Experimental Analysis on Collaborative Human Behavior in a Physical Interaction Environment

by

Pallavi Shrinivas Shintre

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved November 2020 by the
Graduate Supervisory Committee:

Wenlong Zhang, Chair
Jennie Si
Yi Ren

ARIZONA STATE UNIVERSITY

December 2020

ABSTRACT

Daily collaborative tasks like pushing a table or a couch require haptic communication between the people doing the task. To design collaborative motion planning algorithms for such applications, it is important to understand human behavior. Collaborative tasks involve continuous adaptations and intent recognition between the people involved in the task. This thesis explores the coordination between the human-partners through a virtual setup involving continuous visual feedback. The interaction and coordination are modeled as a two-step process: 1) Collecting data for a collaborative couch-pushing task, where both the people doing the task have complete information about the goal but are unaware of each other's cost functions or intentions and 2) processing the emergent behavior from complete information and fitting a model for this behavior to validate a mathematical model of agent-behavior in multi-agent collaborative tasks. The baseline model is updated using different approaches to resemble the trajectories generated by these models to human trajectories. All these models are compared to each other. The action profiles of both the agents and the position and velocity of the manipulated object during a goal-oriented task is recorded and used as expert-demonstrations to fit models resembling human behaviors. Analysis through hypothesis teasing is also performed to identify the difference in behaviors when there are complete information and information asymmetry among agents regarding the goal position.

DEDICATION

To my family and my fiance, who have always inspired me to work hard and let me do whatever I wished to do. I would not be able to accomplish this without your support and patience with me.

ACKNOWLEDGMENTS

I would like to acknowledge my advisor, Dr.Wenlong Zhang for guiding me and providing valuable insights throughout my research at the RISE lab. I am deeply thankful to Yiwei Wang for his continuous support and leadership throughout the project. I would also like to acknowledge YiZhuang Garrard and Shatadal Mishra for guiding me throughout the process of software design of the experimet. I would further like to extend my gratitude towards Sunny Amatya, Venkatesh Vaidyanathan, Bradley Edward Goldberg for helping with the experiment conduction and data collection. I would also like to thank the members of RISE Lab for helping with the alpha testing of the experimental setup and also for being a strong support system throughout. I would like to thank all my roommates for their constant support throughout my Masters course. I would like to thank Karishma Patnaik, for being a constant motivator and an inspiration whenever I needed it. I would like to thank my committee, Dr. Jennie Si and Dr. Yi Ren for their invaluable comments and feedback on the work. Lastly, I would like to thank my parents, my family and my fiance for always believing in me.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Chapter 1

INTRODUCTION

1.1 Background

Human-robot interaction is an interesting field of study that tackles the problem of modeling human and engineered behaviors. Humans are very complex entities. It is not easy to predict their actions or intentions accurately. Therefore, designing robotic devices that can work with humans is a difficult problem. Previously, robotics researchers have worked with the problem of human-robot interaction that involves the prediction of human behaviors for effective and safe control of robotic devices interacting with them [2; 3]. A big challenge in this problem is intent recognition and generating an accurate response to it. This process involves signaling and decision making between the involved agents [4; 5]. Intent can be anything ranging from strategy (cost function) involved for reaching the goal, the goal position, hidden obstacles only known to a few agents, etc. However, intent detection requires the agents to have some prior knowledge about the motion planning framework of their partner and also a medium of communication between them [6]

There has also been previous research on using interaction models from economic theories to simulate multi-agent interaction behaviors [7; 8; 9; 10]. While these behaviors work well with simulations and application to multi-robot interaction scenarios, there has not been enough work on verification of these behaviors with multi-human interactions. It is important to understand and mathematically model multi-human interactions, to apply the theoretical algorithms that work in the case of multi-robot interactions. The behavior of humans collaborating with other humans would be different from them collaborating with robotic devices. However, to design robots that collaborate with humans, modeling

human-human interaction behavior is an important step. The behavior of humans can be expected to follow a similar model when they interact with robots, though not be the same as the human-human collaboration models.

1.1.1 Modelling Human Behavior in Collaborative Tasks

The following presents a review of work related to modeling human behavior in general as well as in human-robot interaction tasks.

