input. Researchers in [84] introduced a way to calculate and combine only the admissible inputs from human and automatic system in the control of a simulated inverted pendulum.

While most existing studies employ linear blending, such as in [85, 86, 87, 88, 89], where the final control input u is given by

$$u = (1 - \alpha)u_h + \alpha u_a$$

where u_h is the human input, u_a is the input from automatic feedback control system, and $a \in [0, 1]$ is the blending parameter. Some researchers suggest that the linear blending method is not the optimal option [90, 91, 92, 93]. Though the non-linear blending formulations, such as the one proposed in [90, 91], are theoretically more comprehensive by modeling human intent and the environment as various probability distributions and stochastic processes, there are complexities associated with incorporating physical insights of the application as well as practical implementation.

4.2 Adaptive Blending with Conflict Awareness

Considering the advantages and disadvantages of different blending methods and the subgoal-based learning and predicting framework, we propose a linear blending method where the blending parameter dynamically is dynamically updated according to the prediction confidence, and also provides a mechanism to yield control authority to the human operator when a conflict is detected based on the human operator's action. Based on DADE, we define a parameter called the Deviation Threshold Angle (DTA) θ_d to encode such conflict awareness. We say that if $\theta_i > \theta_d$, $\forall i \in 1, 2, \cdots, n$, then the blending scheme will yield control authority to the human operator, that is, if all the angles from DADE are greater than the threshold value, we assume that the operator is not seeking any of the learned subgoals and fully yield control authority to the operator. Otherwise, we will share the control authority between the human operator and the closed-loop subgoal-tracking action according to our prediction probability. This blending scheme thus encodes conflict-awareness by taking into consideration all subgoals from the entire joint space in real-time, unlike the ones considered in [94] where only conflicts between a single pair of starting and destination points are considered.

When blending control inputs according to the prediction probability using linear blending, one has to normalize the prediction probability to be in the range of $[0, 1]$. We consider this problem by defining the probability p^* of the predicted subgoal to be

$$p^* = P(G = k|P, V, L) = \frac{T_{ik} \cdot e^{-\theta_k}}{\sum\limits_{j,\lambda} T_{ij} \cdot e^{-\theta_j}} \tag{4.1}$$

which renders $P(G = k|P, V, L) \in [0, 1]$, and k is the index of our predicted next subgoal from the previous section. Then, we linearly blend the operator's input, denoted by u_h, with the real-time closed-loop subgoal tracking controller input, denoted by u_e, through

$$u = (1 - p^*)u_h + p^* u_a \tag{4.2}$$

where u is the blended control input to the robot.

Taking into account the DTA-based conflict-awareness, we specify the total control input as

$$u_b = \begin{cases} (1 - p^*)u_h + p^* u_a, & \exists \theta_i < \theta_d, \;\; i \in 1, 2, \cdots, n \\ u_h, & \text{otherwise} \end{cases} \tag{4.3}$$

where u_b is the final blended control input with conflict-awareness. The real-time implementation of this BCS method is given in Algorithm 5.

Algorithm 5 Real-Time BSC/CA

```
Input: Operator control input $u_h$, automatic control input $u_e$, probability $p^*$ of predicted subgoal,
    dynamic angle difference vector $\theta \in \mathbb{R}^n$, threshold parameter $\theta_d$
Output: Blended control input $u_b$
    Initialization $\overline{p} = p^*$
    if $\min\, \theta > \theta_d$ then
3:      $\overline{p} = 0$
    end if
    Calculate $u_b = (1 - \overline{p})u_h + \overline{p}u_a$
```

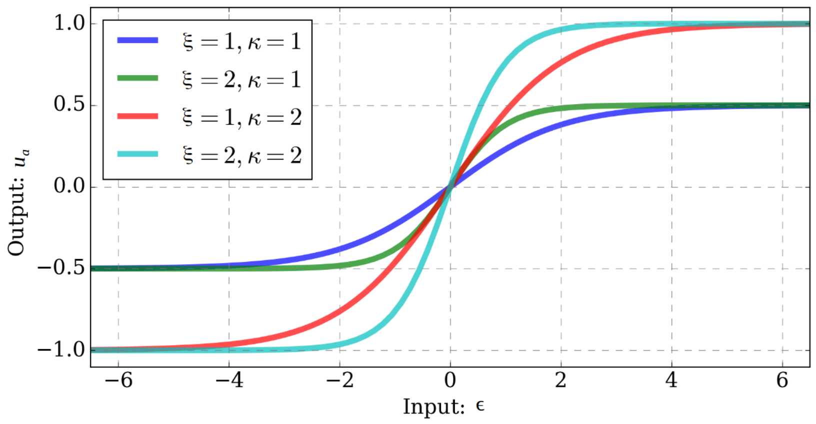


Figure 4.1: Parameter Selection vs. Dynamic Sigmoid Controller

4.3 Coordinating Controller Gain and Normalizing Different Inputs

There are two additional aspects one should consider. First, bringing the automatic control and human inputs into the same scale to facilitate appropriate blending. Second, relating the controller gain, namely the response speed, to the magnitude of operator's input in some reasonable fashion based on intent prediction. Hence, we propose the concepts of dynamic sigmoid controller (DSC) and operator's effective magnitude for seeking a particular subgoal.

The dynamic sigmoid controller is given by

$$u_a = \frac{2\kappa}{1 + e^{-\xi\epsilon}} - \kappa \tag{4.4}$$

where ϵ is the error between the current measured position and desired position based on predicted subgoal, the parameter κ is related to the limits of control saturation, and the parameter ξ controls the controller gain and is related to operator's input magnitude. Fig. 4.1 provides some examples of the mapping of DSC with different parameter selections. With the upper and lower saturation limits imposed to the input from automatic control, it is then possible to scale the operator's input accordingly such that u_h and u_a in (4.3) are in the same scale before they are blended to provide overall input.

To relate the magnitude of the operator's input to the controller gain for tracking the predicted subgoal, we propose the concept of operator's effective magnitude (OEM). It is defined as the magnitude of the portion of the operator's input effectively acting along the direction as the closed-loop action for tracking the predicted subgoal. We use the following method based on vector projection to quantify such OEM. Let ϕ be the effective magnitude which is given by

$$\phi = \overline{a} \cdot \frac{a_{CLi}}{||a_{CLi}||} = \overline{a}\cos\theta_i \tag{4.5}$$

where θ_i denotes the dynamic angle between operator's action vector and the closed-loop action vector a_{CLi} for tracking the predicted subgoal, λ_i and $\overline{a}$ is the normalized operator's action vector with each of its element normalized along the corresponding joystick axis. To illustrate this, Fig. 4.2(a) provides the axis-wise normalization of the operator's action input, where each joystick axis is normalized to take a value in the interval $[-1, 1]$. Fig. 4.2(b) provides a two-dimensional visual illustration for calculating the operator's effective magnitude.

With such a formulation, it is then possible to relate the controller gain parameter ξ in (4.4) to the operator's input in a manner that the effective magnitude is proportional to the controller gain. We propose that in (4.4), the parameter ξ to be

$$\xi = \nu\phi \tag{4.6}$$

where $\nu \in \mathbb{R}^+$ is a design parameter that scales the proportionality between the effective magnitude ϕ and controller gain parameter ξ.

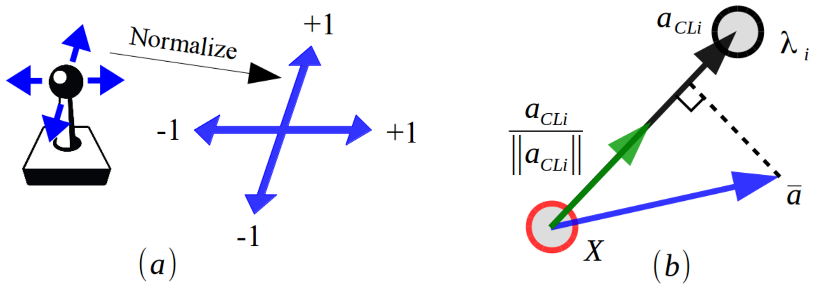


Figure 4.2: Normalized Joystick Axes and Visualization of Effective Magnitude

For a faster response, a larger value should be assigned to ν in (4.6). If the aim is to have the controller gain to be very sensitive to the effective magnitude of operator's input, then one can assign ν to be of small value, such as in the case surgical robots where the systems have fast dynamics and operational precision is critical. In the case of experiments with the excavator, we selected a larger value for ν because the hydraulic dynamics of the excavator are slow [95]. The DSC controller then effectively resembles a proportional controller with a large gain and saturates very fast once the error term is provided. Also notice that the operator's input along different axes gets normalized to the range of $[-1, 1]$ to facilitate the formulation of effective magnitude; thus, in practice, it is suggested to set κ such that both inputs are in the range of $[-1, 1]$ for blending. The blended input then gets mapped to the required range of physical actuation signals, for example PWM duty cycles or voltage levels, by a proper affine transformation.

5. SUBGOAL ADJUSTMENT

A dynamic adjustment of the nominal subgoals learned from demonstration is necessary for BSC to provide effective long-term assistance to human operators. One important reason is that the environment in which the task is performed, or the target object may change significantly due to repeated execution of the task. The nominal subgoals learned from the demonstration performed in initial task condition have to be adjusted to adapt to such changes. This is a significant challenge for BSC and has not been addressed in the literature. This chapter addresses this challenge by proposing a method to modify the nominal subgoals gradually during dynamic task execution. Section 5.1 introduces the motivation for subgoal adjustment. The characteristics of operator behavior, which inspired the development of subgoal adjustment tools, are provided in Sec. 5.2. A method for detecting the subgoal adjustment actions initiated by a human operator is provided in Sec. 5.3, and a tool for incorporating such adjustment information into the nominal subgoal model is presented in Sec. 5.4.

5.1 Necessity of Adjusting Nominal Subgoals

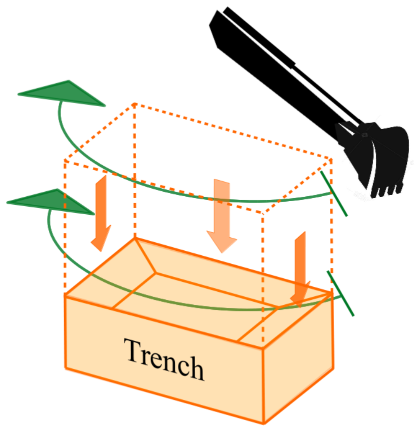


Figure 5.1: Gradual but Significant Accumulative Changes of Target Object

In collaborative robotics applications, operators are largely required to perform cyclic tasks where each cycle is similar but lacks well-organized repetition. Subgoals can change significantly during the process of task execution. For example, Fig. 5.1 shows gradual but significant accumulative change of the trenching area and depth in a construction task with excavators as execution units. These changes typically can be adapted by making small adjustments in subgoals during each task cycle while having the operational patterns remain largely the same. Execution of such tasks will benefit significantly from the application of BSC. However, BSC methods which consider only nominal subgoals without making any adjustments during the task operation will not be able to provide effective assistance in off-nominal situations. If nominal subgoals are assumed all the time, there could be gradual deterioration of automatic control assistance.

Human operators, with their unique sensory abilities, environmental awareness, and domain knowledge, can generally make decisions and take actions to adapt to those changes. Predicting those actions and including them a priori into a model is generally not feasible. Yet, meaningful interpretations of such decisions or actions can be obtained in real-time via operator intent and can be encoded into dynamic subgoal adjustment which will lead to improving the long-term performance of blended shared control.

5.2 Characteristics of Operator Behavior

Based on observations of excavator operators during task execution, subgoal adjustments by human operators are typically realized by making small modifications to the initial/nominal subgoals. Such observations are generally supported by studies and results from extensive research in psychology [96, 97], i.e., complex human behavior is learned through the modification of simpler behaviors, especially when environmental changes are gradual. For example, operators in excavator trenching tasks usually make small adjustments to the subgoals each cycle to compensate for the environmental changes, since it is an intuitively simple way to stay productive, such as removing as much dirt as possible. With this viewpoint, we employ the following strategy for the subgoal adjustment. If the operator commands a velocity for each robot joint, and if the operator could make a correct adjustment to compensate for the subgoal changes, then the operator knows the level of adjustment needed for each robot joint. It is assumed that such knowledge is instantiated in the operator's actions taken for each robot joint near the subgoal which needs to be adjusted. In the following, we will describe how to adjust the subgoals based on operator actions by utilizing the notions of a hyper-rectangle (which embeds the subgoal with proper volume) and a skill-weighted action integral (which accounts for the amount of adjustment needed).

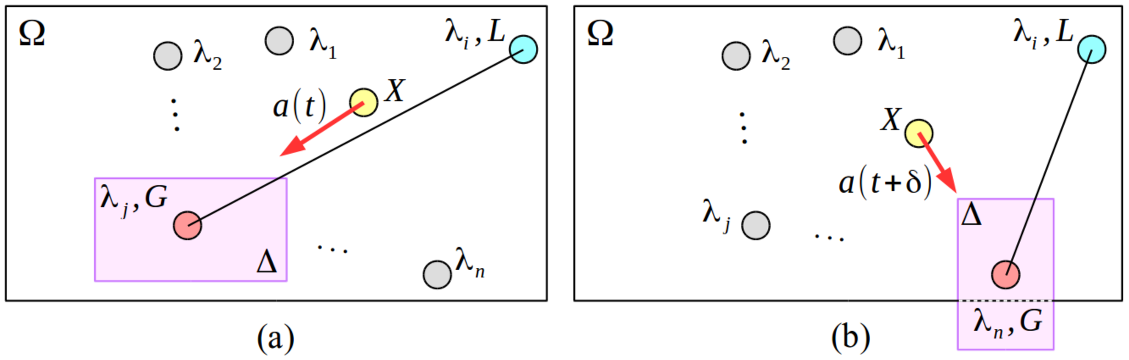


Figure 5.2: Joint Space Ω and Adjustment Encoding Hyper-Rectangle Δ in 2D

5.3 Detecting Subgoal Adjustment Actions

In order to detect operator's subgoal adjustment information, with the assumption that such behavior appears near the target subgoal, we define an Adjustment Encoding Hyper-Rectangle (AEHR), denoted by Δ to enclose the subgoal vicinity. It is a generalization of a rectangle to higher dimensions and is dynamically created and centered at the predicted subgoal. The AEHR also dynamically changes its center and edge length when the prediction updates the target subgoal. The AEHR either lies within the m-dimensional robot joint space, denoted by Ω, or overlaps with it. This process is illustrated in Fig. 5.2 with a two-dimensional visualization of the AEHR in the robot joint space Ω. At time t, as shown in Fig. 5.2(a), the prediction result is λ_j according to operator's action vector $a(t)$; hence, the AEHR is created and centered at λ_j. However, at time $t + \delta$, as shown in Fig. 5.2(b), the prediction updates the target to λ_n based on the new action vector $a(t + \delta)$; thus, the AEHR changes its center and edge length accordingly. The AEHR thus defines a dynamic region around the target subgoal whose size is dependent on the position of the last visited subgoal and predicted target subgoal. Once the robot enters into the subgoal vicinity, until it leaves, the main focus is to collect the adjustment information instead of predicting.

The next challenge is to properly construct the volume of the AEHR, i.e., a reasonable small region centered at the subgoal in which adjustment action will potentially take place. We want the size of AEHR to be large enough to accumulate and encode sufficient adjustment information from the operator and small enough such that the prediction is updated for active assistance as much as possible. The solution we propose for this problem is to employ a Hyperbolic Slope Transition Function. It is defined to be the ratio of the edge length of the AEHR and the distance of the predicted subgoal along the direction of that edge length. Further, we set an upper limit on this ratio to ensure that the AEHR is not too large when the distance between the last visited subgoal and predicted subgoal is large.

We construct the HSTF function as a single function that is smooth and easy to implement in practice. When the subgoals are close, we define

$$\delta_r = d_r \tag{5.1}$$

where δ_r, $r \in 1, 2, \cdots, m$, denotes the edge lengths of Δ and d_r denotes the normalized distance between subgoals along the r-th coordinate of the m-dimensional joint space Ω. The normalized distance $d_r \in [0, 1]$ is given by

$$d_r = \frac{|\lambda_{ir} - \lambda_{jr}|}{\max\limits_{p,q} |s_p - s_q|} \tag{5.2}$$

where λ_{ir} and λ_{jr} denote the distance of subgoals i and j along the r-th coordinate of the m-dimensional joint space, and s_p and s_q denote two arbitrary points along the same coordinate. When the subgoals are far from each other, we set a constant limit for each δ_r by

$$\delta_r = \mu \tag{5.3}$$

where $\mu \in [0, 1]$ is the constant upper limit. We then connect $\delta_r = d_r$ and $\delta_r = \mu$ together smoothly by constructing the HSTF given by

$$\delta_r = \Gamma^{\mu}_{\kappa,\xi}(d_r) = \xi d_r + \frac{1 + \tanh[\kappa(\xi d_r - \mu)]}{2}(\mu - \xi d_r) \tag{5.4}$$

where $\kappa \in \mathbb{R}^+$ and $\xi \in (0, 1)$ are design parameters; κ controls by how much (5.4) resembles the combination of $\delta_r = d_r$ and $\delta_r = \mu$, and ξ is selected based on κ to ensure that the maximum slope of (5.4) is less than $\delta_r = d_r$.

5.4 Encoding Adjustment Information

To encode operator adjustment actions within the AEHR, we define a method for interpreting and encoding such information via a Skill-Weighted Action Integral (SWAI) ρ, which is given by

$$\rho = \beta \int a(t)\mathbb{1}(\Omega \cap \Delta | X \neq G) \, dt \tag{5.5}$$

where $\beta \in \mathbb{R}^+$ is a design scaling parameter (controls how much the designer wants to scale the integration results based on knowledge of the skill level of the operator), $\rho \in \mathbb{R}^m$ is the vector denoting the adjustment for the m-dimensional target subgoal, and the action vector is given by

$$a(t) = [a_1(t) \; a_2(t) \; \cdots \; a_m(t)]^T \in \mathbb{R}^m \tag{5.6}$$

which are the operator inputs, where $a_1, a_2, \cdots, a_m$ denote operator command for each robot joint. The indicator function $\mathbb{1}(\Omega \cap \Delta | X \neq G)$ is given by

$$\mathbb{1}(\Omega \cap \Delta | X \neq G) = \begin{cases} 1, & X \in (\Omega \cap \Delta) \text{ given } X \neq G \\ 0, & \text{otherwise} \end{cases} \tag{5.7}$$

The indicator function provides a way for SWAI to encode subgoal adjustment information only when the robot is within the intersection of Ω and Δ and when it has not reached the target subgoal. Figure 5.3 provides an illustration of such conditions. Figure 5.3(a) illustrates this concept by showing that in Δ, the robot trajectory τ is perturbed by the operator's subgoal adjustment actions $a(t), a(t+\delta)$. The closed-loop tracking actions for target subgoal G are denoted by $a_{CL}(t), a_{CL}(t+\delta), a_{CL}(t+2\delta)$, the positions of the robot are denoted by $X(t), X(t+\delta), X(t+2\delta)$. Notice that at the time $t+2\delta$, the operator may be satisfied with the adjustment and take no additional action, thus, $a(t+2\delta) = 0$ is not shown in the figure. Figure 5.3(b) illustrates another scenario where the SWAI finishes encoding adjustment information since the robot has left the AEHR without reaching the target subgoal.

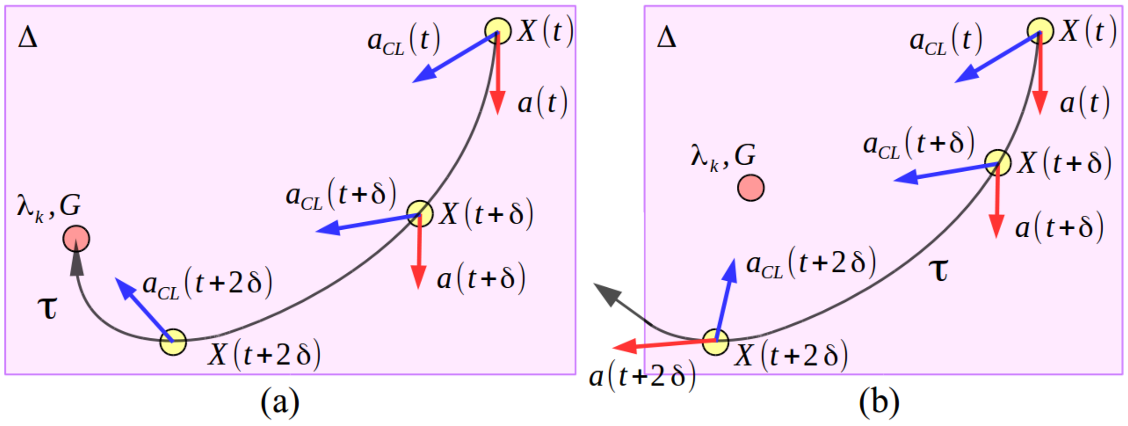


Figure 5.3: Skill Weighted Action Integration in the Target Subgoal Vicinity

Once the robot either reaches the subgoal or leaves the AEHR as shown in Fig. 5.3, we update the last visited subgoal L to the previous prediction result G which is the subgoal λ_k in subgoal set λ. Then, adjustment to this subgoal is made in the m-dimensional robot joint space by

$$\lambda_k^+ = \lambda_k^- + \rho \tag{5.8}$$

where λ_k^+ denotes the adjusted position, λ_k^- denotes its previous position.