The problem of modeling human behavior has been widely explored across different tasks such as load transport, walking and dancing, and also to understand human behavior in human-robot interaction scenarios [11; 12; 13; 14; 15]. Humans tend to build prediction models during a task to complete the task in a fast, accurate, and optimized manner [16]. A data-driven approach for robots to learn from human demonstrations and to model human behavior is proposed in [17]. Modeling roles such as leader-follower and teacher-student for cases involving information asymmetry in human-robot teams has been explored and studied by [6; 18]. Improvements like negotiations based on haptic feedback to improve interaction behaviors in virtual human-computer interaction tasks have been suggested by [19]. A real-time continuous scenario based on the existing game-theory and negotiation literature has been used in their work. Visualization algorithms for searching, reinforcement learning, and computational game theory for an extensive form of games has been discussed by [20]. The work discusses games that are designed based on partially and fully observable scenarios. Besides the game theory approach, Bayesian Estimation has been used to model the humans in [21]. Recursive Bayesian filtering approach models and the use of multiple non-verbal observations to identify the user's goal has been presented in [22]. Another Bayesian method to estimate human intent in a human-robot collaborative task has been proposed by [23].

The use of theory of mind to model human behaviour has been proposed by [1] (refer to figure-1.1) and [24]. They have suggested that humans make decisions based on certain objective functions. The theory of mind concept has been used in [25] to solve problems with purely cooperative games with two to five players and imperfect information. A comparison between model-free, black-box model-based, and Theory-of-Mind-based methods for human-robot interaction in the driving domain has been presented by [26].

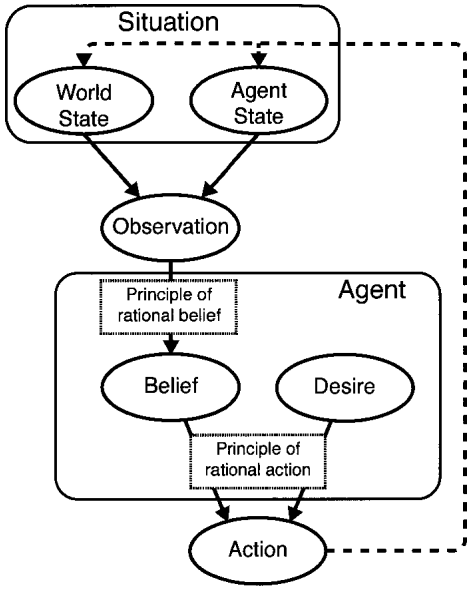


Figure 1.1: Concept of theory of mind as illustrated in [1]. This concept has been extensively used to model human behaviors in several works.

Concepts describing human behavior in collaborative tasks, e.g. Bayesian Persuasion [27]. Hidden Markov Models [28], theory of mind [29] etc also explain patterns in human behaviour in collaborative tasks.

1.1.2 Inverse Optimal Control to Model Expert Behaviours

There has been a lot of work on the theoretical development of multi-agent interaction algorithms. Human behavior can be described as a solution to an optimization problem [30]. It could be the minimization of a cost function or maximization of a reward function. The cost/reward function is a function of the features that the agent controls for optimally

achieving the goal. The features are task-specific. They can include anything ranging from visual feedback, auditory feedback to force, or haptic feedback [31; 32]. Similarly, the control can be in the form of continuous or discrete inputs. Inputs can also be in the form of speech, gaze, physics quantities such as force or velocity setpoints, or written/typed messages [33; 34]. Each of the features has a weight assigned to it (which may differ for different agents, be it a human or an autonomous agent). The weight determines how much importance is given to a feature. These cost/reward functions can be linear, quadratic, or higher orders based on how the features are related to the actual cost function [34; 31]. Modeling the agent behavior using the inverse optimal control approach is a task of finding the best weights assigned to the features that result in a trajectory that has the closest resemblance to the agent's trajectory. In this problem, the optimization framework remains constant. This means that the task-specific features remain the same. Also, the nature of the cost function (linear/quadratic) also remains the same. This problem can be stated as inverse optimal control or inverse reinforcement learning.

Inverse optimal control is ultimately an optimization problem. This optimization problem runs the forward optimal control loop at every iteration. The forward optimal control is a function of the weights therefore the inverse optimal control is also a function of the weights. The optimizer fits these weights to replicate the reference behavior. Existing literature in inverse optimal control attempts to fit cost functions to imitate human trajectories [35; 36]. However, using the inverse optimization technique to describe more complex forward problems like multi-agent co-ordination behaviors has not been explored.

Similarity of a fitted model with the expert (agents') behavior can be evidence to show that an agent uses the same features as selected for the inverse problem. It also shows that while the values of weights may be different, the forward model can be generalized as a baseline framework used by most of the agents doing the task. To obtain the expert trajectories, 'humans' in our case, conduction of human-human collaboration experiments is

required. The design of these experiments must involve intent-recognition and decoding, as the main aim is to model collaborative multi-agent behaviors. A lot of prior work involves experimental analysis for studying human interaction patterns. There have been studies based on physical as well as virtual experimental setups for the collection of interaction data.

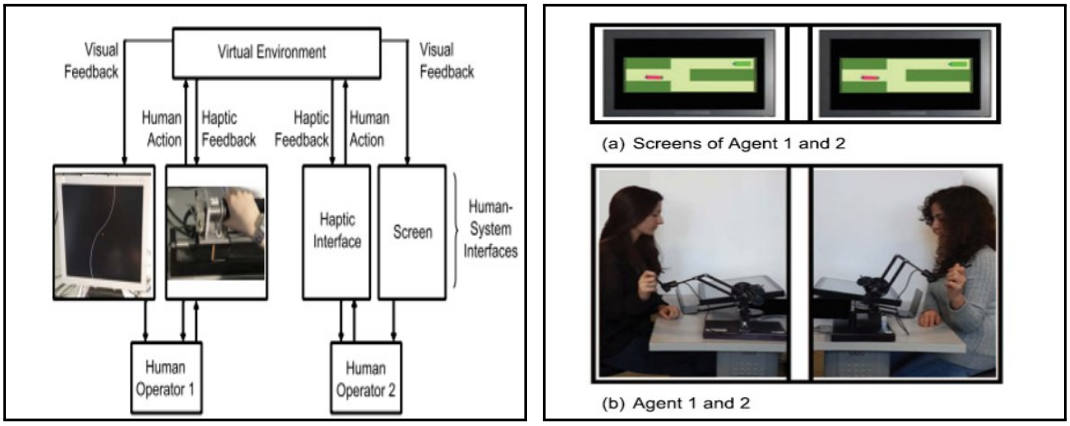
1.1.3 Virtual Human in the Loop Experiments

This portion discusses a review of the existing work concerning virtual experiments involving human subjects.

A shared-control interaction methodology has been described in [37; 38] using haptic interface systems to collect experimental data. The virtual experimental setup must contain a haptic device to collect the user input. A technique of obtaining continuous user input from the NOVINT Falcon Haptic device, one of the haptic devices suitable for such experiments has been shown in [39].

Study of human-human interaction using the haptic interface for assistance and training can be found in [40]. A similar system was used to understand the haptic interactions between two people elucidating the behavior of an individual working alone has been used in [41]. The work sheds light on understanding how humans can intuitively and cooperatively work with a robot on physical tasks. An example of how a system can be used to perform dynamic tasks, by providing real-time visual and haptic feedback through a virtual environment can be found in [42; 43]. Studying the distribution of authority and the role allocation in multi-human teams performing the collaborative task has also been a subject of interest [44]. Haptic interaction patterns in such a shared virtual environment have been studied in both collaborative and conflicting decision situations in [45; 46]. A virtual setup to collect human inputs in human-human interaction tasks have been presented in both of these works (refer to figures-1.2(a)), 1.2(b).

Furthermore, a subjective analysis of the performance measures in such an interaction pattern has also been performed in [47]. An analysis that associates the energy flow between interacting partners with role distributions has been presented in [48].



(a) Virtual experiment setup by [46] (b) Virtual experiment setup by [45]

Figure 1.2: Visualization of virtual experiment setups from reviewed literature

1.2 Motivation and Overview

It can be observed from the review of literature that there has been a substantial amount of work in mathematical model development of multi-agent collaborative systems. A lot of work has also been done on analyzing human-human interaction with hypothesis proving. Researchers have also worked on understanding the optimization features that humans consider during tasks. Work has also been performed on building robot models that work in human-robot teams with conduct safer, more reliable performance. However, work on building human models in human-human interaction behaviors based on the theoretical multi-agent models has not been found in the reviewed literature. Modeling the behavior of humans when they collaborate with other humans is an important step to develop robots that have effective prediction models of collaborative human behavior. Such robots would be able to work more efficiently on collaborative tasks with human partners.

This thesis explores the process of building mathematical models that resemble collaborative human-human behavior. To perform analysis that is the most relevant to this study, available data from open source resources could not be used. Therefore, data collection is necessary. The first step is designing an experiment to conduct collaborative human-human experiments.