One additional aspect is that once the robot enters Δ, we stop the prediction updates and keep the last predicted subgoal as the target subgoal, and we fix the blending parameter to a constant such that human operator's blending weight is always higher than that of the automatic control input for tracking the target subgoal. We also stop updating the operator's effective magnitude and fix it with its last computed value. Therefore the shared control assistance is still providing active assistance but the operator can always override the assistance to make arbitrary adjustments.

6. BLENDED SHARED CONTROL EXPERIMENTS AND RESULTS

This chapter presents the hardware platforms, design of experiments, and results that corroborate the methods developed in the previous chapters. It starts by introducing the physical platforms with their hardware and software in Sec. 6.1. Since the methods developed for learning can work independently and yield meaningful results for both full autonomy and shared control tasks, the rest of the chapter is organized as follows. First, we illustrate the tasks being demonstrated to the algorithms and present the learning results by themselves in Sec. 6.2. Then, the integrated results on performance improvement for BSC with learning, prediction, and subgoal adjustment are provided in Sec. 6.3 and 6.4, respectively. Lastly, we another experiment to corroborate the effectiveness of the subgoal transition learning method and the corresponding results are presented in Sec. 6.5.

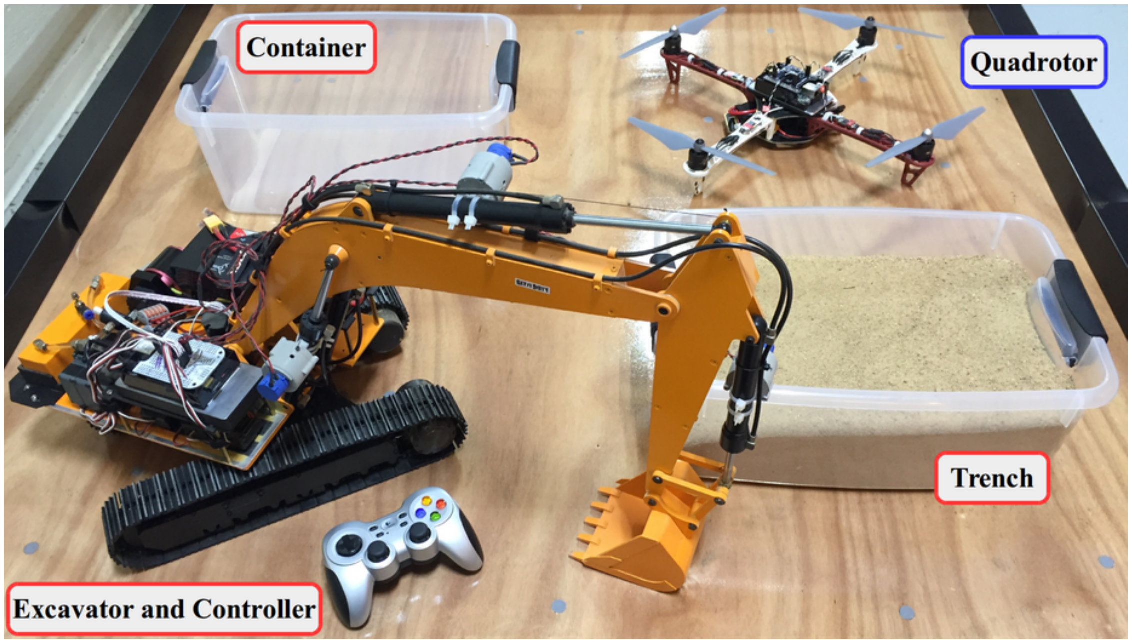
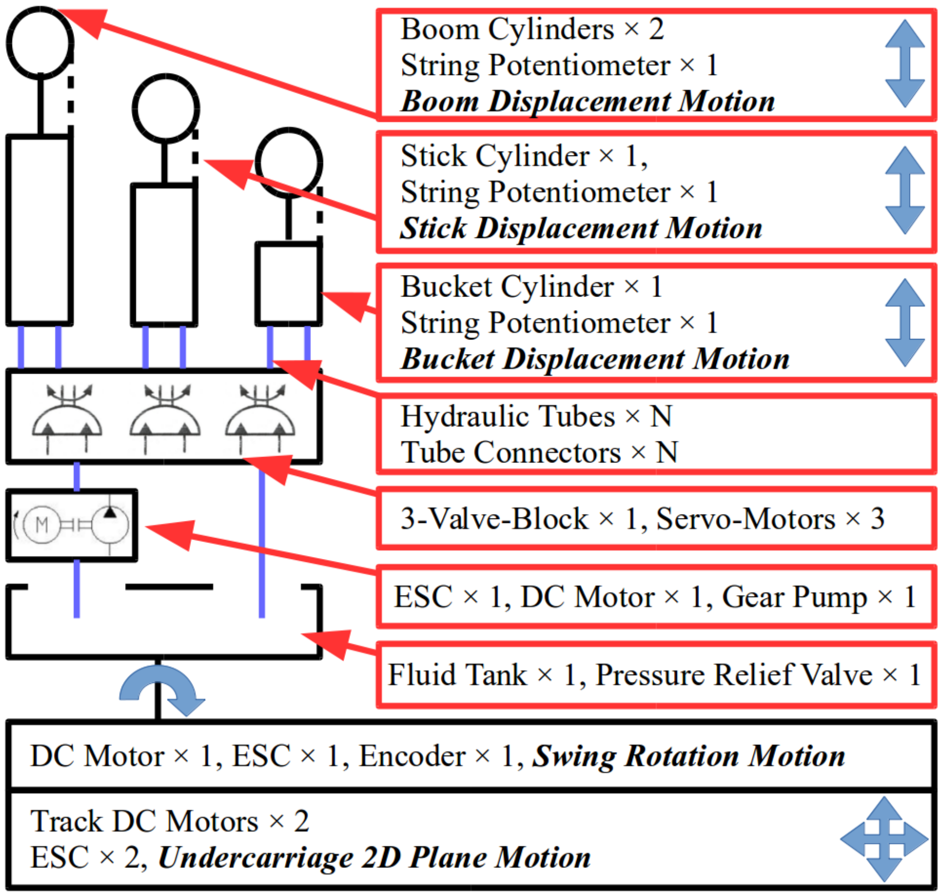


Figure 6.1: Hardware Platforms

6.1 Physical Platforms

To corroborate the developed methods, two physical platforms are employed for experiments. One is an excavator for a trenching-and-truck-loading task and the other is a quadrotor with a sequence of acrobatic flight motions. The hardware setups are provided in Fig. 6.1; the upper right corner has the quadrotor and the rest of the items are a 1/12th scaled hydraulic excavator, its controller, a container with sand and an empty container simulating the trench and a truck into which the sand picked up by the excavator bucket is dumped.



Boom Cylinders × 2
String Potentiometer × 1
Boom Displacement Motion
Stick Cylinder × 1,
String Potentiometer × 1
Stick Displacement Motion
Bucket Cylinder × 1
String Potentiometer × 1
Bucket Displacement Motion
Hydraulic Tubes × N
Tube Connectors × N
3-Valve-Block × 1, Servo-Motors × 3
ESC × 1, DC Motor × 1, Gear Pump × 1
Fluid Tank × 1, Pressure Relief Valve × 1
DC Motor × 1, ESC × 1, Encoder × 1, *Swing Rotation Motion*
Track DC Motors × 2
ESC × 2, *Undercarriage 2D Plane Motion*
M

Figure 6.2: Excavator Operation

A detailed diagram of the hydraulic and mechatronic systems of the excavator is shown in Fig. 6.2. In this setup, we placed 4 sensors to track the displacement of the three cylinders (for boom, stick, bucket), and rotation of motor (swing motion); thus, the excavator is considered with 4 DOF. The computer system consists of a laptop with Intel Core i7-6600U CPU@2.60GHz for heavy-duty computation, a Logitech F710 wireless controller with 2 standard joysticks interfacing with the operator, a Beaglebone Black with AM335x 1GHz ARM Cortex-A8 processor and GPIO pins for collecting sensor data and outputting actuation signals, a custom designed PCB board for breaking out signal connectors, and a WiFi router for communication. A coordinate frame assignment based on DH convention is shown in Fig. 6.3.

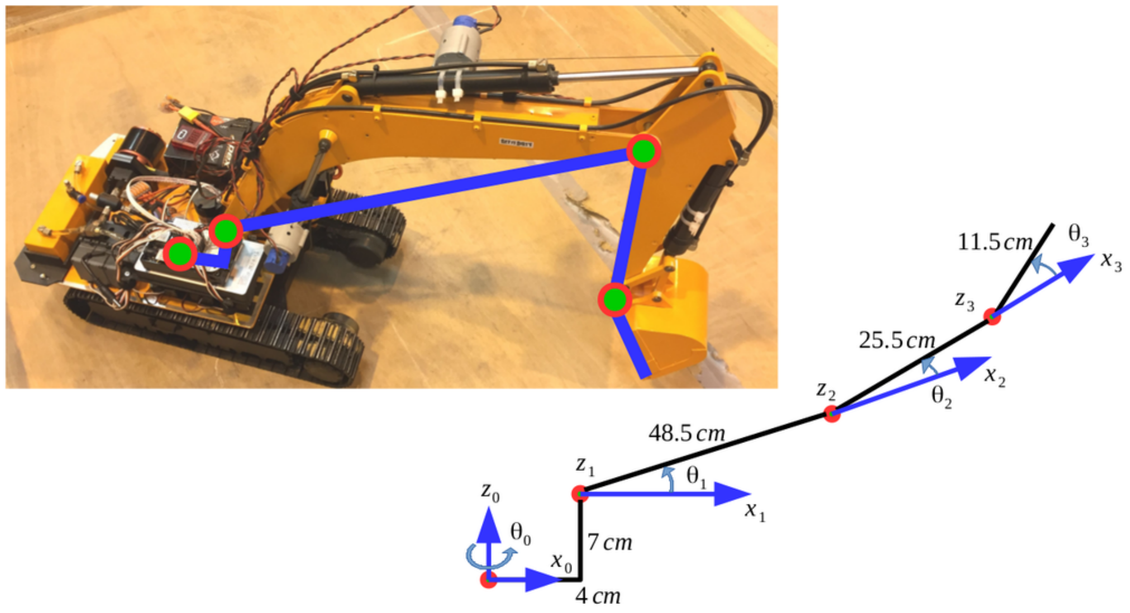


Figure 6.3: Coordinate Frame Assignment

The system software is implemented in Robot Operating System (ROS) [98] with different ROS nodes. A detailed ROS software network structure and control flow is provided in Fig. 6.4; this is the overall structure for supporting all the functions needed for the entire shared control process, including learning, prediction, blending, and adjusting. The nodes running on the laptop-side include joy reader (registering joystick input), reference generator, controller, blender, classifier, and predictor. On the Beaglebone-side, a state reader node and a driver node are employed for low-level sensing and actuation for the excavator. ROS topics, through which information are published and subscribed, are programmed to pass on computed information. Note that we have considered a simple Proportional-Integral (PI) controller for each excavator actuator to generate the automatic control subgoal tracking inputs.

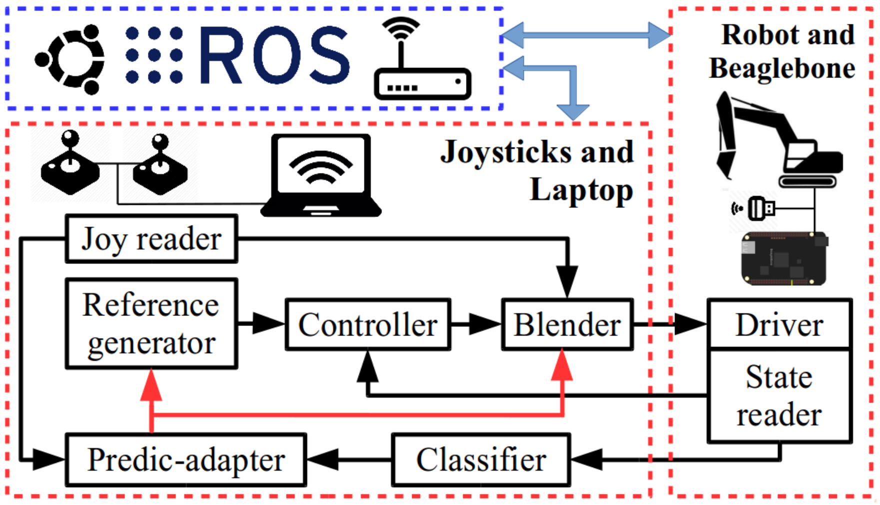


Figure 6.4: ROS-based Software Structure and Control Flow

In the quadrotor setup, sensors are employed to measure the rotation and translation of the quadrotor in a plane. A similar ROS software structure is employed to support different needs of the experiments including data recording and learning. We consider the quadrotor with 3 DOF since demonstration and sensing is designed to be carried out in a planar space.

6.2 Illustration of Task Demonstration and Learning Results

Two experiments are designed to test the developed algorithms for learning and prediction. The hydraulic excavator experiment is designed with an skilled operator demonstrating the trenching and truck-loading task for three cycles. The quadrotor flight demonstration consisting of having an operator hand-holding the quadrotor to fly through a diamond trajectory with flip motions. A visual illustration of such quadrotor demonstration process is provided in Fig. 6.5, where the human operator holds the quadrotor roughly going through the points marked with 1, 2, 3, 4 and back to 1 with a flip between 2 and 3, and an inverted flip between 3 and 4.

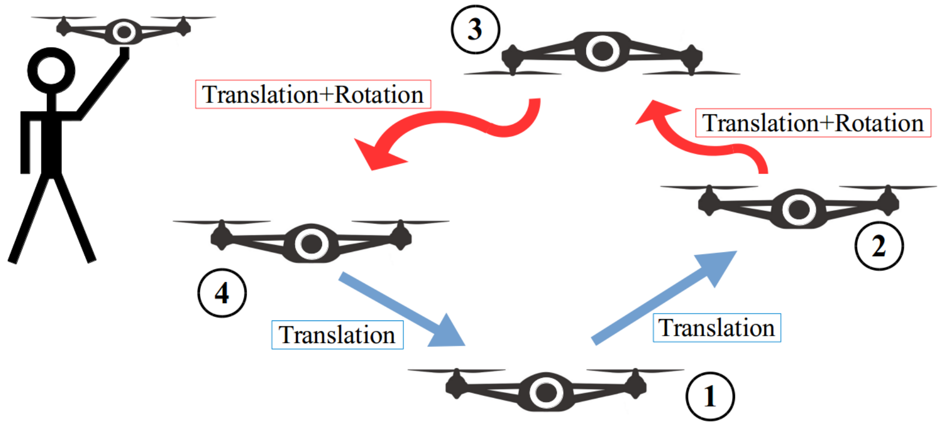


Figure 6.5: Illustration of Quadrotor Demonstration

Visualization of the distribution extraction and subgoal learning results for the excavator demonstration based on MCV method are shown in Fig. 6.6, where each four-line-combination represents a configuration of the 4 DOF excavator at a sample time. Subplot (a) provides the configuration

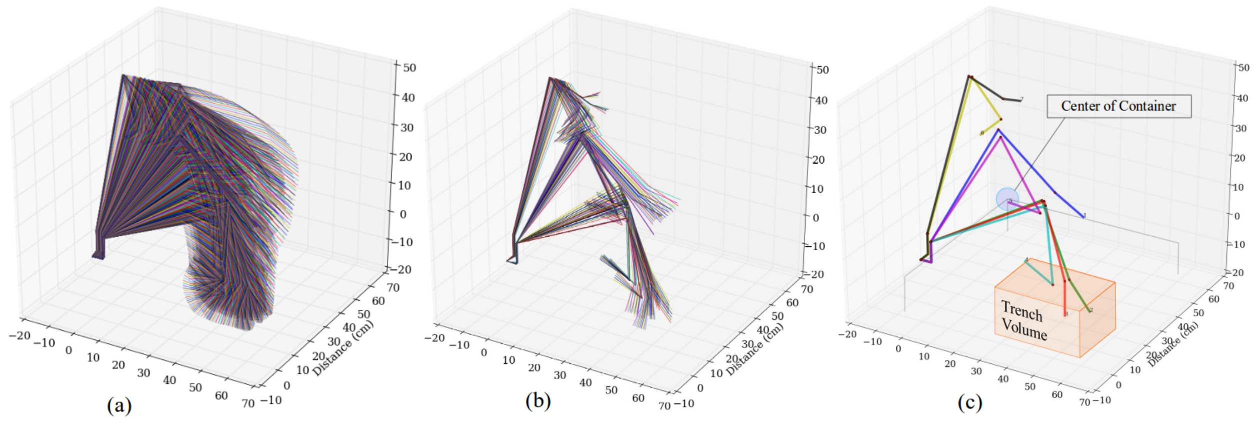


Figure 6.6: MCV Learning Process and Results of Excavator Demonstration

trajectory that excavator swept through during the entire demonstration, (b) provides the subgoal distributions after the data set is processed via the MCV method, and the learning results in the form of subgoals (each realized by a robot configuration) are provided in (c). To illustrate the advantage of MCV method on noise-sensitivity issues, the same demonstration data set is used with the threshold-based method in [15, 32]. Subgoal distributions extracted and learned subgoals are provided in Fig. 6.7 (a) and (b), respectively. One can observe that although the subgoals can still be learned properly, the subgoal distributions, i.e., data clusters in the mixture model, are not as tight as the ones resulting from the MCV method. Such discrepancy is likely a result of threshold crossing caused by noise, which would introduce statistical difficulties for BNPC inference in that (1) it would take more iterations for Gibbs sampler to converge and (2) the clustering would be more sensitive to the concentration hyper parameter of the CRP and, thus, error-prone.