A large advantage of designing virtual experiments is that the environment is fully controlled and therefore simpler to model. The design also demands the incorporation of a mode of interaction. The task is designed such that the collaborating agents get other agents' feedback through the states governed by the underlying joint dynamics. The object to be manipulated is controlled by both the agents and can be manipulated to the goal only if the agents show active collaboration throughout the task. The interaction framework is adapted from [34] which also uses underlying joint dynamics for interaction between the agents.

Pre-requisites of the experiment conduction include the development of a virtual platform for experiment conduction, preliminary testing of the setup, automating the experimental procedure as much as possible for the efficiency of conduction and application, and approval of an IRB for conducting the study with human subjects.

After the pre-requisites for the experiment conduction are met, data collection with human subjects must be performed, followed by the analysis of the data for model development. To begin modeling and simulation of the high-level interaction dynamics between the agents, a low-level dynamic model (physics) of the UNITY physics engine is required. Therefore, system identification of the dynamic model in the UNITY engine is performed. Data for different input patterns simulated for long periods is collected and used for the analysis. Once the low-level dynamic model is available, the work proceeded to the optimal control framework formulation and identification of the cost functions from it (inverse optimal control).

The experimental framework for simulating multi-agent collaboration is designed to receive constant feedback from the states and find optimal inputs for each agent, separately. The agents find Nash equilibrium input sets from the action candidate set and apply these optimal inputs. Nash equilibrium is a concept from game theory. When multiple agents are involved in a game, the best inputs that can be applied by these agents are called Nash equilibrium sets [49].

Model fitting using the above framework (baseline) began with the implementation of a baseline inverse optimal control approach to model the agent behavior (cost functions of the participating agents) that resulted in trajectories closest to human trajectories. Further amendments were made to this model to improve the error between trajectories generated by these models and the human trajectories. Amendments included changing the fitness function to capture the trends in the human trajectories rather than trying to replicate them. Another amendment is to design every agent to predict their and their partner's inputs using different cost functions. Therefore, each agent would consider that their partner doesn't have the same decision-making framework as themselves. The final amendment is, incorporating a stochastic component in the input selection of each agent. Analysis and comparisons between results obtained from several such amendments are performed, to make conclusions about the model that can describe human behavior most closely.

The rest of the document is organized as follows. The experimental and hardware setup used for data collection is described under Chapter 2. Chapter 3 discusses the procedure for pre-processing the data for analysis which involves system identification of the UNITY physics engine. Formulation of the human-model fitting problem is explained in Chapter 4, followed by the analysis, Chapter 5 contains the conclusion, summary, and future work.

Chapter 2

EXPERIMENTAL AND HARDWARE SETUP

An experimental study was planned to analyze human behavior in a collaborative setup. A collaborative couch-pushing experiment was designed and a virtual setup was developed to conduct this experiment. This section presents the experiment setup and design used for this study as well as the procedure followed for the experiment conduction.

2.1 Experimental Setup

The visual feedback for each participant and the physics engine for simulating the dynamics was developed in the UNITY environment. The setup of the experiment has been shown in figure 2.1. The grasping points on the object for both participants were displayed as blue and yellow colored handles. The current frame and the desired end position were visually presented on the display screen for both the agents. The final configuration was displayed as a colored block, the same size as the manipulated object. For the scenes with complete information scenarios, it can be observed that the final (goal) configuration is a green colored block.

2.1.1 Hardware Setup

Both agents were able to apply their inputs orthogonal to the object at the given grasping point present at the opposite ends of it. Each of the participants was asked to apply input forces on the knob of the respective NOVINT Falcon Haptic device provided to them. PD feedback control was implemented to convert the force inputs applied on the knob of the haptic device to input values for the UNITY physics engine. The participants were able to feel a sense of difficulty applying higher inputs than the lower ones due to the addition

of damping factor (PD control). For all the experiments, the interface was connected to the respective computer for visual feedback during the task. The Falcon's sensors can keep track of the handle's position to sub-millimeter resolution, and the motors are updated 1000 times per second ($1kHz$), giving a realistic sense of touch.