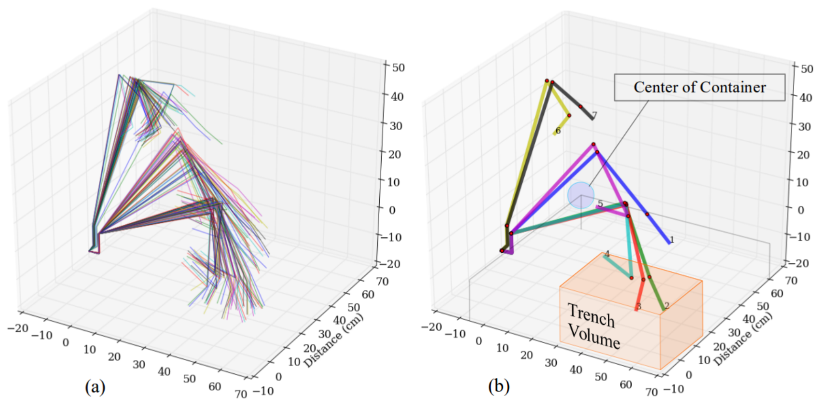


Figure 6.7: Learning Results via Primitives-based Methods

In the excavator demonstration, we have access to robot states as well as joystick input values. However, in some other engineering platforms, one may not have direct access to operator's command input values. The learning data set thus has to be collected via measuring the robot velocities that the operator's commands get mapped to, and the proposed MCV method is able to generalize to such scenarios. The quadrotor demonstration is one such case, where we only have access to robot states since the demonstration is executed with the operator hand-holding the quadrotor. The learning process and the results of the quadrotor demonstration are provided in Fig. 6.8, where (a) provides the complete trajectory of the robot, (b) provides the extracted subgoal distributions via MCV method, and (c) provides the final learned subgoals. As one can observe from the figure, the MCV method effectively captures the subgoal distributions for statistical inference leading to sub-

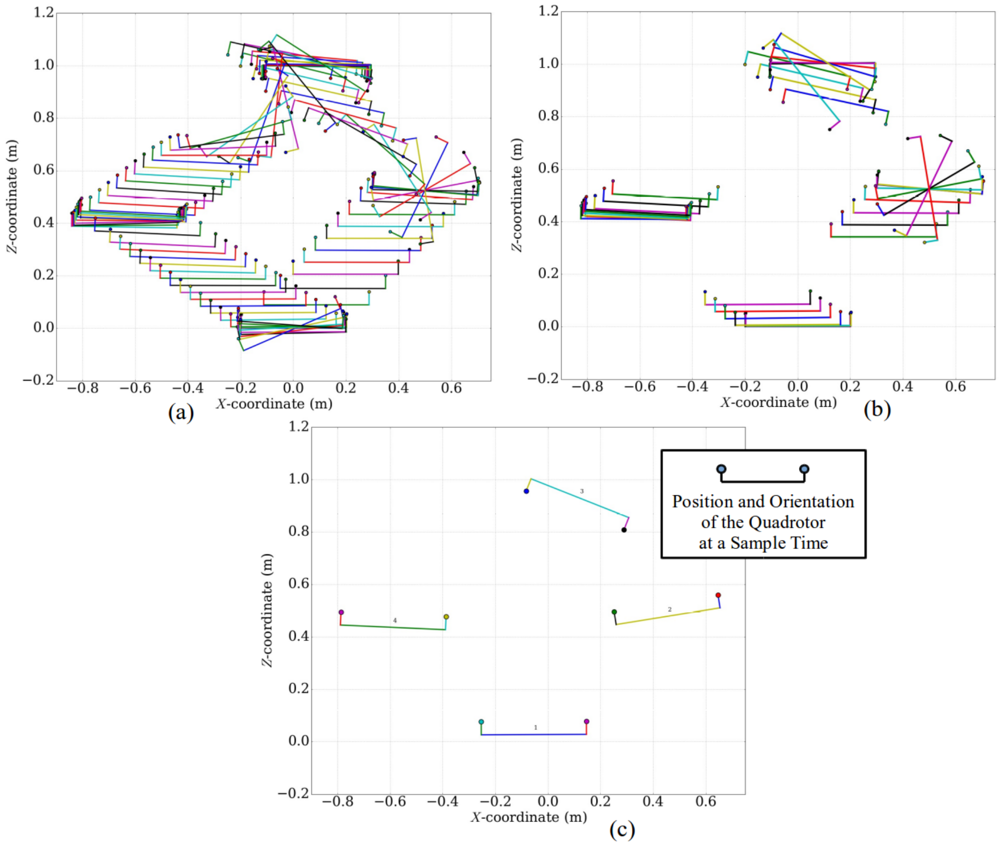


Figure 6.8: Learning Results from Quadrotor Flight Demonstration

goals which can serve as references for prediction and control assistance for SC purposes. Fig. 6.9 provides the quadrotor velocities (same data set was used in Sec. 2.2.2) along with the MCV values over time. The figure reveals the advantage of MCV method over threshold-based method on handling the command patterns that are hard to categorize through simple thresholds. In situations where setting thresholds does not lead to meaningful categorizations, the peak of $\mathcal{M}(t)$ at time zero and the following four peaks in the MCV curve provide intuitive indications of distinguishable command patterns occurring at the subgoals; this can be observed in Fig. 6.9 corresponding to points 1, 2, 3, 4 and back to 1 in Fig. 6.5. The extracted distributions also lead to proper subgoal learning results.

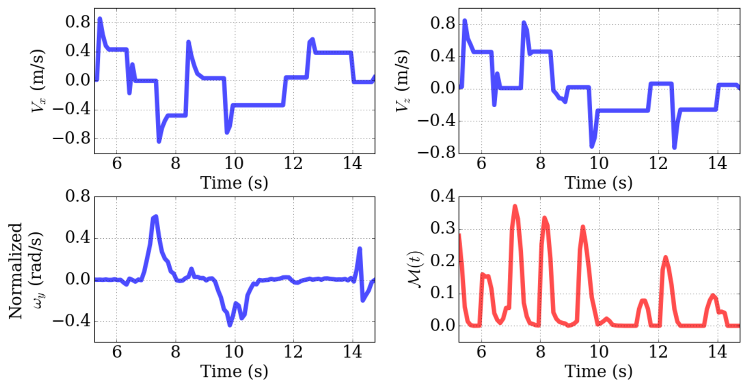
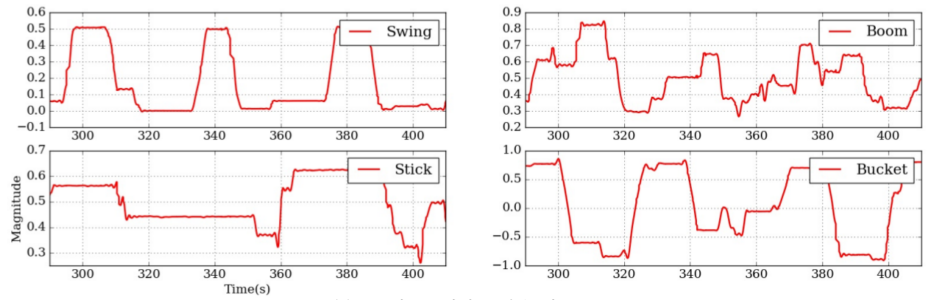


Figure 6.9: Quadrotor Velocities and MCV Value

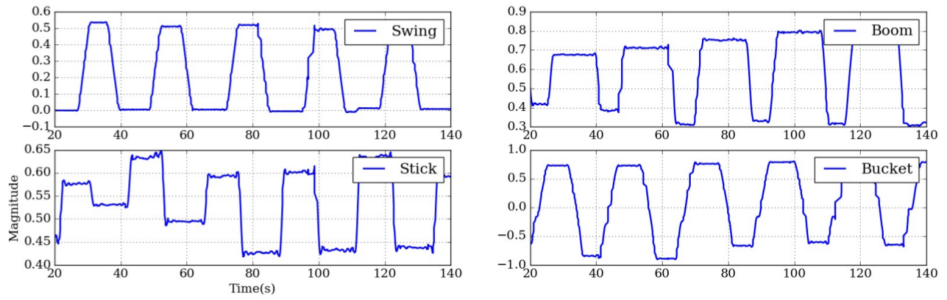
6.3 Integrated Results on Performance Improvement with BSC

Since the overall purpose of developing the BSC algorithms is to improve the performance for collaborative robotic tasks, an integrated BSC experiment is designed in the following manner. We invited 8 novice operators and 1 skilled operator to participate the excavator trenching and truck-loading task. The novice operators first familiarize themselves with the excavator operation with an operational instruction guide (which is always available to novice operators during the entire experiment). We call this stage the preparation stage (PSt). After this initial training and notification of getting ready for the task, novice operators are asked to execute the task. We call this stage the manual stage (MSt). Each task cycle is timed and the amount of sand each operator loaded to the truck is weighed. Then the skilled operator performs the same task as demonstration to the machine. The machine will learn from the skilled operator's demonstration and run the learning algorithms for task quantification as the foundation for providing intelligent assistance to the novice operators. We call this stage the demonstration stage (DSt). Lastly, the novice operators are asked again to execute the same task three cycles under the same condition, except for this time, there is assistance from our BSC algorithms. The process is timed and the amount of sand collected is weighed for each operator again. We call this the blended stage (BSt).

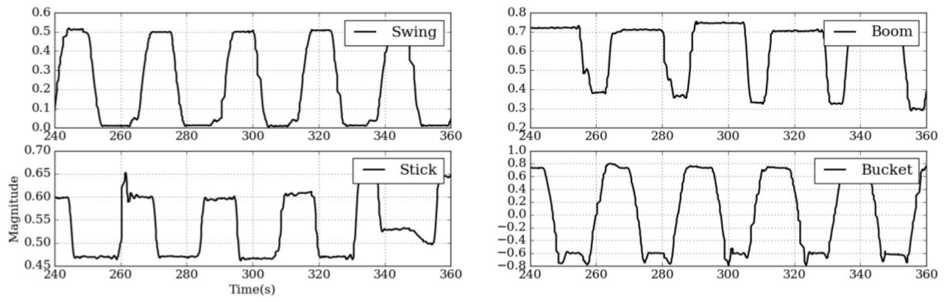
To illustrate the improvement with and without assistance, in Fig. 6.10, we present 3 sets of data which show normalized joint states of the excavator for the same task from a novice oper-



(a) Novice Trial wo/ Assistance



(b) Skilled Operator Demonstration Trial



(c) Novice Trial w/ Assistance

Figure 6.10: Excavator States from Different Operation Trials

ator without intelligent assistance (a), a skilled operator (b), and the same novice operator with intelligent control (c). Each trial is a repetition of the same task for the same duration of 120 s. From the figure, one can observe that the novice operator's improvement in performance with BSC assistance is more consistent between cycles and closely matches the states of the skilled operator.

Figure 6.11 shows the improvement in performance of the 8 novice operators due to assistance in terms of the average cycle time and the amount of sand moved per minute. On average, the cycle time improved from 63.17 s to 32.84 s and the sand moved per minute increased from 1.25 lb to 3.91 lb. This constitutes an improvement in cycle time of 52% and weight per minute improvement of 213%.

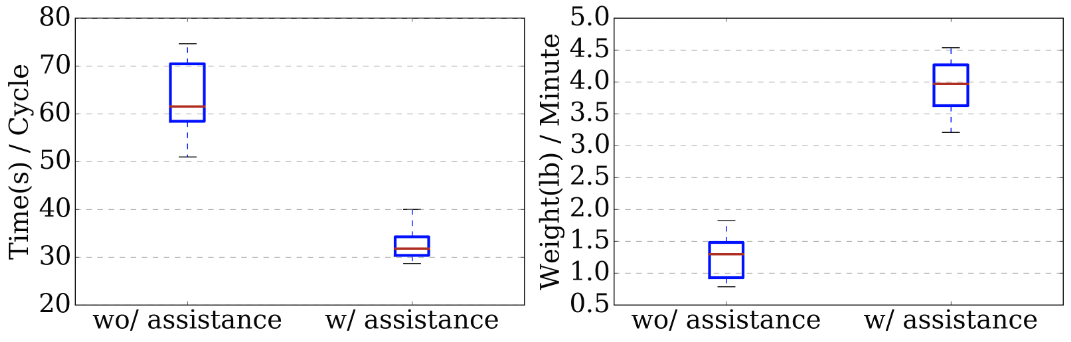


Figure 6.11: Box Plots of Productivity and Efficiency Improvement

6.4 Integrated Results on Subgoal Adjustment BSC

In order to test whether the subgoal adjustment algorithms are able to provide more flexibility to the BSC scheme and offer sustainable performance improvement, another series of experiments were designed in a similar manner with modifications. Based on a similar typical excavator trenching and truck-loading task, 10 Novice Operators (NO), 1 Skilled Operator (SO) were asked to participated in the experiments. The experiments were divided into various stages: preparation stage (PSt), manual stage (MSt), demonstration stage (DSt), BSC stage (BSC-St), and BSC with subgoal adjustment stage (BSC/SA-St).

In PSt, the basics of excavator operation and trenching and truck loading task are explained to the NOs. We provide them with excavator operational instruction guide (which is always available to novice operators), and have our NOs familiarize themselves with the excavator operations. After the notification from NOs of getting ready for the task, we move to MSt and ask the NO's to start executing the task for 3 cycles (each cycle consists of picking up a bucket of sand, moving the actuator to the location of the container and dumping into the container) without algorithm assistance. The process is timed and the amount of sand loaded to the truck is weighed. We then transition into DSt where we have the SO demonstrate the task, record the excavator states, and run the learning algorithm to learn the subgoals from the SO's demonstration. With the learned subgoals, we start the BSC-St and activate the BSC algorithm with intent prediction to provide active assistance. The NO's are asked to execute the same task for 9 cycles. The time and weight data are collected and analyzed for each of the 3 cycles. Finally in the BSC/SA-St, we repeat the process of BSC-St but provide active assistance with the the proposed subgoal adjustment BSC algorithm. The data are collected and analyzed in the same manner as in BSC-St.

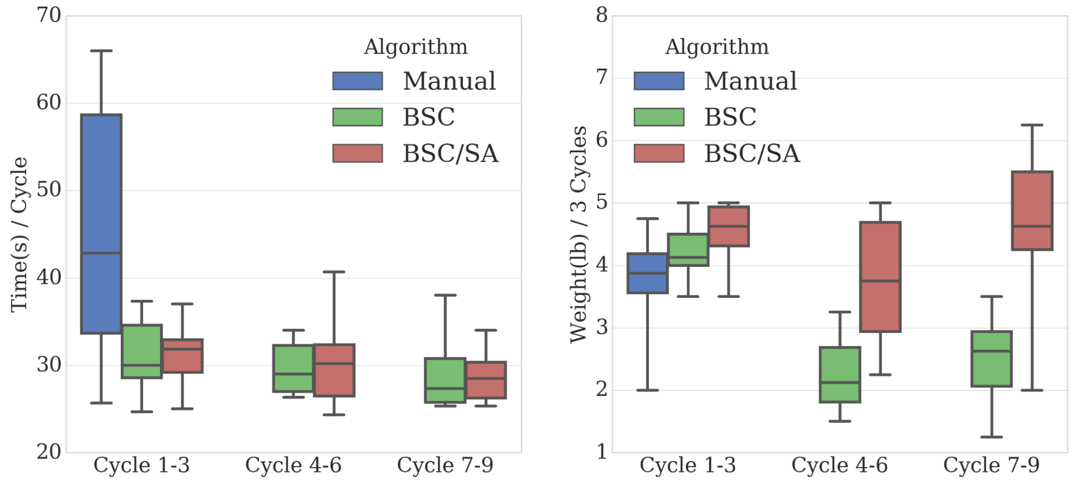


Figure 6.12: Performance Comparison for each 3 out of 9 Task Cycles

Figure 6.12 provides a comparison of the operator performance for each of the 3 task cycles. We observe that for the first 3-cycles when there is no significant shape change of the target object, the two BSC algorithms are all able to improve the operator performance from the manual operation without significant quality difference. On average, the cycle time improved from 45.53 s (MSt) to 31.10 s (BSC-St) and 31.13 s (BSC/SA-St), and the sand moved per 3 cycles improved from 3.78 lb (MSt) to 4.25 lb (BSC-St) and 4.55 lb (BSC/SA-St). For the last two 3-cycles, we observed that the time improvement from the two BSC schemes, with respect to manual operation, stays very similar to the first 3 cycles. However, the performance in terms of the amount of sand moved per 3-cycles is different for the two BSC schemes. For the intent prediction BSC, the amount of sand moved on average goes down from 4.25 lb (cycle 1-3) to 2.28 lb (cycle 4-6) and 2.56 lb (cycle 7-9), because of target object shape change versus nominal subgoals. However, for the subgoal adjustment BSC, the weight performance on average changes from 4.55 lb (cycle 1-3) to 3.75 lb (cycle 4-6) and 4.65 lb (cycle 7-9). The subgoal adjustment BSC maintains its assistance

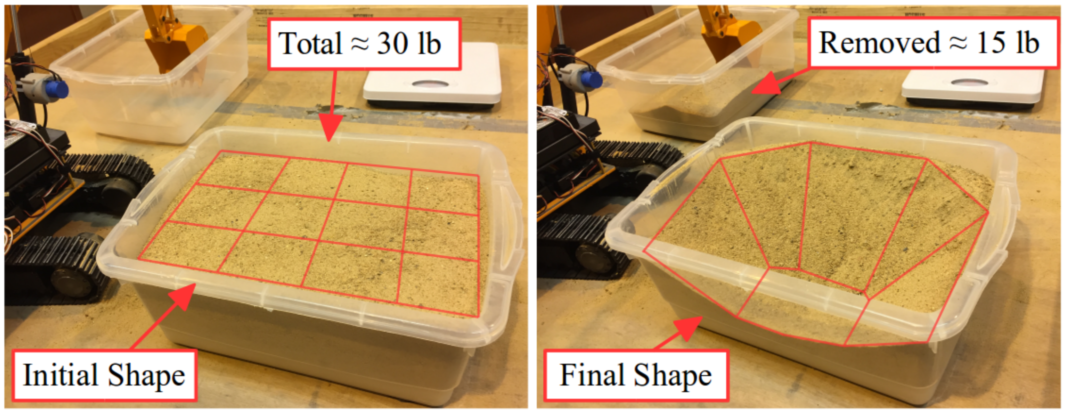


Figure 6.13: Shape Change of the Target Object after 9 Task Cycles

quality almost throughout in the presence of gradual but significant shape change of the target object as shown in Fig. 6.13. Furthermore, the recorded performance in the later cycles is still

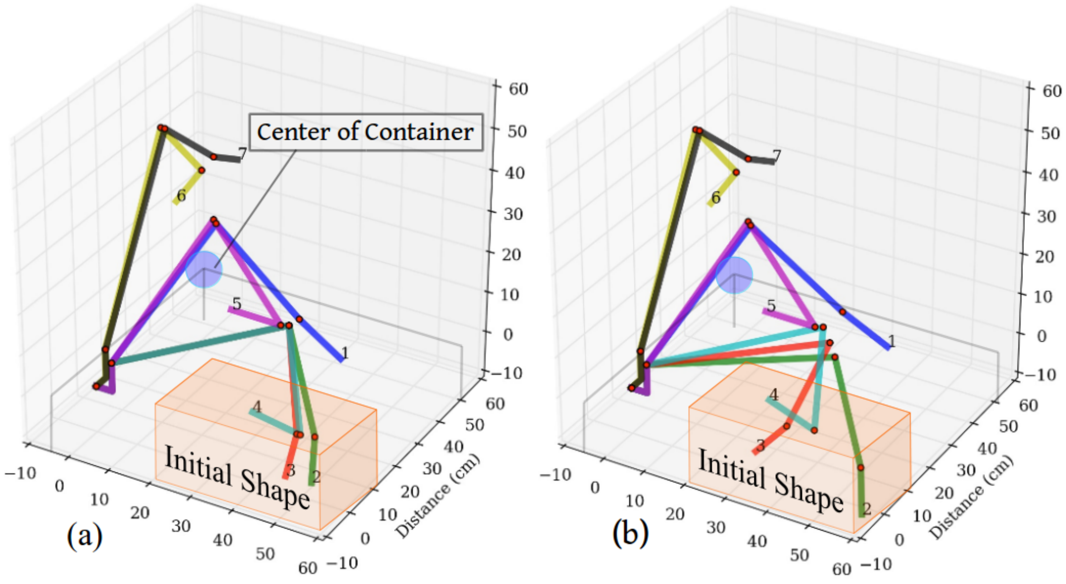


Figure 6.14: Visualization of Subgoal Adjustments and Initial Shape

better than manual operation for the first 3-cycles when the task operation is relatively easier. In addition, we observed that although the weight per 3-cycles data from the BSC/SA-St shows sustainable performance improvement, the data spread is relatively wide. One possible reason could be the fact that in our implementation, we fixed the value of the skill level parameter of SWAI since it is hard to evaluate one's skill level and operation style in our case. However, for industry applications, that problem could be solved by designing a standardized test to rate the skill level of operators and relate that to the selection of skill level parameter.

To illustrate the effectiveness of our algorithm from other perspectives, we present in Fig. 6.14 a visualization of subgoal adjustments made by the SO. In the visualization, each subgoal is realized by converting its position in robot joint space to its configuration space of the excavator through forward kinematics. In Fig. 6.14, (a) shows the original learned subgoals and the initial target shape, and the initial target shape is shown again along with the adjusted subgoals in (b) to emphasize the amount of adjustment.

6.5 Subgoal Transition Probability Learning Results

To corroborate the effectiveness of the subgoal transition probability learning method, another experiment is conducted. We first ask a skilled operator to demonstrate the same task again, due to a different operational style, 6 subgoals are learned as indicated by Fig. 6.15.

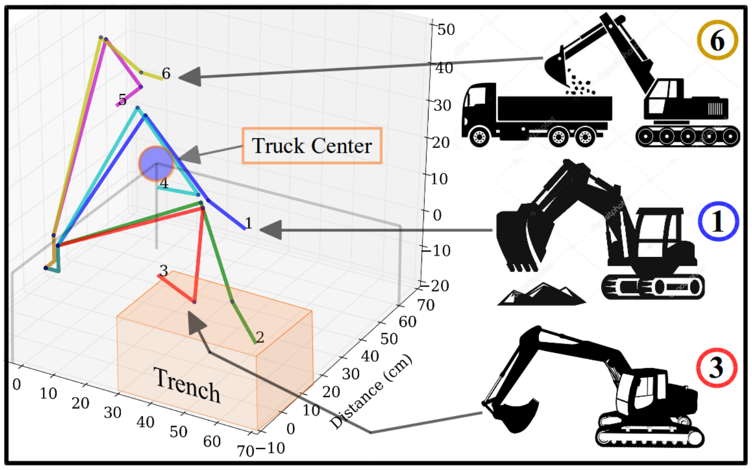


Figure 6.15: Six Learned Subgoals from a Different Demonstration

We then ask the operator to execute the task many cycles again. However, instead of asking the operator to execute the task only by going through the subgoals in their nominal sequence 1 through 6 repetitively, we also asked the operator sometimes to go back to subgoal 2 after visiting subgoal 3 and continue the nominal sequence. This is to simulate the following realistic situation seen very often in excavator trenching practice. Going from subgoal 2 to subgoal 3 is the typical subtask for scooping a bucket of dirt, however, due to the stiffness of the dirt in the trench, an operator often uses this scooping motion to loosen the dirt first, and then repeat the scooping motion again to pick up a full bucket of dirt to improve productivity. So, the operator in some cycles of task execution goes through subgoals in the nominal order of 1→2→3→4→5→6, some other cycles (less often) goes through them in the order of 1→2→3→2→3→4→5→6. Since there are six subgoals, we initialize the stochastic transition matrix T with a 6×6 matrix where every entry is equal to $1/6$, the convergence limit $Q = 0.5$, and we record the evolution T to corroborate the effectiveness of our method.

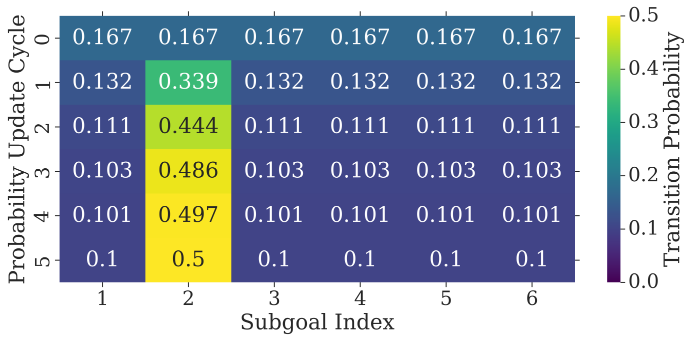


Figure 6.16: Evolution of the First Row of the Stochastic Matrix

The first five update cycles of the first row is provided in Fig. 6.16 from top to bottom, where the probabilities of each entry is given and the correlation between probabilities and colors is shown on the right for better visual interpretation. From Fig. 6.16, we observe that the second entry gets closer to $Q = 0.5$ each cycle and eventually converges, where the others decrease uniformly since the operator always visit subgoal 2 after visiting subgoal 1. The first nine update cycles of the third row is provided in Fig. 6.17 in the same fashion where we observe the following facts. At the 2nd

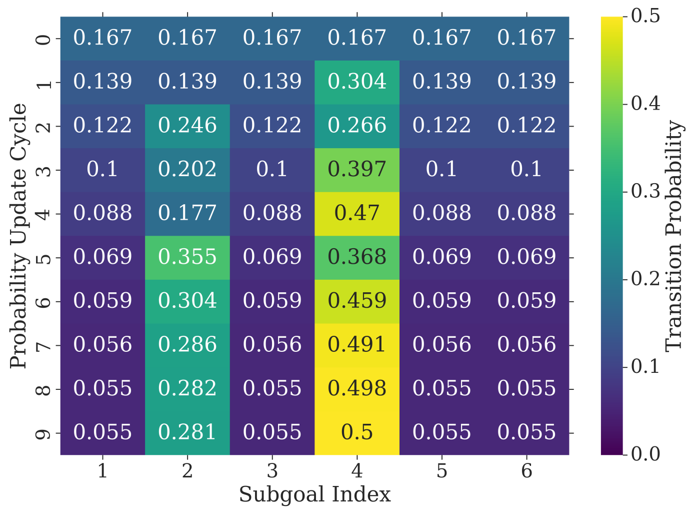


Figure 6.17: Evolution of the Third Row of the Stochastic Matrix

and 5th cycles, the operator visited subgoal 2 after 3, each resulting in an increase to the 2nd entry and a decrease to the 4th entry; for the rest of the cycles, the operator visited subgoal 3 and 4 in the nominal sequence. Eventually, the 4th entry converges to $Q = 0.5$ and the 2nd entry is left with a significant larger probability due to the fact that the evidence shows that subgoal transition from 3 to 2 happened a few times in addition to subgoal transition from 3 to 4. The final form of the 6×6 stochastic transition matrix is provided in Fig. 6.18 where each element of the 6×6 grid is the corresponding entry of the final stochastic transition matrix. From Fig. 6.18, one observes that for subgoal 1, 2, 4, 5, and 6, they all have a 0.5 probability of transitioning to the next subgoal. However, for subgoal 3, in additional to a 0.5 transition probability toward the next subgoal, it also has a 0.281 probability of transition to the previous subgoal. These results are expected from the proposed method, namely, it effectively updates the model towards convergence based on observed transitions with subgoal-visiting memory.

To illustrate the effectiveness of the intent prediction method from another perspective, the improvement in terms of probability increase over update cycles for the correctly predicted subgoals based on operator's action is provided in Fig. 6.19. Figure 6.19(a) provides the prediction probability samples of visiting subgoal 2 in situations when the last visited subgoal was subgoal 1, and

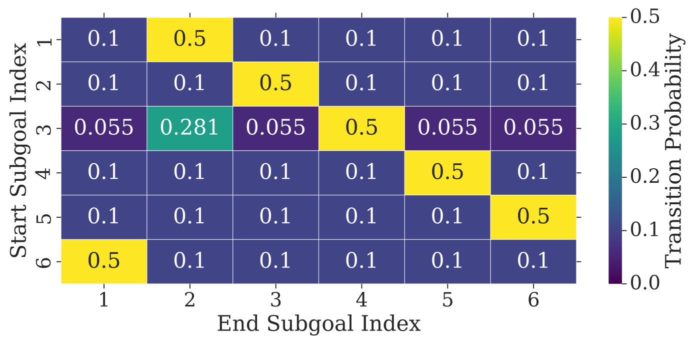


Figure 6.18: Final Form of the Stochastic Matrix

the operator takes actions intended for visiting subgoal 2. Figure 6.19(b) provides the prediction

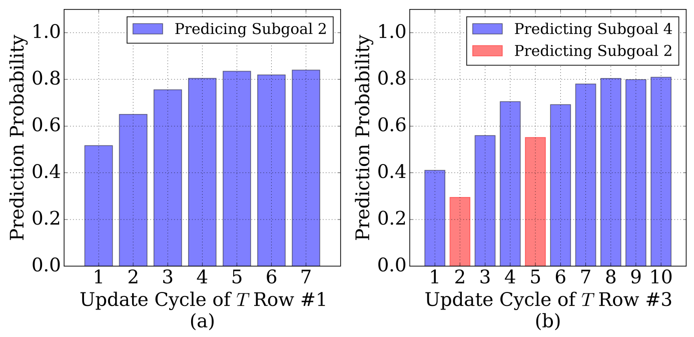


Figure 6.19: Prediction Probability Improvement over Update Cycles

probability samples of visiting subgoals 2 and 4, in red and blue, respectively, in situations when the last visited subgoal was 3, and the operator takes actions intended for visiting subgoals 2 and 4, respectively. In the provided scenarios, one observes that the probabilities of correct predictions increase and tend to stabilize around higher values with observed evidence and the corresponding model update. One can also verify the values corresponding to the first few update cycles from Fig. 6.19 with the evolution of the 1st and 3rd rows of matrix T provided in Fig. 6.16 and Fig. 6.17 and the update law given in Sec. 3.5.

7. COLLABORATIVE OPERATION OF ROBOTIC MANIPULATORS

Robot manipulators have been successfully employed in automating tasks in well-defined factory environments. However, in many circumstances, it is difficult for robot manipulators to execute tasks autonomously due to environmental uncertainties and sensing limitations. Providing human operators efficient and intuitive ways to operate robotic manipulators has the potential to help achieve tasks that are (1) dangerous and taxing for humans to carry out manually by themselves and (2) challenging for full robotic automation due to sensing limitations and environmental uncertainties.

Surface finishing operations, for example, are typically performed at the end of a product manufacturing cycle and are critical for the quality of many mechanical products. Many of these operations, such as sanding/grinding of aerospace structures and chamfering/deburring of cast and machined parts, are labor intensive and can expose human operators to hazardous conditions. These operations, however, have been performed mostly by human operators manually using handheld motorized tools due to cost, uncertainty handling, and sensing limitations for automation using robots [99, 100, 101]. Furthermore, surface finishing of free-form curved surfaces is difficult to automate because of irregular geometry with uncertainties as well as the problem of registering the part in the robot workspace.

In agriculture, timely inspection and sampling of crops is essential to detect and isolate plants with disease. Take cotton plants example, roughly 90% of the production is determined during the bloom and flowering stage. The inspection and sampling operations of cotton plants are currently performed by human crop consultants (HCCs). Executing these tasks in large farms is time-consuming and labor-intensive. It often takes HCCs weeks to inspect and sample only a small portion of a large farm for disease detection. It is envisioned that manipulators with cameras and grippers can be mounted on mobile platforms and HCCs can remotely command the robot arm to perform necessary operations for disease detection with active assistance provided by shared control. Many other applications (underwater construction, dangerous material handling in disaster response, etc.) also require similar object inspection and handling operations, and are still mostly done by humans. These tasks typically require observing a target object from different angles, then deciding an appropriate angle to take pictures, approach and collect samples for evaluation.

7.1 Existing Methods and Key Challenges

As mentioned earlier, employing remotely controlled robotic manipulators with human-machine shared control has the potential to deal with challenges that are often difficult to solve by employing methods that rely only on full autonomy. However, to design an effective robot manipulator system with human-machine shared control, one has to overcome several key challenges. How does one decompose and map the six degrees-of-freedom (DOF) motion of the end-effector into the input device such as a joystick or a general-purpose game controller? When holding a joystick-like device, the tendency of the human operator is to provide input (such as up, down, left, and right) with respect to a fixed reference that is associated with the image. How does one design the operational interface such that the sense of up, down, left, and right of the human input can be mapped to the robot end-effector motion, which does not require the human to think about a change of reference? Further, how should one design the operational interface such that the human operator's input can be quantitatively compared with the automatic control input for a certain action which will form the basis for human intent prediction and shared control?

Various joystick devices have been employed to facilitate human-controlled manipulator operations [102, 103], which allow operators to command end-effector motions in the robot base frame. This can be convenient for certain applications; however, it is challenging when a task requires moving/orienting a robot along some trajectory with respect to a target object. Touch-screen devices [104], motion capture systems [105], and master-slave mechanisms [106, 107] have also been employed to control robot manipulators. The obvious benefit is that they allow the human operator to intuitively specify the robot end-effector pose using body gestures or slave devices. However, fast and precise registration of operator's input via touch-screens or motion capture systems is difficult.

In the following, we first present two methods for the collaborative operation of a robotic manipulator. The first method requires the human operator to observe the robot and task object/environment from a third-person view while controlling the robot. It uses a mixed world and robot frame and employs a hybrid motion and force control to realize coordinated shared control. The second method requires the human operator to observe the object/environment from a first-person view through a camera mounted on the robot end-effector. It uses computer vision to detect objects and predict the human operator's intent, and shares the control authority between the human and vision-based closed-loop control to provide intelligent assistance. These methods can be applied to different applications and each of them has its own advantages. Different experiments are designed for each method and results are presented to corroborate the effectiveness of the methods.

7.2 Third-Person View Collaborative Operation with Hybrid Control

7.2.1 Operating Frame Construction and Joystick Interface Design

To extract features and frames from typical manual operations that can be applied to robot manipulators, we consider common characteristics in different applications that are visually illustrated in Fig. 7.1 (all figures in this chapter are provided at the end of the chapter). The frames $\{X_s, Y_s, Z_s\}, \{X_b, Y_b, Z_b\}$, and $\{X_N, Y_N, Z_N\}$ denote the world/space frame, end-effector/body frame, and normal frame (to the surface), respectively. We will use the notation $\{s\}$, $\{b\}$, and $\{N\}$ to denote these three frames for simplicity.

If one were to pick up the object shown on the inclined surface in Fig. 7.1 by the robot, a simple procedure for achieving this task consists of the following elements. Orienting the robot end-effector such that the frame $\{b\}$ is aligned with the frame $\{N\}$, whose orientation is easy to specify using frame $\{s\}$ as reference from a human's perspective, then moving the end-effector in its own frame $\{b\}$ to adjust and approach. Another example is the motion on a curved surface shown in Fig. 7.1: if part of a surface finishing operation requires the end-effector to travel along the surface trajectory (highlighted in orange color) with a desired contact force, an effective and intuitive way can be described as follows. First, orienting the end-effector according to frame $\{N\}$

on the left, aligning Z_b with Z_N, and moving along Z_b until a desired contact force is achieved. Then, while maintaining contact, to travel along the trajectory, one simply needs to move along the Y_b while, at the each instantaneous moment, orienting frame $\{b\}$ such that Z_b is always normal to the surface and Y_b is pointing to the travel direction.

Because of the aforementioned features of this approach, we refer to it as the Instantaneous Surface Normal Approach (ISNA). To apply it in the context of robot manipulator operations, one can consider the following elements. Assuming that the human operator has a desired trajectory based on domain knowledge, then the following aspects are easy to observe and specify from the operator's perspective: the orientation of frame $\{b\}$ in frame $\{s\}$ which is the reference frame, and the translation direction in frame $\{b\}$. From the perspective of a robot manipulator, if a reference contact force is desired along one of its $\{b\}$ frame axes, then tracking this reference force is straightforward with a closed-loop controller. Next, we consider the interface design of a general purpose joystick such that these instructions can be delivered to a robot manipulator effectively by a human operator.

Among the commercially available general purpose joysticks, most of them provide the following common features which are employed in our design to encode manipulator operation commands. A 3-axis stick which provides 3 continuous states; employed to encode reference orientation of $\{b\}$ in $\{s\}$. A 2-axis navigator button where each axis provides 3 discrete states such as forward/backward/stop; employed to command translation on the X_b-Y_b plane. A slider which provides a continuous state; employed to encode the magnitude of translation velocity. A trigger and a few programming buttons with each providing one discrete on/off state; can be employed to trigger force control on Z_b-axis or command end-effector to move along Z_b-axis and grip. Some examples of commercial joysticks are provided in Fig. 7.2, including models by PXN, Thrustmaster, Logitech from left to right. We also denote the frame of the 3D-stick with $\{X_j, Y_j, Z_j\}$, or $\{j\}$ for simplicity.

Such a design would allow a human operator to command a robot end-effector to travel along any trajectory on a curved surface or execute similar operations with a general purpose joystick effectively and intuitively. The effectiveness comes from the fact that human operator's joystick motions are closely mapped to the motions of robot end-effector from a geometric perspective. The process, with human-machine collaboration in the form of coordinated shared control, occurs in the following manner. Human operator, based on visual observation of robot and target object, provides: (1) desired orientation via the stick in $\{s\}$ frame, (2) translation direction via navigator and its velocity via slider in $\{b\}$ frame, and (3) indication of making contact with desired force or moving/gripping via trigger or on/off buttons in $\{b\}$ frame. The human signals are combined with the measured robot states and contact force to generate error terms for automatic control. The automatic control, then, coordinates the motion of individual robot joints to achieve end-effector motion specified by the human operator using the intuitive ISNA.

7.2.2 Design and Implementation of a Hybrid Control Law in the Mixed Frame

To realize the command provided by human operator described in Section 7.2.1, we require a hybrid control law capable of tracking mixed reference input consisting of orientation (position) in frame $\{s\}$ and translation (velocity) and force in frame $\{b\}$. In formulating such a hybrid control law, we employ the concepts of screw-axis and robot kinematics based on the Product of Exponential (PoE) formulas [108, 109]. The main reason for this selection is that these tools provide easy access to space and body Jacobians, in contrast to the traditional analytical Jacobian, which are needed in operating the robot using ISNA. We also design the control law based on the assumption that the robot allows users to command the velocity of each joint independently which is guaranteed by its low-level controllers. In the following we will describe, from the perspectives of generating the control input, the following aspects: (1) orientation error calculation, (2) construction of constant end-effector velocity, and (3) integration of force with orientation and velocity to formulate a single hybrid control law.

7.2.2.1 Orientation Error

Since the 3-axis stick has three rotation axes, as introduced in Sec. 7.2.1, we use these three rotation angles to provide orientation reference for the end-effector. We denote the three rotation angles provided by the operator as ψ, θ, ϕ for the joystick axes Z_j, Y_j, X_j provided in Fig. 7.2. Note that when the joystick is in its neutral position, its frame $\{X_j, Y_j, Z_j\}$ is aligned with robot space frame $\{X_s, Y_s, Z_s\}$ and one can represent the orientation of $\{X_j, Y_j, Z_j\}$ with respect to $\{X_s, Y_s, Z_s\}$ by

$$\begin{aligned} R_d &= \underbrace{R_Z(\psi)}_{\text{Yaw}} \times \underbrace{R_Y(\theta)}_{\text{Pitch}} \times \underbrace{R_X(\phi)}_{\text{Roll}} \\ &= \begin{bmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - c_\phi s_\psi & s_\psi s_\phi + c_\psi c_\phi s_\theta \\ c_\theta s_\psi & c_\psi c_\phi + s_\psi s_\theta s_\phi & c_\phi s_\psi s_\theta - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \end{aligned} \tag{7.1}$$

where $R_d \in SO(3)$ is the rotation matrix which contains the desired orientation and c_q, s_q denote $\cos q, \sin q$, respectively, for $q = \psi, \theta, \phi,$. The measured orientation of the robot end-effector $R \in SO(3)$ with respect to its space frame can be extracted from

$$T(\mathbf{q}) = \begin{bmatrix} R & \mathbf{L} \\ \mathbf{0} & 1 \end{bmatrix} \tag{7.2}$$

where $\mathbf{q} \in \mathbb{R}^n$ is a vector of joint angles $q_i \in \mathbb{R}$ for $i = 1, 2, \cdots, n$ of an n-DOF robot, and $\mathbf{L} \in \mathbb{R}^3$ is the position of end-effector in the space frame. $T(\mathbf{q}) \in SE(3)$ is the homogeneous transformation of robot forward kinematics which can be computed based on the PoE method by

$$T = Me^{[\beta_1]q_1 \cdots [\beta_n]q_n} \tag{7.3}$$

where $[\beta_i] \in se(3)$ is the matrix representation of the i^{th} screw axis $\beta_i \in \mathbb{R}^6$ expressed in the body frame. $M \in SE(3)$ encodes the end-effector configuration in the space frame when all q_i's are zero. Realizations of M, $[\beta_i]$, β_i are given in (7.17), (7.18), and (7.19), respectively, with a particular robot used for the experiments.

The orientation error vector $\Omega_e \in \mathbb{R}^3$, then, can be calculated using

$$\Omega_e = \mathcal{F}[\log(R^T R_d)] \tag{7.4}$$

where $\log(R^T R_d) \in so(3)$ is the matrix logarithm of the matrix representation of orientation error, and $\mathcal{F}(\cdot) : so(3) \mapsto \mathbb{R}^3$ is a mapping which takes a 3×3 skew-symmetric matrix to its vector form. Note that the orientation error Ω_e is generated by (1) human operator providing reference orientation R_d and (2) robot providing measured end-effector orientation R. This will be used in the hybrid control law to make R follow R_d.

7.2.2.2 Constant Translation Velocity with Adjustable Magnitude

When moving the robot end-effector, a constant translation velocity is desired. For example, in surface finishing operations, the time a finishing tool stays on a surface is proportional to the material it removes while a constant contact force is maintained. Note that the operator's input for translation is provided by the 2-axis navigator which provides two discrete signals that can be encoded into a vector $v = [x, y]$ where $x, y \in \{-1, 0, 1\}$. Then, to generate a constant translation velocity $V_c \in \mathbb{R}^2$ with its direction indicated by $[x, y]$, one can normalize the input vector v using

$$V_c = \alpha \left[\frac{x}{||v||}, \frac{y}{||v||} \right]^T \tag{7.5}$$

where the magnitude of V_c is specified by relating the scaling parameter $\alpha \in \mathbb{R}^+$ with the joystick slider.

To find the corresponding robot joint velocities for accomplishing V_c, the following kinematics mapping exists:

$$\dot{\mathbf{q}} = J_b(\mathbf{q})^\dagger \begin{bmatrix} 0 & 0 & 0 & V_c^T & 0 \end{bmatrix}^T \tag{7.6}$$

where $J_b(\mathbf{q})^\dagger \in \mathbb{R}^{n \times 6}$ is the pseudo-inverse of the body Jacobian (a key element for the hybrid control law, instead of space or analytical Jacobians) for an n-joint robot represented in the $\{b\}$

frame. Following the PoE formulation, the body Jacobian $J_b(\mathbf{q})$ can be obtained by

$$J_b(\mathbf{q}) = \begin{bmatrix} J_{b1} & J_{b2} & \cdots & J_{bn} \end{bmatrix} \tag{7.7}$$

where J_{bi} for $i = 1, 2, \cdots, n$ are columns of the matrix $J_b(\mathbf{q})$ and are given by

$$J_{bi} = \mathrm{Ad}_{e^{-[\beta_n]q_n} e^{-[\beta_{n-1}]q_{n-1}} \cdots e^{-[\beta_{i+1}]q_{i+1}}}(\beta_i) \tag{7.8}$$

for $i = n-1, n-2, \cdots, 1$ with $J_{bn} = \beta_n$, where $e^{(\cdot)} : se(3) \mapsto SE(3)$ is the matrix exponential function and $\mathrm{Ad}_{(\cdot)} : SE(3) \mapsto \mathbb{R}^{6\times 6}$ is the adjoint operator of a homogeneous transformation. Taking $T(\mathbf{q})$ in (7.2) as an operand, the result of the adjoint mapping is simply

$$\mathrm{Ad}_{T(\mathbf{q})} = \begin{bmatrix} R & \mathbf{0} \\ \mathbf{L}R & R \end{bmatrix} \tag{7.9}$$

7.2.2.3 Hybrid Control Law

With a robot capable of measuring the contact force F along the Z_b-axis, and the desired reference force F_d for a specific application, we can define the force error as

$$F_e = F_d - F \tag{7.10}$$

Note that $F_e \in \mathbb{R}$ from (7.10) is represented in the Z_b-axis and $V_c \in \mathbb{R}^2$ from (7.5) is represented in X_b and Y_b axes. Note also that they, along with $\Omega_e \in \mathbb{R}^3$ from (7.4), are decoupled in the $\{b\}$ frame. Therefore, at any given time $t \in \mathbb{R}^+$, one can construct a control input term $H_e(t) \in \mathbb{R}^6$ by combining those decoupled elements in the form of

$$H_e(t) = \begin{bmatrix} \Omega_e & V_c & F_e \end{bmatrix}^T \tag{7.11}$$

The hybrid control law allowing the robot to follow human operator's command input using ISNA is then given by

$$\dot{\mathbf{q}} = J_b(\mathbf{q})^\dagger \left[K_P H_e(t) + K_I \int_0^t H_e(t)dt \right] \tag{7.12}$$

where $\dot{\mathbf{q}} \in \mathbb{R}^6$ is the final output to the robot in the form of individual joint velocities.

To ensure that the hybrid control law would result in the end-effector following the command provided by the human operator via joystick in the fashion discussed in Section 7.2.1. The control parameters in (7.12) are designed as follows. $K_P \in \mathbb{R}^{6\times 6}$ is given by

$$K_P = \mathrm{diag}(p_\phi, p_\theta, p_\psi, 1, 1, p_f) \tag{7.13}$$

and $K_I \in \mathbb{R}^{6\times 6}$ is given by

$$K_I = \mathrm{diag}(i_\phi, i_\theta, i_\psi, 0, 0, i_f) \tag{7.14}$$

where scalars $p_\phi, p_\theta, p_\psi, p_f$ and $i_\phi, i_\theta, i_\psi, i_f$ are proportional and integral gains for orientation and force terms, respectively.

The hybrid control law in (7.12) with gains provided in (7.13) and (7.14) will result in end-effector orientation and contact force converging to their references given in frame $\{s\}$ and the instantaneous $\{b\}$ frame along Z_b-axis, respectively, while moving along the reference direction with a constant speed in the X_b-Y_b-plane of the instantaneous $\{b\}$ frame. The speed of convergence will depend upon individual parameters selection and robot physical capabilities. However, the effectiveness of the hybrid control law in (7.12) can be understood by observing that the design of matrices K_P and K_I which essentially combines into one equation decoupled entries from: (1) the velocity kinematics

$$\dot{q} = J_b(q)^\dagger V_b \tag{7.15}$$

for end-effector translation motion with velocity $V_b \in \mathbb{R}^6$ in the $\{b\}$ frame; and (2) the task-space motion control law

$$\dot{q} = J_b(q)^\dagger \left[K_P X_e(t) + K_I \int_0^t X_e(t)dt \right] \tag{7.16}$$

for end-effector position control with $X_e \in \mathbb{R}^6$ encoding the difference between desired and measured orientation and position. The effectiveness and analysis of (7.15) and (7.16) can be found in any PoE-based robotics references such as [108, 109]. The crucial observation is that both (7.15) and (7.16) use the body Jacobian to map end-effector motion to joint velocities. The construction of (7.11) essentially decouples space frame orientation, body frame translation and force and makes the combined term applicable to (7.12) while maintaining the relationships in (7.15) and (7.16) by the design of (7.13) and (7.14).

7.2.2.4 Kinematics Setup for the Proposed Joystick Interface

We employ a UR5 robot, a 6-DOF robot with 6 revolute joints, as an example to illustrate a kinematics setup which will facilitate intuitive human operation and the hybrid control law implementation. However, the procedure is applicable to any open-chain robots. The same robot is also used later for conducting experiments. The UR5 robot is provided in Fig. 7.3 with all joints at their zero configuration. In addition to the same notation used in frames $\{s\}$ and $\{b\}$, we use $s_1, s_2, \cdots, s_6 \in \mathbb{R}^3$ to denote the joint rotation axes and $H_1, H_2, W_1, W_2 \in \mathbb{R}^+$ to denote lengths.

The necessary geometric information for calculating the robot kinematics is summarized in Table 7.1 where the expression of each joint axis s_i and a selected point a_i on the axis are provided as vectors in the robot body frame. Thus, the M matrix in (7.3) is given by

$$M = \begin{bmatrix} 1 & 0 & 0 & -(L_1 + L_2) \\ 0 & 0 & 1 & -(W_1 + W_2) \\ 0 & -1 & 0 & H_1 - H_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.17}$$

and the i^{th} body screw axis in its matrix form can be computed by

$$[\beta_i] = \begin{bmatrix} \mathcal{F}^{-1}(s_i) & -s_i \times a_i \\ \mathbf{0} & 0 \end{bmatrix} \tag{7.18}$$

where $\mathcal{F}^{-1}(\cdot)$ is the inverse mapping of $\mathcal{F}(\cdot)$ in (7.4). The corresponding vector form of the same screw axis as in (7.18) is then given by

$$\beta_i = \begin{bmatrix} s_i \\ -s_i \times a_i \end{bmatrix} \tag{7.19}$$

Table 7.1: Robot Geometry

Joint axis s_i in $\{X_b, Y_b, Z_b\}$		Selected point a_i on s_i in $\{X_b, Y_b, Z_b\}$	
s_1	$(0, -1, 0)$	a_1	$(L_1 + L_2, 0, W_1 + W_2)$
s_2	$(0, 0, -1)$	a_2	$(L_1 + L_2, -H_2, 0)$
s_3	$(0, 0, -1)$	a_3	$(L_2, -H_2, 0)$
s_4	$(0, 0, -1)$	a_4	$(0, -H_2, 0)$
s_5	$(0, 1, 0)$	a_5	$(0, 0, W_2)$
s_6	$(0, 0, -1)$	a_6	$(0, 0, 0)$

7.2.3 Experiments and Results

To test the effectiveness of the developed method, two experiments are conducted simulating surface tracking and pick-and-place type applications. A physical platform is employed for both experiments which includes a 6-DOF UR5 robot, a Logitech Freedom 2.4 GHz wireless joystick, an ATI AXIS-80 force/torque sensor, experiment-specific end-effectors and operation platforms. The UR5 allows a user to command its joint velocities independently, and provides feedback measurements of its joint angles and velocities in real-time. The ATI force/torque sensor provides force and torque measurements along its x, y, z axes.

7.2.3.1 Surface Tracking Experiments

The setup for the first experiment is shown in Figure 7.4. A 3D printed end-effector equipped with a marker is attached to the robot. The experiment requires a human operator commanding the robot end-effector to draw an L-shaped line inside the constrained area indicated by the black tapes on the curved surface shown in Fig. 7.5(a). A detailed illustration of the end-effector is provided in Fig. 7.5(b), where we have designed sharp tips for the fixture (as shown) as an indication of x-y translation directions to the human operator.

During the line-drawing operation, we also require the human operator to keep the marker normal to the curved surface by properly orienting the end-effector via joystick. We record the operational process, specifically, the force measured by the ATI sensor along the z-direction of the robot body frame and the configuration of the robot during the process. Note that the geometry of the curved surface is unknown to the robot and is not measured by any sensor except through the observation of human operator which the operator uses to orient the end-effector via the joystick.

Sampled robot configurations, in temporal order indicated by numbers, during the operation are presented in Fig. 7.6. From the figure, one observes that the robot end-effector gets aligned perpendicular to the curved surface which corresponds to the orientation commanded by the human operator via joystick. Fig. 7.7(a) provides a plot of the force measurement along the end-effector z-axis versus time, where the desired force level is set to 10 N. Results from the force plot indicate that the contact force is regulated to within ± 1 N range, indicated by red and green dashed lines. The L-shaped line drawn by the robot is shown in Fig. 7.7(b).

7.2.3.2 Material Handling Experiments

The setup for the second experiment is shown in Fig. 7.8 where the end-effector is equipped with a block, and there are two 3D printed structures with the corresponding cavities to fit the block. This experiment requires the human operator to command the robot to move the block from one platform to another. The cavities are designed such that the block can only fit when it is correctly oriented with respect to the opening space. Similar to the previous experiment, there are no additional sensors except human observation.

Sampled robot configurations, in temporal order indicated sequentially by numbers, during the operation are presented in Fig. 7.9. The robot is shown to be picking up the block from the left platform and dropping the block in the right platform. Again, one observes that the orientation of the end-effector effectively follows the operator's input via joystick as shown in the lower-left of Fig. 7.9. Figure 7.10 provides a wire-frame diagram of robot configurations during the whole process: subplot (a) shows the picking up process where the robot picks up the object from the left platform and moves to a neutral position; subplot (b) shows the placing process where the robot moves from the neutral position and places the block in the right platform. The wire-frame color in each of Fig. 7.10 (a) and (b) is shaded from light to dark to highlight passage of time as the operation progresses.

We also recorded the controller performance by tracking end-effector orientation with respect to the reference input provided by the joystick. Fig. 7.11 shows the plots of robot performance against roll, pitch and yaw reference inputs provided via the joystick. We can observe that the end-effector tracks the desired inputs effectively.

7.3 First-Person View Collaborative Operation with Vision and Intent Prediction

There are certain applications where the previous method may not work well, or the human operator may find it difficult to operate. We illustrate these additional challenges and propose method to address them in this section.

7.3.1 Robot System and Operational Interface

To illustrate challenges of remote robot operation for some other applications where the previous method may not work well, consider another example task shown in Figure 7.12 of a robot end-effector inspecting an object (a trophy in this case) around its X_o axis and in its Y_o-Z_o plane, and grabbing it along its Z_o axis. We first discuss human operation based on the fixed world camera followed by the camera mounted on the end-effector. If the operator looks through the world camera and provides commands in the robot base frame $\{X_s, Y_s, Z_s\}$, the end-effector motion directions are relatively straightforward to specify; however, executing an end-effector motion trajectory around the trophy is difficult because the object frame $\{X_o, Y_o, Z_o\}$ is not aligned with the robot base frame. One can also consider the operator looking through the world camera and providing commands in the end-effector frame $\{X_b, Y_b, Z_b\}$, where the joystick left-right and up-down axes correspond to end-effector X_b and Y_b motions, respectively. This also creates a problem because the human operator tends to have a fixed reference for up-down-left-right with respect to the input device as shown in Fig. 7.13(a), the corresponding X_b, Y_b motions are not fixed in the world camera's view and are difficult to identify.

To overcome the above issues of employing a fixed world camera, we propose an operational interface designed with a robot camera (also shown in Fig. 7.12) mounted at the end-effector center, where the horizontal and vertical directions of the camera's video images are aligned with the end-effector X_b, Y_b axes, respectively. We employ a general purpose game controller with two joysticks (each with two DOF) and several command buttons as shown in Fig. 7.14, and map the 6-DOF end-effector motions as provided in Table 7.2. This allows for intuitive mapping of left and right joystick up-down-left-right to translating and rotating the end-effector up-down-left-right with respect to an object of reference provided by the camera images. This task can be simplified into the following steps. (1) Rotate along the X_b axis such that the X_o-Y_o plane is parallel with the X_b, Y_b plane as shown in Fig. 7.13(b). (2) Move along the X_b, Y_b axes to center the object as shown in Fig. 7.13(c). (3) Rotate along the Z_b axis to correct the object orientation in the video as shown in Fig. 7.13(d). Lastly, (4) rotate along Y_b axis to inspect and/or grab the object. This interface design also helps develop an orientation control law, provided in Sec. 7.3.2, that requires minimum vision feedback, i.e., without the object's depth or orientation. One can then compare human operator input with the orientation control input to formulate human intent prediction and human-machine shared control as provided in Sec. 7.3.3.

Table 7.2: Game Controller Keys versus Robot Motion

End-Effector Motion	**Key Name**	**Signal Type**	**Range**
$\pm X_b, \pm Y_b$ Translation	L-Joystick	Continuous (2-DOF)	$[-1, 1]$
$\pm X_b, \pm Y_b$ Rotation	R-Joystick	Continuous (2-DOF)	$[-1, 1]$
$-Z_b$ Translation	RB	Discrete	$\{0, 1\}$
$+Z_b$ Translation	LB	Discrete	$\{0, 1\}$
$-Z_b$ Rotation	RT	Discrete	$\{0, 1\}$
$+Z_b$ Rotation	LT	Discrete	$\{0, 1\}$
Gripper Open	A	Discrete	$\{0, 1\}$
Gripper Close	B	Discrete	$\{0, 1\}$

To realize the robot motion in the end-effector X_b, Y_b, Z_b frame mapped from the game controller, one can employ the body Jacobian in [110] to compute the required joint velocities $\dot{\mathbf{q}} \in \mathbb{R}^n$ given by

$$\dot{\mathbf{q}} = J_b(\mathbf{q})^\dagger \mathbf{u}_h \tag{7.20}$$

where $\mathbf{q} \in \mathbb{R}^n$ is the robot joint positions, $J_b(\mathbf{q})^\dagger$ is the pseudo-inverse of the body Jacobian $J_b(\mathbf{q}) \in \mathbb{R}^{6\times n}$ for an n-DOF robot, and $\mathbf{u}_h$ is the human input given by

$$\mathbf{u}_h = \Sigma \begin{bmatrix} \omega_x & \omega_y & \omega_z & v_x & v_y & v_z \end{bmatrix}^T \tag{7.21}$$

where $\omega_i \in \mathbb{R}$ and $v_i \in \mathbb{R}, i = x, y, z$, respectively, are the angular and translational velocities, and $\Sigma = \text{diag}(\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6)$ where $\sigma_i \in \mathbb{R}^+$, $i = 1, 2, \cdots, 6$ are design parameters that allow the designer to scale the corresponding velocities. We assume that the robot is in velocity-mode, i.e., joint velocities can be specified as inputs; most industrial robots provide an option for one or more command modes, i.e., position, velocity or torque modes, and we can similarly design the commanded input in other modes.

7.3.2 Orientation Control via Computer Vision

With the operational interface proposed in Section 7.3.1, one can still find that, in many situations, simultaneous coordination of translation and orientation to be challenging. We can utilize the camera and provide vision feedback to automate the X_b, Y_b orientation control such that the end-effector would keep pointing at the target object; while the human operator shares the control authority by focusing only on the control of rotation around Z_b and translation along X_b, Y_b, Z_b. This automatic orientation control will be activated based on human intent prediction and integrated into a human-machine shared control approach developed in Sec. 7.3.3. In the following we will first discuss general orientation control methods that require either depth and/or orientation of the object, which typically cannot be provided by low-cost cameras. This is in contrast to the method that we subsequently describe which does not require depth and orientation of the object.

A general control approach for moving the end-effector to a desired orientation is through the following joint velocity control action: $\dot{\mathbf{q}} = J_b(\mathbf{q})^\dagger u_e(e_\Omega)$ with u_e as a function of the orientation error $e_\Omega = [\Omega_e, 0, 0, 0]^T$ where $\Omega_e = \mathcal{F}[\log(R^T R_d)]$ and $R, R_d \in SO(3)$ are rotation matrices that encode the current and desired orientation in the robot base frame, $\log(R^T R_d)$ is the matrix logarithm of $R^T R_d$, and $\mathcal{F}[\cdot] : so(3) \mapsto \mathbb{R}^3$ maps a 3×3 skew-symmetric matrix to its vector form. One can choose any control action $u_e(e_\Omega)$; for example, a PID controller. The current end-effector orientation R can be found via robot forward kinematics. The desired orientation R_d is typically computed by one of the following two methods.

The first method is to specify the three Euler angles, roll ϕ, pitch θ, and yaw ψ, and then compute $R_d = R_Z(\psi)\, R_Y(\theta)\, R_X(\phi)$, where $R_I(j)$ for $i \in \{X, Y, Y\}$ and $j \in \{\phi, \theta, \psi\}$ denotes a rotation matrix encoding rotation along axis i by angle j. However, obtaining the Euler angles requires the computer vision algorithms to detect the orientation information in addition to the object's relative location in the camera's frame. The second method is to construct a coordinate frame that encodes the desired end-effector orientation and express it as a rotation matrix. For example, to align the end-effector to the normal direction of a surface, one can find, through measurement, two vectors $x_m, y_m \in \mathbb{R}^3$ (and $x_m \perp y_m$) that are tangent to a point on the surface. Then, by constructing a third vector from the point satisfying $x_m \perp y_m, y_m \perp z_m, z_m \perp x_m$, one can express the desired orientation by a rotation matrix as $R_d = [\hat{x}_m, \hat{y}_m, \hat{z}_m]$, where $\hat{x}_m, \hat{y}_m, \hat{z}_m$ are x_m, y_m, z_m normalized, respectively. Obtaining the vectors to which the end-effector gets aligned generally requires spatial location of the object computed from precise depth information. Additionally, to construct Ω_e, one has to compute the end-effector current orientation R via forward kinematics that can require significant computation.

We propose the following method for orientation control that does not require object's depth or orientation information. We consider a camera placed at the origin of the end-effector frame with the camera image axes aligned with the end-effector X_b, Y_b axes; this allows us to use orientation control around these axes to bring the object to the center of the image, as illustrated in Fig. 7.15(a). From the images, we can obtain the values of x_o and y_o which can be utilized to do orientation control. We define the X_b, Y_b orientation error to bring the object to the image center as

$$\mathbf{e}_o = \begin{bmatrix} +y_o & -x_o & 0 & 0 & 0 & 0 \end{bmatrix}^T . \tag{7.22}$$

We can form a control action $u_a(\mathbf{e}_o)$, which is a function of the orientation error, and consider the following joint velocity control:

$$\dot{\mathbf{q}} = J_b(\mathbf{q})^{\dagger}\mathbf{u}_a(\mathbf{e}_o). \tag{7.23}$$

One simple example control action is proportional control given by $\mathbf{u}_a(\mathbf{e}_o) = K_P\,\mathbf{e}_o$. This control law generates angular velocities to rotate the end-effector up and right until y_o and x_o are zero, respectively, and it does not require the depth information or computation of current rotation matrix.

7.3.3 Human Intent Prediction and Shared Control

In this section, we will first describe the method to predict the target object based on human intent that will facilitate generation of automatic orientation control input. Then, we will describe the shared control approach that blends the automatic orientation control input (rotation around X_b, Y_b axes) to bring the target to the center of the image, and human input to control translations along X_b, Y_b, Z_b axes and rotation around Z_b axis.

Assuming there are m objects detected from the image as illustrated in Fig. 7.15(b), showing three here for visual simplicity, we would get their location information (x_{oi}, y_{oi}) for $i = 1, 2, \cdots, m$. An automatic X_b, Y_b orientation control input to center object i in the image is given by $u_{ai} = K_P\mathbf{e}_{oi}$ where $\mathbf{e}_{oi} = [+y_{oi}, -x_{oi}]^T$. If it is observed that the human input as given by (7.21), then the X_b, Y_b orientation components can be utilized to obtain the intent of the human to center a specific object, which can be utilized to activate the automatic orientation control input towards that target object. Denote the vector $a_h = [\omega_x, \omega_y]^T$ containing the X_b, Y_b orientation components of the human input. Now, we compare the a_h human action vector with all the directional vectors u_{ai} for $i = 1, 2, \cdots, m$ and choose the object that closely aligns with a_h as the predicted target. We quantify this prediction as to which of the objects the human is intended to center by constructing a dynamic angle given by

$$\theta_i = \arccos\left(\frac{a_h \cdot u_{ai}}{||a_h||\,||u_{ai}||}\right). \tag{7.24}$$

Then, the index of the most likely target is given by

$$k = \underset{i}{\operatorname{argmin}}\ \theta_i \tag{7.25}$$

for $i = 1, 2, \cdots, m$, and the smallest angle is θ_k.

Then, to provide intelligent assistance to the human operator where the X_b, Y_b orientation is assisted by the automatic control, we propose the following shared control law:

$$\dot{\mathbf{q}} = J_b(\mathbf{q})^{\dagger}\mathbf{u}_b \tag{7.26}$$

where $\mathbf{u_b}$ is a shared control input term. To facilitate the expression of $\mathbf{u_b}$, we first re-write the human input in (7.21) as

$$\mathbf{u}_h = \begin{bmatrix}\mathbf{u}_{h1}^T & \mathbf{u}_{h2}^T\end{bmatrix}^T \tag{7.27}$$

where $\mathbf{u}_{h1} \in \mathbb{R}^2$ and $\mathbf{u}_{h2} \in \mathbb{R}^4$ are the first two and last four components of $\mathbf{u}_h$. Then, the shared control input term can be written as

$$\mathbf{u}_b = \begin{cases}[\mathbf{0}^T\ \ \mathbf{u}_{h2}^T]^T + \mathbf{u}_a, & \theta_k < \theta^* \\ \mathbf{u}_h, & \text{otherwise}\end{cases} \tag{7.28}$$

where θ^* is a design parameter for a prediction confidence threshold. It means that the X_b, Y_b orientation control is delegated to automatic control if there is a predicted target, and we do so only when we are confident about the human intent by checking if $\theta_k < \theta^*$. The realtime implementation for this shared control law is provided in Algorithm 6.

Remark 4. *Unlike conducting path-planning before moving a robot where the target location is*

Algorithm 6 Realtime Shared Control

```
Input: Objects location $(x_{oi}, y_{oi}) \in \mathbb{R}^2$ for $i = 1, 2, \cdots, m$, human operator input $\mathbf{u}_h \in \mathbb{R}^6$,
    prediction confidence threshold parameter $\theta^* \in \mathbb{R}^+$
Output: Required robot joint velocity vector $\dot{\mathbf{q}} \in \mathbb{R}^n$
    Initialization 1 Start with manual control given by (7.20), set a global variable $K$ to store the
    predicted object index
 2: while (robot is not shutdown) do
      if (no object is detected) then
 4:     Use manual control law given by (7.20)
      else
 6:     if ($\mathbf{u}_{h1} \neq \mathbf{0}$) then
          for (object index $i = 1, 2, \cdots, m$) do
 8:         Compute $u_{ai}$, i.e., set $a_h = \mathbf{u}_{h1}$
            Compute $\theta_i$ given by (7.24)
10:       end for
          Compute the predicted object index $k$ given by (7.25)
12:       Update global variable $K = k$
        else
14:       Preserve the previous value for global variable $K$
        end if
16:     Use shared control law given by (7.26)
      end if
18: end while
```

known and fixed in the environment, achieving end-effector motion via robot Jacobians such as in (7.26) *may sometimes encounter robot configurations that are close to being singular. We provide the following heuristic solution to address this issue and process the output joint velocities of Algorithm 6 before sending them to the robot. Assuming that the robot joint velocities are limited by motors as*

$$-\dot{q}_{\max} \leq \dot{q}_i \leq \dot{q}_{\max} \tag{7.29}$$

for $i = 1, 2, \cdots, n$, where $\dot{q}_{\max} \in \mathbb{R}^+$ is the magnitude of the limiting speed. Then, we process the required joint velocity vector $\dot{\mathbf{q}}$ by

$$\dot{\mathbf{q}}^* = \begin{cases} \frac{\dot{q}_{\max}}{\dot{q}_j}\dot{\mathbf{q}}, & \dot{q}_j \geq \dot{q}_{\max} \\ \dot{\mathbf{q}}, & \text{otherwise} \end{cases} \tag{7.30}$$

where $\dot{q}_j = \max(\dot{q}_1, \dot{q}_2, \cdots, \dot{q}_n)$ *is the maximum among all the joint velocities computed in* (7.26)*, and* $\dot{\mathbf{q}}^*$ *is the final joint velocity sent to the robot. The result of such processing is a velocity reduction (when approaching physical limits or singularities) without affecting the intended motion direction by maintaining the proportionality among the elements of* $\dot{\mathbf{q}}$.

Remark 5. *Note that (a) singularities often affect some, not all, DOF, (b) joint velocity scaling in* (7.30) *would make it increasingly slow for the robot to actually reach singularities, and (c) the human operator can always sense a significant reduction of speed when the input magnitude from the game controller stays the same. Thus, this approach of scaling joint velocities allows the human operator to choose a different path before the computer fails to compute* $J_b(\mathbf{q})^\dagger$.

7.3.4 Experiments and Results

To corroborate the proposed methods, we employed the robot system provided in Fig. 7.16 for an object inspection and handling task. In this setup, we integrated a six DOF ABB robot with a gripper (the development can be adapted to any general articulated robots). The gripper consists of a Beaglebone controller, two servo motors connected with two fingers, a center camera and a side camera. When the object and the gripper fingers are aligned at the center of the image, it is difficult for the human operator to ascertain the distance from the fingertips to the object for accurate gripping. The side camera is employed to resolve this problem by providing the human operator another view from a different angle. The Beaglebone controller controls the motion of the motors. The cameras can provide images and use computer vision algorithms to detect the X-Y locations of objects with features specified by the user. The task requires a human operator controlling the robot to inspect a tennis ball along a desired trajectory shown in Fig. 7.17 while (1) keeping the object at the center of the images, and (2) keeping the trajectory line at the center of the object. After reaching the end point of the trajectory, the human operator is to approach and grab the object with the gripper. During this process, the human operator is only allowed to look at the video provided by the gripper camera on a computer.

The real-time control with the proposed method is implemented in the Robotic Operating System with a wireless network connecting a computer and the Beaglebone Controller. An illustration of the software operation is provided in Fig. 7.18, where ABB Externally Guided Motion (ABB EGM) library allows the user to specify robot joint velocities via the computer; the Beaglebone Controller commands the finger motors via Pulse Width Modulation (PWM) signals.

Sampled robot configurations are provided in Fig. 7.19, and their temporal order is indicated by numbers from 1 to 9 with 1 as the initial configuration. The tennis ball was not aligned with the end-effector center at the initial configuration, whereas at configurations 2 through 9, the tennis ball was centered with respect to the end-effector. Figure 7.20 provides sampled images from the human operator's perspective with their temporal order indicated by Roman numerals from i to vi. One can observe that the tennis ball was centered to the images and the marked trajectory was at the center of the object as required. Note that sampled robot configurations and images from the human's perspective are obtained while the end-effector is moved to follow the arrow trajectory indicated on the tennis ball.

Figure 7.21 provides the object's coordinates detected by the vision algorithm, human operator's input, and the dynamic prediction angle evolution during the entire process. Note that all the inputs and measurements are normalized to the range of $[-1, 1]$. The confidence threshold angle in (7.28) was set to $\theta^* = 10^\circ$ for the experiment. There are four stages of the process as indicated in Fig. 7.21(b) and (c). Initially, the object was off-center in the image. During the 1st stage, the human operator issued command which includes orientation components. One can observe that in the meantime, the dynamic angle in Figure 7.21(d) gradually decreases and drops below the threshold; the automatic orientation control gets activated, and the robot starts to center the object as reflected in Fig. 7.21(a). During the 2nd and the 3rd stages, the human only controls the X_b and Y_b translation of the end-effector, respectively, before and after hitting the turning point provided in Fig. 7.17; while the automatic control keeps the object at the center of the image. During the 4th stage, the robot has reached the end point and the human operator slightly adjusted the X_b and Y_b translation for some final calibration, then approached and grabbed the object as shown in configuration 9 of Fig. 7.19.

One can observe that the dynamic angle at Stages 2, 3, and 4 stays at the same value even when the X_b, Y_b components of human orientation input are zero. This is because in the implementation of Algorithm 6, we preserve the last predicted object index and the dynamic angle if the X_b, Y_b components of human orientation input go back to zero (before detecting the first X_b, Y_b orientation components from human input, there is no dynamic angle computed). This makes it easier for the human operator because once the operator realizes that the automatic orientation control is activated correctly, the operator can just focus on making the X_b, Y_b, Z_b translation and Z_b rotation, without having to worry about the X_b, Y_b orientation.

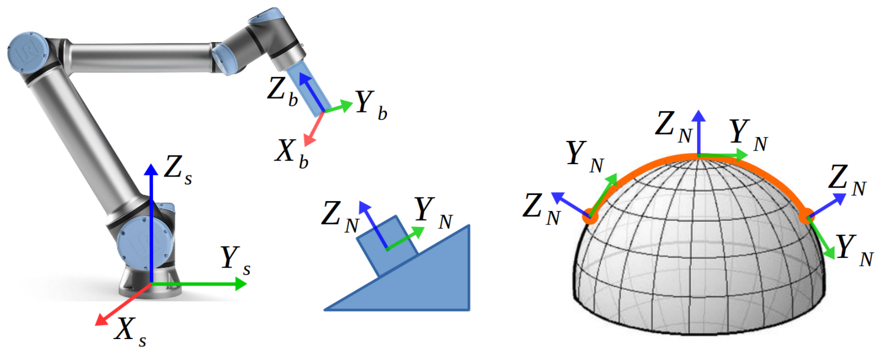


Figure 7.1: Extraction of Operating Frames and Procedure

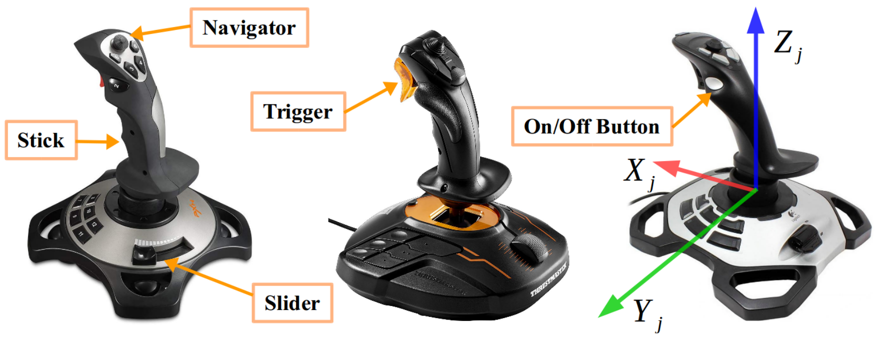


Figure 7.2: General Purpose Joysticks and Common Features

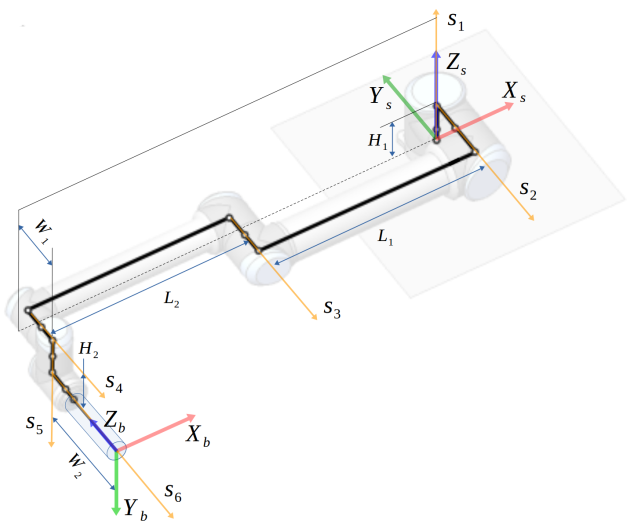


Figure 7.3: UR5 Frames and Axes Definitions

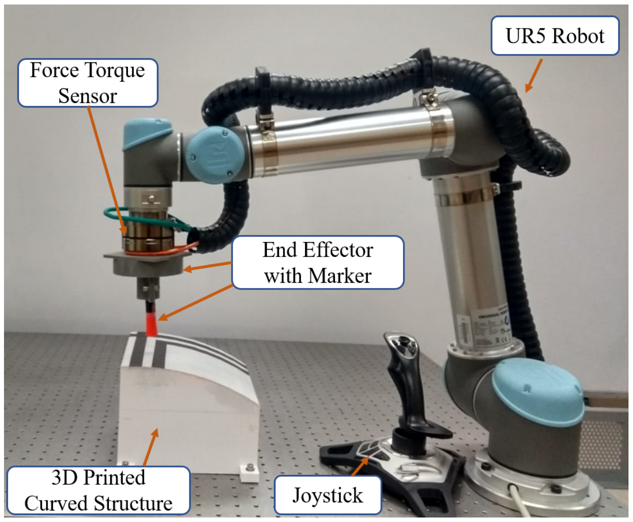


Figure 7.4: Setup for Surface Tracking Experiment

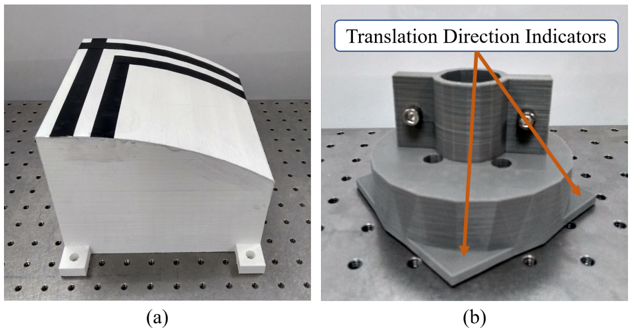


Figure 7.5: Curved Surface with Constrained Area and End-effector

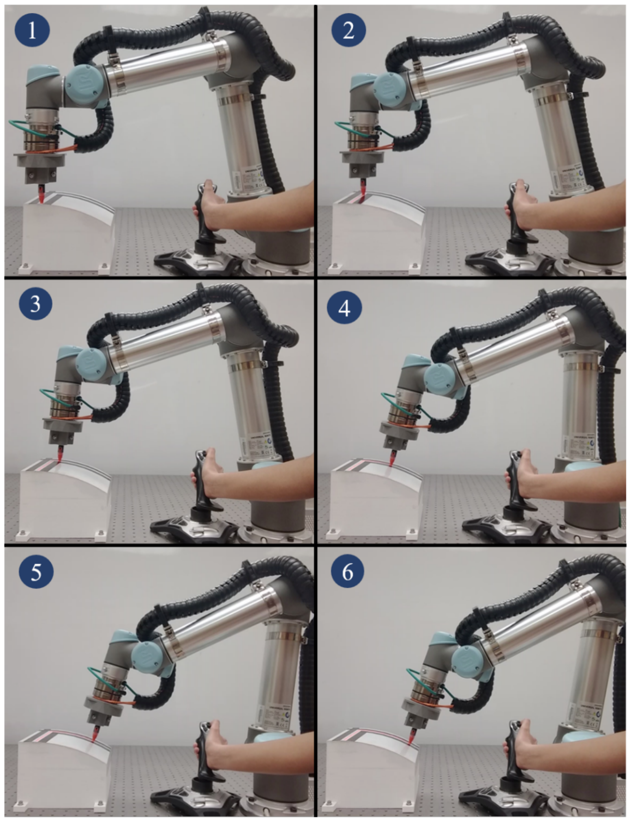


Figure 7.6: Sampled Robot Configurations during Surface Tracking Experiment

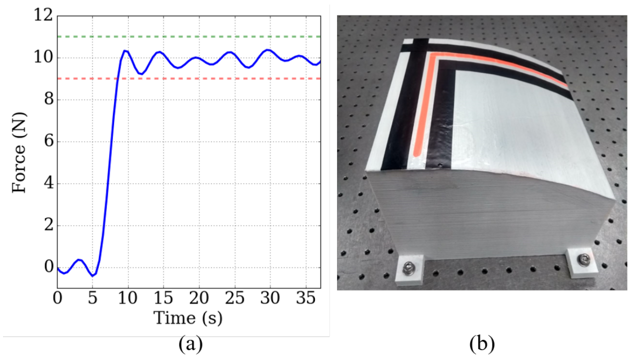


Figure 7.7: Contact Force and Finished Surface

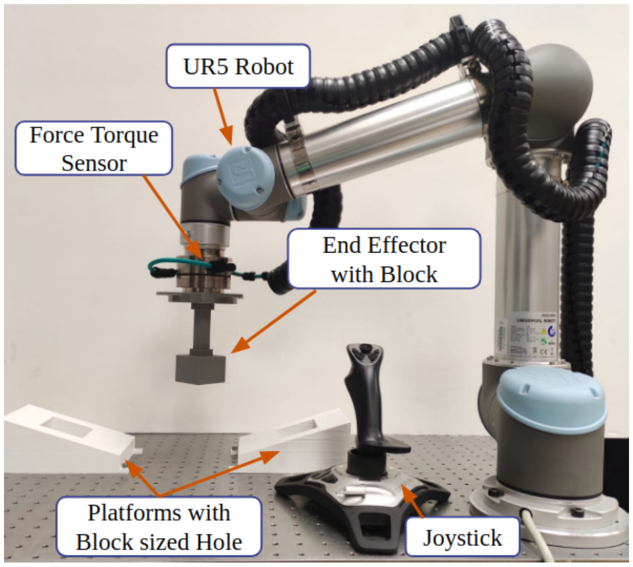


Figure 7.8: Setup for Material Handling Experiment

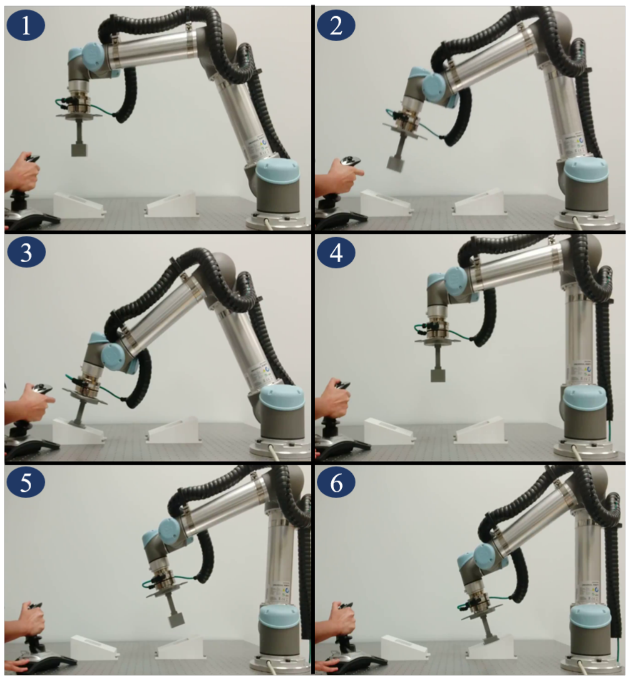
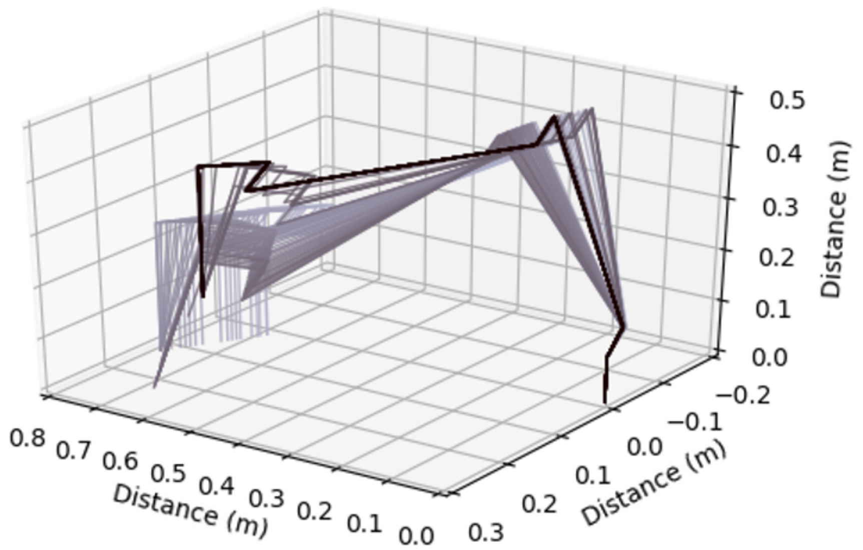
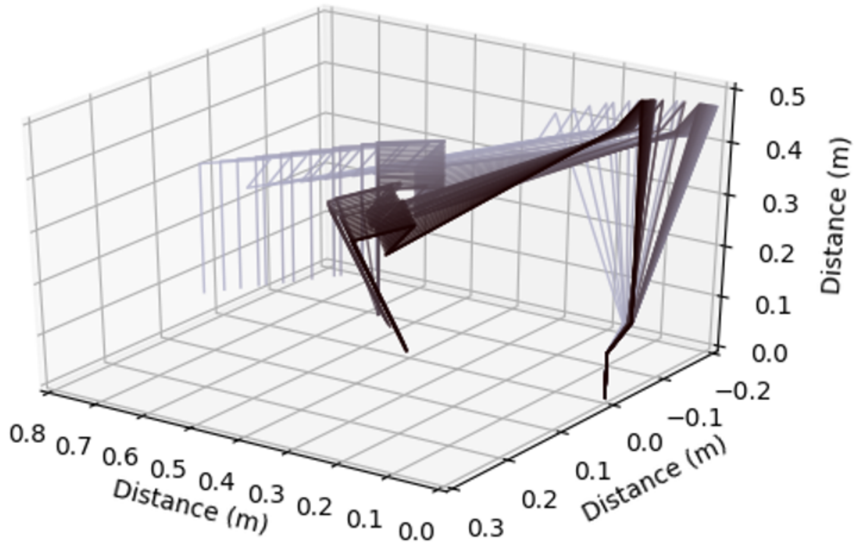


Figure 7.9: Sampled Robot Configurations during Material Handling Experiment



(a)



(b)

Figure 7.10: Robot Configurations Wire-frame Diagram during Operation

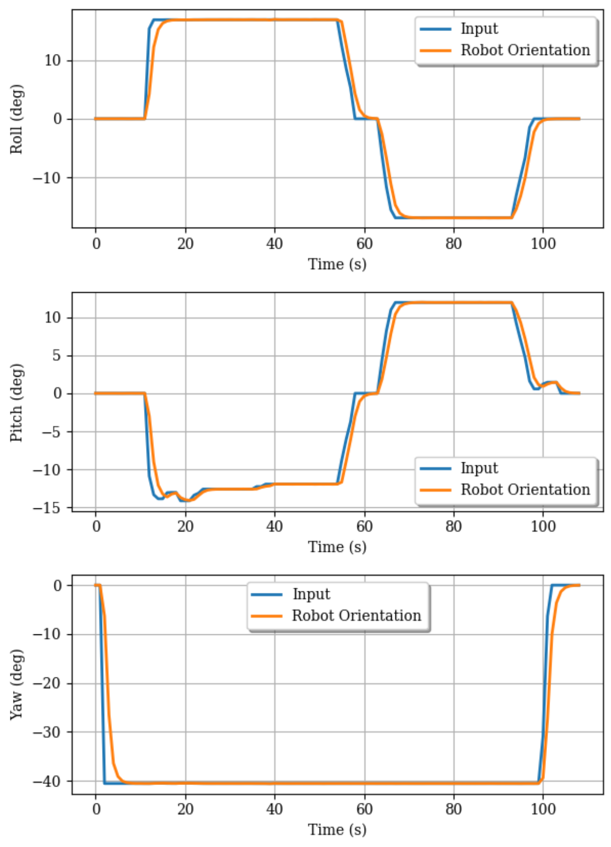


Figure 7.11: End-Effector Orientation Tracking with Joystick Input

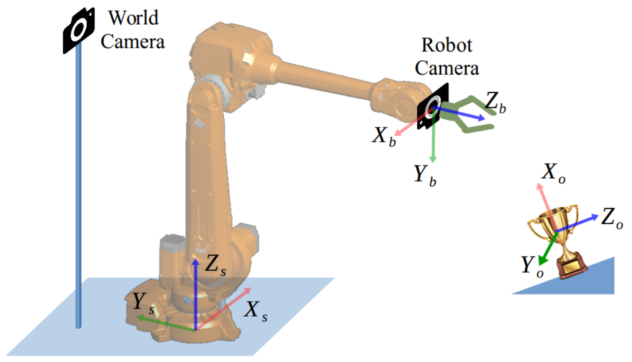


Figure 7.12: Example of Using a Robot to Inspect/Handle an Object with World or End-Effector Camera via Remote Human Operation

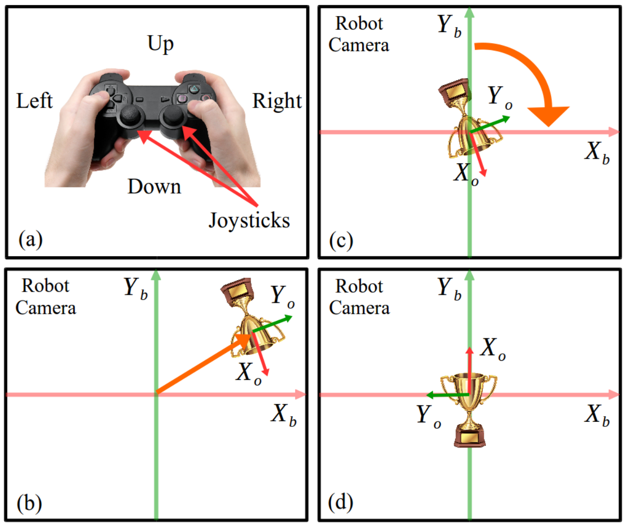


Figure 7.13: Steps to Realize Object Inspection and Handling

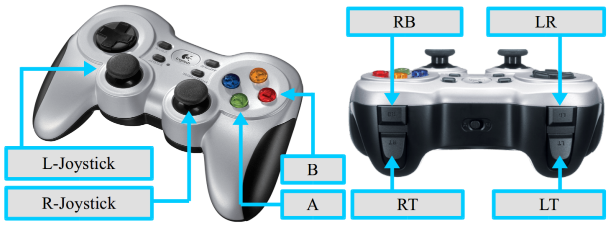


Figure 7.14: Operational Interface with a General-Purpose Game Controller

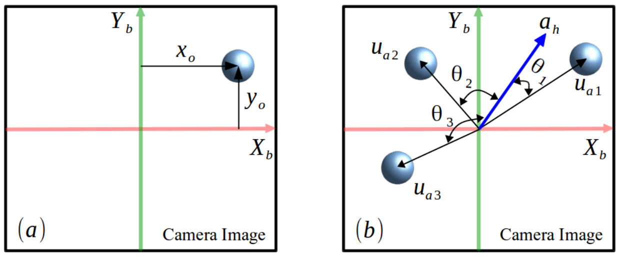


Figure 7.15: Object in Camera's View and Human Intent Prediction Concept

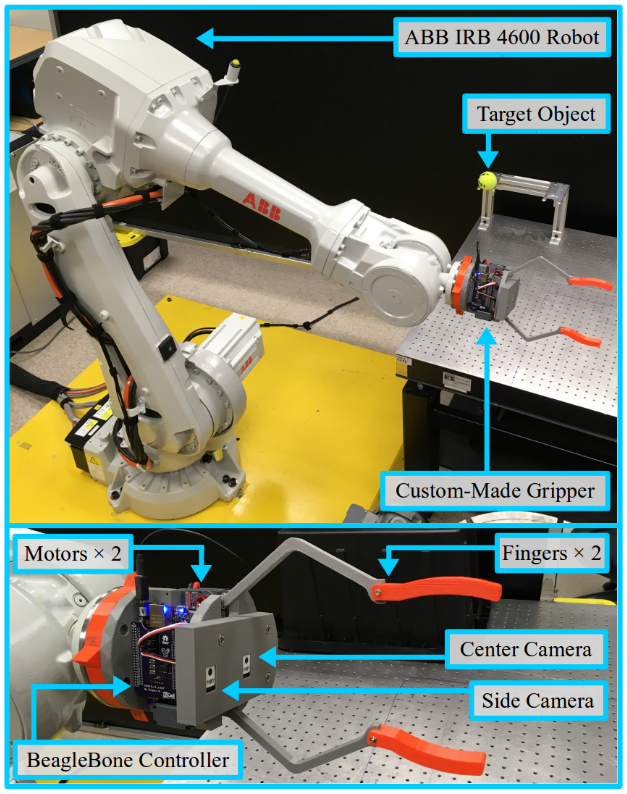


Figure 7.16: Realization of the Proposed Robot System with a Gripper

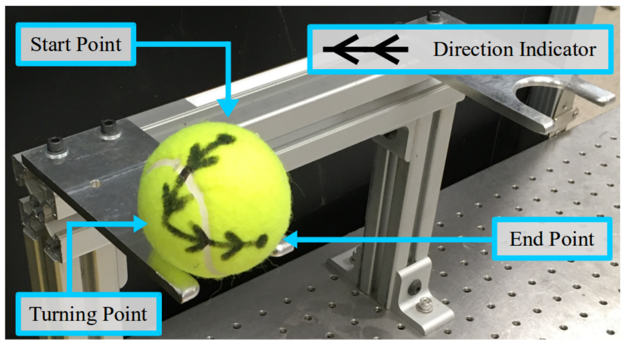


Figure 7.17: Target Object with Marked Inspection Trajectory and Directions

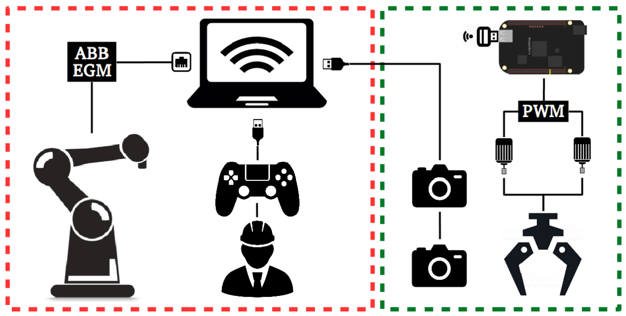


Figure 7.18: Software Operation with Robotic Operating System

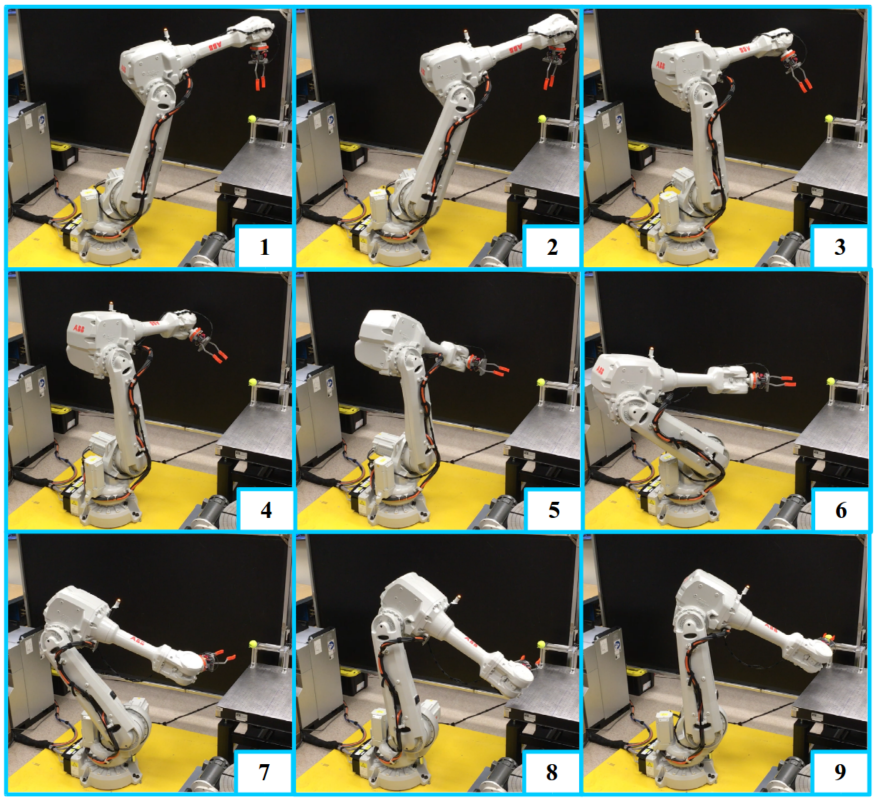


Figure 7.19: Sampled Robot Configurations during Task Execution

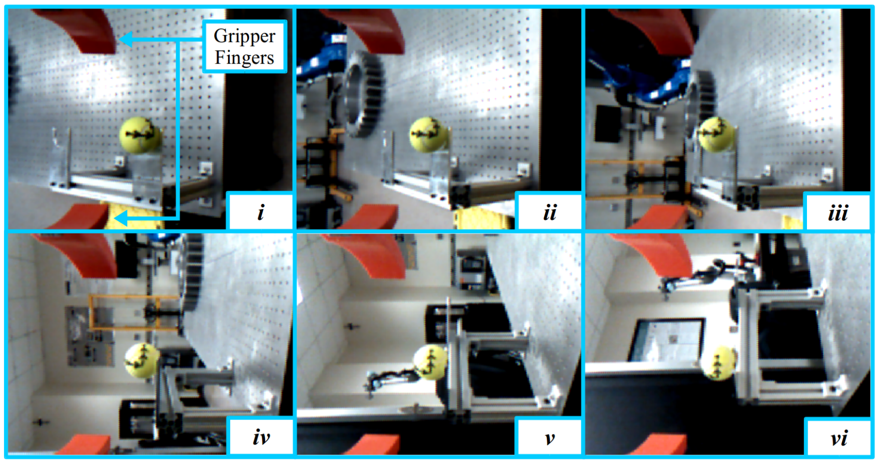


Figure 7.20: Sampled Images from Human's View on the Computer Screen

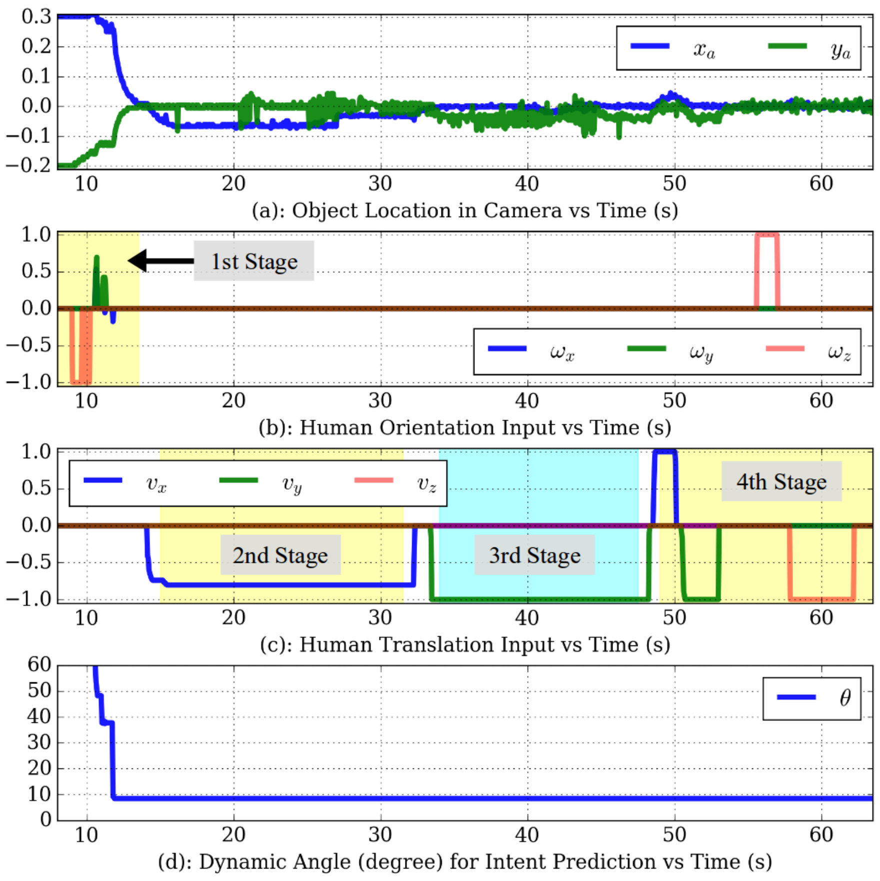


Figure 7.21: Object Location from Camera feedback, Human Orientation Input, Human Translation Input, and Dynamic Angle for Intent Prediction

8. SUMMARY AND FUTURE WORK

In this dissertation, we have presented general concepts and developed novel methods for human-machine shared control for collaborative robotics applications. Among the proposed methods, we have introduced new concepts in task learning by employing the notions of operator primitive based segmentation, motion command variance, and Bayesian non-parametric clustering with temporal ordering. With regard to predicting the intent of the human operator for transition from one subgoal to another, we have modeled the intent prediction as a Markov chain, employed the notions of empirical stochastic transition matrix to encode prior knowledge of a task and subgoals, and proposed dynamic angle difference exponential to quantify in real-time the operator's intent for seeking a particular subgoal. We also introduced a method to dynamically learn and update the ESTM based on observed subgoal transitions instead of relying on empirical knowledge to set fixed values for the stochastic transition matrix. In terms of adjusting nominal subgoals of a given task, adjustment encoding hyper-rectangle, hyperbolic slope transition function, and skill-weighted action integral were proposed. A technique for blending automatic controller input and human operator input with conflict awareness was also provided. Experimental results on a scaled excavator platform corroborated the effectiveness and performance enhancement of the proposed BSC tools.

In addition to the BSC methods developed and corroborated with the excavator as an example, we also proposed SC methods for the collaborative operation of robotic manipulators. We developed two novel operational interfaces for a human operator to effectively control the motion of a robot end-effector, each with a different set of applicable tasks. We integrated vision/force/motion hybrid control to realize the intended motion of the end-effector based on human input and sensor feedback. We also proposed an human intent prediction methods for manipulator operation by comparing the orientation input from the human operator and the closed-loop orientation control input based on vision feedback. Based on the intent prediction, we proposed a method to share control authority between the human operator and vision-based controller. We implemented our methods for a UR5 robot and an ABB IRB-4600 robot and showed the effectiveness of the methods with experimental results.

However, there are still many open questions in human-machine shared control. For example, one can consider the following topics for further research. (1) How to formulate and evaluate closed-loop system stability when the human operator is in-the-loop and how to utilize more advanced control algorithms to provide automatic control assistance? (2) The input blending formulation presented in this work, as well as in existing studies, assume that the human is directly controlling the actuators. In many collaborative robotics applications, the human may be providing input in the form of the Cartesian trajectory reference for the end-effector. In these situations, one has to investigate the problem of mapping the joystick input to a task reference frame and translating it to the end-effector reference frame. Subsequently, one has to address the question of whether to blend the inputs at the trajectory level or at the actuator input level? (3) How does one augment the capability of human operators, hence the performance of shared control, by introducing sensory information such as haptic feedback? With advances in haptic feedback devices and their accessibility, researchers having been applying haptic feedback in many applications with promising results. Haptic feedback has the potential to help a human operator to gain more insight of the robot and its environment such as getting close to singularity, approaching limits of workspace, having significant contact force or workload. It might also help novice operators to learn the operation of robots more quickly by providing resistance to the joysticks along directions irrelevant to the training tasks, etc.

REFERENCES

[1] M. Tomasello, *A Natural History of Human Thinking*. Harvard University Press, 2014.

[2] N. Sebanz, H. Bekkering, and G. Knoblich, "Joint action: Bodies and minds moving together," *Trends in Cognitive Sciences*, vol. 10, pp. 70–76, 03 2006.

[3] S. S. Obhi and N. Sebanz, "Moving together: toward understanding the mechanisms of joint action," *Experimental Brain Research*, vol. 211, p. 329, May 2011.

[4] R. Parasuraman, T. B. Sheridan, and C. D. Wickens, "A model for types and levels of human interaction with automation," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 30, pp. 286–297, May 2000.

[5] M. A. Goodrich and A. C. Schultz, "Human–robot interaction: A survey," *Foundations and Trends® in Human–Computer Interaction*, vol. 1, no. 3, pp. 203–275, 2008.

[6] E. de Visser and R. Parasuraman, "Adaptive aiding of human-robot teaming: Effects of imperfect automation on performance, trust, and workload," *Journal of Cognitive Engineering and Decision Making*, vol. 5, no. 2, pp. 209–231, 2011.

[7] T. B. Sheridan, *Telerobotics, Automation, and Human Supervisory Control*. Cambridge, MA, USA: MIT Press, 1992.

[8] E. Hollnagel and D. D. Woods, "Cognitive systems engineering: New wine in new bottles," *International Journal of Human-Computer Studies*, vol. 51, no. 2, pp. 339 – 356, 1999.

[9] F. Flemisch, D. Abbink, M. Itoh, M. Pacaux-Lemoine, and G. WeBel, "Shared control is the sharp end of cooperation: Towards a common framework of joint action, shared control and human machine cooperation," *13th IFAC Symposium on Analysis, Design, and Evaluation of Human-Machine Systems*, vol. 49, pp. 72 – 77, 2016.

[10] V. Dimitrov and T. Padır, “A shared control architecture for human-in-the-loop robotics applications,” in *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pp. 1089–1094, Aug 2014.

[11] D. P. Losey, C. G. McDonald, E. Battaglia, and M. K. O’Malley, “A review of intent detection, arbitration, and communication aspects of shared control for physical human-robot interaction,” *Applied Mechanics Reviews*, vol. 70, 02/2018 2018.

[12] A. Enes and W. Book, “Blended shared control of zermelo’s navigation problem,” in *American Control Conference*, pp. 4307–4312, 2010.

[13] Y. Li, K. P. Tee, W. L. Chan, R. Yan, Y. Chua, and D. K. Limbu, “Continuous role adaptation for human-robot shared control,” *IEEE Transactions on Robotics*, vol. 31, pp. 672–681, June 2015.

[14] M. J. A. Zeestraten, I. Havoutis, and S. Calinon, “Programming by demonstration for shared control with an application in teleoperation,” *IEEE Robotics and Automation Letters*, vol. 3, pp. 1848–1855, July 2018.

[15] H. Maske, E. Kieson, G. Chowdhary, and A. Charles, “Learning task-based instructional policy for excavator-like machines,” in *IEEE International Conference on Robotics and Automation*, 2018.

[16] A. M. Brandt and M. B. Colton, “Haptic collision avoidance for a remotely operated quadrotor uav in indoor environments,” in *2010 IEEE International Conference on Systems, Man and Cybernetics*, pp. 2724–2731, Oct 2010.

[17] J. Chun, I. Lee, G. Park, J. Seo, S. Choi, and S. H. Han, “Efficacy of haptic blind spot warnings applied through a steering wheel or a seatbelt,” *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 21, pp. 231 – 241, 2013.

[18] C. Guo, C. Sentouh, J. C. Popieul, B. Soualmi, and J. B. Haué, "Shared control framework applied for vehicle longitudinal control in highway merging scenarios," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 3098–3103, Oct 2015.

[19] P. Wang, Z. Man, Z. Cao, J. Zheng, and Y. Zhao, "Dynamics modelling and linear control of quadcopter," in *2016 International Conference on Advanced Mechatronic Systems (ICAMechS)*, pp. 498–503, Nov 2016.

[20] L. Sciavicco, B. Siciliano, and B. Sciavicco, *Modelling and Control of Robot Manipulators*. Berlin, Heidelberg: Springer-Verlag, 2nd ed., 2000.

[21] A. Hansson and M. Servin, "Semi-autonomous shared control of large-scale manipulator arms," *Control Engineering Practice*, vol. 18, pp. 1069–1076, 2010.

[22] J. Philips, J. del R. Millan, G. Vanacker, E. Lew, F. Galan, P. W. Ferrez, H. V. Brussel, and M. Nuttin, "Adaptive shared control of a brain-actuated simulated wheelchair," in *2007 IEEE 10th International Conference on Rehabilitation Robotics*, pp. 408–414, June 2007.

[23] G. Pires and U. Nunes, "A wheelchair steered through voice commands and assisted by a reactive fuzzy-logic controller," *Journal of Intelligent and Robotic Systems*, vol. 34, pp. 301–314, Jul 2002.

[24] L. Kitagawa, T. Kobayashi, T. Beppu, and K. Terashima, "Semi-autonomous obstacle avoidance of omnidirectional wheelchair by joystick impedance control," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, vol. 4, pp. 2148–2153 vol.4, 2001.

[25] C. Ezeh, P. Trautman, C. Holloway, and T. Carlson, "Comparing shared control approaches for alternative interfaces: A wheelchair simulator experiment," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 93–98, Oct 2017.

[26] L. Devigne, V. K. Narayanan, F. Pasteau, and M. Babel, “Low complex sensor-based shared control for power wheelchair navigation,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5434–5439, Oct 2016.

[27] J. G. Storms and D. M. Tilbury, “Blending of human and obstacle avoidance control for a high speed mobile robot,” in *2014 American Control Conference*, pp. 3488–3493, June 2014.

[28] T. K. Stephens, N. J. Kong, R. L. Dockter, J. J. O’Neill, R. M. Sweet, and T. M. Kowalewski, “Blended shared control utilizing online identification,” *International Journal of Computer Assisted Radiology and Surgery*, vol. 13, pp. 769–776, Jun 2018.

[29] K. Shamaei, Y. Che, A. Murali, S. Sen, S. Patil, K. Goldberg, and A. M. Okamura, “A paced shared-control teleoperated architecture for supervised automation of multilateral surgical tasks,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1434–1439, Sept 2015.

[30] A. Enes R., “Shared control of hydraulic manipulators to decrease cycle time,” *Georgia Institute of Technology*, 12 2010.

[31] M. Allain, S. Konduri, H. Maske, P. R. Pagilla, and G. Chowdhary, “Blended shared control of a hydraulic excavator,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 14928 – 14933, 2017. 20th IFAC World Congress.

[32] Z. Jin, P. R. Pagilla, H. Maske, and G. Chowdhary, “Methods for blended shared control of a hydraulic excavator with learning and prediction,” in *IEEE Conference on Decision and Control*, 2018.

[33] Z. Jin, P. R. Pagilla, H. Maske, and G. Chowdhary, “Blended shared control with subgoal adjustment,” in *IEEE Conference on Systems, Man, and Cybernetics*, 2018.

[34] B. Michini, T. Walsh, A. Agha-Mohammadi, and J. How, “Bayesian nonparametric reward learning from demonstration,” *IEEE Transactions on Robotics*, vol. 31, no. 2, pp. 369–386, 2015.

[35] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, “A survey of robot learning from demonstration,” *Robotics and Autonomous Systems*, vol. 67, pp. 469–483, January 2009.

[36] J. Kober, J. A. Bagnell, and J. Peters, “Reinforcement learning in robotics: A survey,” *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[37] M. Wulfmeier, D. Rao, D. Z. Wang, P. Ondruska, and I. Posner, “Large-scale cost function learning for path planning using deep inverse reinforcement learning,” *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1073–1087, 2017.

[38] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan, “Cooperative inverse reinforcement learning,” in *Advances in Neural Information Processing Systems 29* (D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds.), pp. 3909–3917, Curran Associates, Inc., 2016.

[39] M. Wulfmeier, P. Ondruska, and I. Posner, “Deep inverse reinforcement learning,” *CoRR*, vol. abs/1507.04888, 2015.

[40] M. Jin and C. J. Spanos, “Inverse reinforcement learning via deep gaussian process,” *CoRR*, vol. abs/1512.08065, 2015.

[41] S. Zhifei and E. M. Joo, “A survey of inverse reinforcement learning techniques,” *International Journal of Intelligent Computing and Cybernetics*, vol. 5, no. 3, pp. 293–311, 2012.

[42] P. Abbeel and A. Y. Ng, “Apprenticeship learning via inverse reinforcement learning,” in *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML ’04, ACM, 2004.

[43] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pp. 663–670, Morgan Kaufmann Publishers Inc., 2000.

[44] A. Sosic, E. Rueckert, J. Peters, A. M. Zoubir, and H. Koeppl, "Inverse reinforcement learning via nonparametric spatio-temporal subgoal modeling," *CoRR*, vol. abs/1803.00444, 2018.

[45] X. Pan and Y. Shen, "Human-interactive subgoal supervision for efficient inverse reinforcement learning," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, (Richland, SC), pp. 1380–1387, International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[46] R. F. Adler and R. Benbunan-Fich, "The effects of task difficulty and multitasking on performance," *Interacting with Computers*, vol. 27, no. 4, pp. 430–439, 2015.

[47] M. Harré, T. Bossomaier, and A. Snyder, "The development of human expertise in a complex environment," *Minds and Machines*, vol. 21, pp. 449–464, Aug 2011.

[48] I. Cornford and J. Athanasou, "Developing expertise through training," *Industrial and Commercial Training*, vol. 27, no. 2, pp. 10–18, 1995.

[49] H. Maske, E. Kieson, G. Chowdhary, and C. Abramson, "Can co-robots learn to teach?," *CoRR*, vol. abs/1611.07490, 2016.

[50] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[51] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theor.*, vol. 28, pp. 129–137, Sept. 2006.

[52] Z. Jin, P. R. Pagilla, and G. Chowdury, "Subgoal learning from demonstration via operator command quantification for shared control applications (submitted)," in *2019 IEEE Conference on Decision and Control (CDC)*, 2019.

[53] D. Paindaveine, "A canonical definition of shape," *Statistics and Probability Letters*, vol. 78, no. 14, pp. 2240 – 2247, 2008.

[54] L. Wasserman and L. Wasserman, *All of Statistics: A Concise Course in Statistical Inference*. Springer Texts in Statistics, Springer, 2004.

[55] J. Pitman, "Some developments of the blackwell-macqueen urn scheme," *Lecture Notes-Monograph Series*, vol. 30, pp. 245–267, 1996.

[56] R. M. Neal, "Markov chain sampling methods for dirichlet process mixture models," *Journal of Computational and Graphical Statistics*, vol. 9, no. 2, pp. 249–265, 2000.

[57] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *ROBOMECH Journal*, vol. 1, p. 1, Jul 2014.

[58] A. Doshi and M. M. Trivedi, "Tactical driver behavior prediction and intent inference: A review," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 1892–1897, Oct 2011.

[59] F. Gross, J. Jordan, F. Weninger, F. Klanner, and B. Schuller, "Route and stopping intent prediction at intersections from car fleet data," *IEEE Transactions on Intelligent Vehicles*, vol. 1, pp. 177–186, June 2016.

[60] S. Klingelschmitt, M. Platho, H.-M. Groß, V. Willert, and J. Eggert, "Combining behavior and situation information for reliably estimating multiple intentions," in *2014 IEEE Intelligent Vehicles Symposium (IV)*, pp. 388–393, IEEE, June 2014.

[61] A. Bender, J. R. Ward, S. Worrall, and E. M. Nebot, "Predicting driver intent from models of naturalistic driving," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pp. 1609–1615, Sep. 2015.

[62] T. Streubel and K.-H. Hoffmann, "Prediction of driver intended path at intersections," *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pp. 134–139, 2014.

[63] P. Dokania, M. Perrollaz, S. Lefevre, and C. Laugier, "Learning-based approach for online lane change intention prediction," in *IEEE Intelligent Vehicles Symposium, Proceedings*, 06 2013.

[64] J. Gunnarsson, L. Svensson, E. Bengtsson, and L. Danielsson, "Joint driver intention classification and tracking of vehicles," in *2006 IEEE Nonlinear Statistical Signal Processing Workshop*, pp. 95–98, Sep. 2006.

[65] G. S. Aoude, V. R. Desaraju, L. H. Stephens, and J. P. How, "Driver behavior classification at intersections and validation on large naturalistic data set," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, pp. 724–736, June 2012.

[66] A. Zyner, S. Worrall, J. Ward, and E. Nebot, "Long short term memory for driver intent prediction," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1484–1489, June 2017.

[67] A. F. Tarhan, E. Koyuncu, M. Hasanzade, U. Ozdemir, and G. Inalhan, "Formal intent based flight management system design for unmanned aerial vehicles," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 984–992, May 2014.

[68] Y. Lin, J.-w. Zhang, and H. Liu, "An algorithm for trajectory prediction of flight plan based on relative motion between positions," *Frontiers of Information Technology & Electronic Engineering*, vol. 19, pp. 905–916, Jul 2018.

[69] K. Kim and I. Hwang, "Intent-based detection and characterization of aircraft maneuvers in en route airspace," *Journal of Aerospace Information Systems*, vol. 15, no. 2, pp. 72–91, 2018.

[70] D. A. Jimmy Krozel, "Intent inference with path prediction," *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 2, pp. 225–236, 2006.

[71] I. H. Javier Lovera Yepes and M. Rotea, "New algorithms for aircraft intent inference and trajectory prediction," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 2, pp. 370–382, 2007.

[72] R. Bednarik, H. Vrzakova, and M. Hradis, "What do you want to do next: A novel approach for intent prediction in gaze-based interaction," in *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '12, pp. 83–90, ACM, 2012.

[73] A. Ajanki, D. R. Hardoon, S. Kaski, K. Puolamäki, and J. Shawe-Taylor, "Can eyes reveal interest? implicit queries from gaze patterns," *User Modeling and User-Adapted Interaction*, vol. 19, pp. 307–339, Oct 2009.

[74] B. P. Bailey and S. T. Iqbal, "Understanding changes in mental workload during execution of goal-directed tasks and its application for interruption management," *ACM Trans. Comput.-Hum. Interact.*, vol. 14, pp. 21:1–21:28, Jan. 2008.

[75] S. Eivazi and R. Bednarik, "Predicting problem-solving behavior and performance levels from visual attention data," in *2nd Workshop on Eye Gaze in Intelligent Human Machine Interaction at ACM IUI*, pp. 9–16, 2011.

[76] M. Kandemir, V.-M. Saarinen, and S. Kaski, "Inferring object relevance from gaze in dynamic scenes," in *Proceedings of the 2010 Symposium on Eye-Tracking Research &#38; Applications*, ETRA '10, pp. 105–108, ACM, 2010.

[77] R. Andersen, S. Musallam, and B. Pesaran, "Selecting the signals for a brain machine interface," *Current Opinion in Neurobiology*, vol. 14, no. 6, pp. 720 – 726, 2004.

[78] L. Bi, X. Fan, and Y. Liu, "Eeg-based brain-controlled mobile robots: A survey," *IEEE Transactions on Human-Machine Systems*, vol. 43, pp. 161–176, March 2013.

[79] M. Fatourechi, A. Bashashati, R. K. Ward, and G. E. Birch, “Emg and eog artifacts in brain computer interface systems: A survey,” *Clinical Neurophysiology*, vol. 118, no. 3, pp. 480 – 494, 2007.

[80] S. N. Abdulkader, A. Atia, and M.-S. M. Mostafa, “Brain computer interfacing: Applications and challenges,” *Egyptian Informatics Journal*, vol. 16, no. 2, pp. 213 – 230, 2015.

[81] J. Thomas, T. Maszczyk, N. Sinha, T. Kluge, and J. Dauwels, “Deep learning-based classification for brain-computer interfaces,” in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 234–239, Oct 2017.

[82] K. R. Müller, M. Krauledat, G. Dornhege, G. Curio, and B. Blankertz, “Machine learning and applications for brain-computer interfacing,” in *Human Interface and the Management of Information. Methods, Techniques and Tools in Information Design*, pp. 705–714, Springer Berlin Heidelberg, 2007.

[83] A. E. Selim, M. A. Wahed, and Y. M. Kadah, “Machine learning methodologies in brain-computer interface systems,” in *2008 Cairo International Biomedical Engineering Conference*, pp. 1–5, Dec 2008.

[84] S. C. Jugade, A. C. Victorino, and V. B. Cherfaoui, “Human-intelligent system shared control strategy with conflict resolution,” in *2018 IEEE 14th International Conference on Control and Automation (ICCA)*, pp. 686–691, June 2018.

[85] P. Inigo-Blasco, F. D. del Rio, S. V. Diaz, and D. C. Muniz, “The shared control dynamic window approach for non-holonomic semi-autonomous robots,” in *ISR/Robotik 2014; 41st International Symposium on Robotics*, pp. 1–6, June 2014.

[86] A. C. Lopes, U. Nunes, L. Vaz, and L. Vaz, “Assisted navigation based on shared-control, using discrete and sparse human-machine interfaces,” in *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, pp. 471–474, Aug 2010.

[87] G. Peinado, C. Urdiales, J. M. Peula, M. Fdez-Carmona, R. Annicchiarico, F. Sandoval, and C. Caltagirone, "Navigation skills based profiling for collaborative wheelchair control," in *2011 IEEE International Conference on Robotics and Automation*, pp. 2229–2234, May 2011.

[88] C. Urdiales, J. M. Peula, M. Fdez-Carmona, C. Barrué, E. J. Pérez, I. Sánchez-Tato, J. C. del Toro, F. Galluppi, U. Cortés, R. Annichiaricco, C. Caltagirone, and F. Sandoval, "A new multi-criteria optimization strategy for shared control in wheelchair assisted navigation," *Autonomous Robots*, vol. 30, pp. 179–197, Feb 2011.

[89] H. Wang and X. P. Liu, "Adaptive shared control for a novel mobile assistive robot," *IEEE/ASME Transactions on Mechatronics*, vol. 19, pp. 1725–1736, Dec 2014.

[90] P. Trautman, "Breaking the human-robot deadlock: Surpassing shared control performance limits with sparse human-robot interaction," in *Robotics: Science and Systems*, 2017.

[91] P. Trautman, "Assistive planning in complex, dynamic environments: A probabilistic approach," *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 3072–3078, 2015.

[92] A. D. Dragan and S. S. Srinivasa, "A policy-blending formalism for shared control," *Int. J. Rob. Res.*, vol. 32, pp. 790–805, June 2013.

[93] A. D. Dragan and S. S. Srinivasa, "Formalizing assistive teleoperation," in *Robotics: Science and Systems*, 2012.

[94] H. Maske, G. Chowdhary, and P. R. Pagilla, "Intent aware shared control in off-nominal situations," in *2016 IEEE 55th Conference on Decision and Control*, pp. 5171–5176, 2016.

[95] V. Honing, T. L. Gibo, R. J. Kuiper, and D. A. Abbink, "Training with haptic shared control to learn a slow dynamic system," in *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 3126–3131, Oct 2014.

[96] B. Skinner, *Science And Human Behavior*. A Free Press paperback, Free Press, 1953.

[97] R. Miltenberger, *Behavior Modification: Principles and Procedures*. Belmont, CA: Cengage Learning, 2015.

[98] M. Quigley *et al.*, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[99] P. R. Pagilla and B. Yu, "Robotic surface finishing processes: Modeling, control, and experiments," *Journal of Dynamic Systems Measurement and Control-transactions*, vol. 123, pp. 93–102, 2001.

[100] S. Niknam, B. Davoodi, P. Davim, and V. Songmene, "Mechanical deburring and edge-finishing processes for aluminum parts: a review," *The International Journal of Advanced Manufacturing Technology*, vol. 95, pp. 1101–1125, 2018.

[101] R. Bogue, "Finishing robots: a review of technologies and applications," *Industrial Robot: the international journal of robotics research and application*, vol. 36, no. 1, pp. 6–12, 2009.

[102] Y. Zhang, L. Xie, Z. Zhang, K. Li, and L. Xiao, "Real-time joystick control and experiments of redundant manipulators using cosine-based velocity mapping," in *IEEE International Conference on Automation and Logistics*, pp. 345–350, 2011.

[103] H. Arioui, L. Temzi, and P. Hoppenot, "Force feedback stabilization for remote control of an assistive mobile robot," in *Proceedings of the American Control Conference*, pp. 4898–4903, 2011.

[104] A. Singh, S. Seo, Y. Hashish, M. Nakane, J. Young, and A. Bunt, "An interface for remote robotic manipulator control that reduces task load and fatigue," *IEEE Conference on Robot and Human Interactive Communication*, pp. 738–743, 2013.

[105] M. Marinho, A. Geraldes, A. Bó, and G. Borges, “Manipulator control based on the dual quaternion framework for intuitive teleoperation using kinect,” in *Brazilian Robotics Symposium and Latin American Robotics Symposium*, pp. 319–324, 2012.

[106] F. Kulakov, G. Alferov, and P. Efimova, “Methods of remote control over space robots - seventh polyakhov’s reading,” in *International Conference on Mechanics*, pp. 1–6, 2015.

[107] J. Webb, S. Li, and X. Zhang, “Using visuomotor tendencies to increase control performance in teleoperation,” in *American Control Conference*, pp. 7110–7116, 2016.

[108] F. Park, “Computational aspects of the product-of-exponentials formula for robot kinematics,” *IEEE Transactions on Automatic Control*, vol. 39, pp. 643–647, 1994.

[109] K. Okamura and F. Park, “Kinematic calibration using the product of exponentials formula,” *Robotica*, vol. 14, pp. 415–421, 1996.

[110] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.