The NOVINT Falcon Haptic Device

The knob of the haptic device can be displaced laterally along the x, y, and z directions [39]. A neutral position $= \begin{bmatrix} x_{neutral} & y_{neutral} & z_{neutral} \end{bmatrix}$ is set in terms of co-ordinates of the knob. The PD feedback control is used to convert the displacement and velocity of the knob with respect to the neutral position to the input value for the UNITY physics engine (explanation of inputs and dynamics of the UNITY physics engine is given in section-3). The vectors of proportionality constants, K_p and derivative constants, K_d are set as $K_p = \begin{bmatrix} 100 & 1000 & 1000 \end{bmatrix}$ and $K_d = \begin{bmatrix} 10 & 10 & 10 \end{bmatrix}$.
Let $p(t) = \begin{bmatrix} x(t) - x_{neutral} & y(t) - y_{neutral} & z(t) - z_{neutral} \end{bmatrix}'$ be the position vector of the knob with respect to the neutral position and let
$d(t) = \begin{bmatrix} \frac{(x(t)-x(t-1))}{\Delta t} & \frac{(y(t)-y(t-1))}{\Delta t} & \frac{(z(t)-z(t-1))}{\Delta t} \end{bmatrix}'$ be the first order derivative of the position (velocity) of the knob. Therefore, at time t, the output of the PD control, which is fed as an input to the UNITY physics engine, $U(t) = \begin{bmatrix} u_x(t) & u_y(t) & u_z(t) \end{bmatrix}'$ is calculated as,

$$U(t) = K_p \cdot p(t) + K_d \cdot d(t) \tag{2.1}$$

The computed input values were fed to two locations.

1. The UNITY environment where they were treated as velocity set points to the two ends of the virtual object.

2. The internal control system of the haptic device, which requires continuous feedback of the inputs applied to it.

The inputs were manipulated using thresholds (clipped) to obtain discrete values before they were sent to the UNITY environment using a zero MQ publish-subscribe communication protocol. The thresholds applied to the calculated values before they were published to the UNITY physics engine were as follows. The inputs were discretized to simplify the model development which was to be done for the collected data. The clipped inputs were obtained as follows:

- 2.1 for a large displacement of the knob of the haptic device
- 1.4 for a medium displacement of the knob of the haptic device
- 0.7 for a small displacement of the knob of the haptic device

To add constraints to the movement of the participants, the movement along the y and the z axis was disabled. Therefore, the inputs applied by the participants are only in the x-dimension (Figure 2.2). These inputs were applied perpendicular to the ends of the object (the couch) in the UNITY scene. See Figure-2.1.

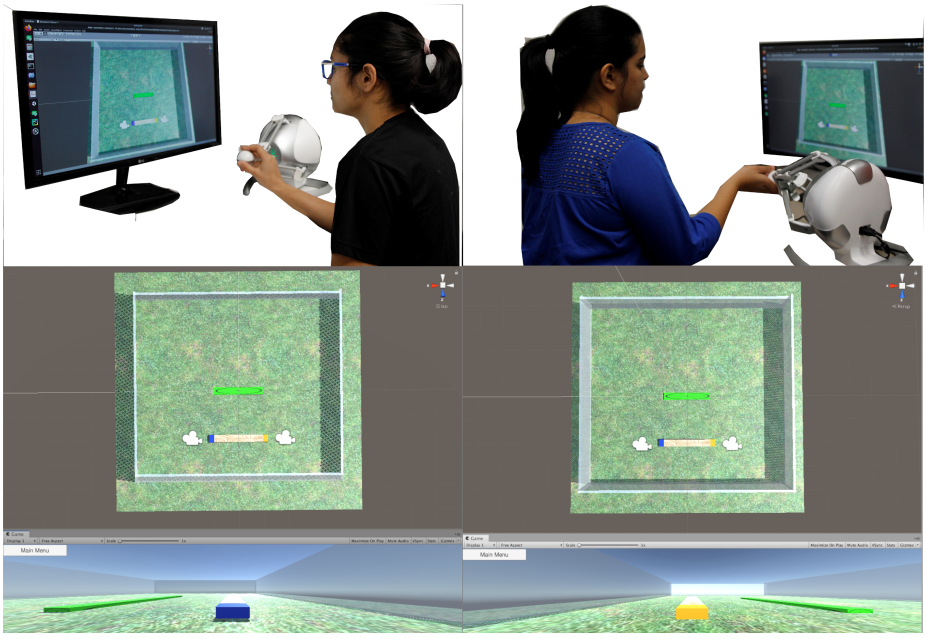


Figure 2.1: Experimental setup consists of two participants, provided with their own haptic devices for input application. The participants are seated facing away from each other. Each participant is also provided with their own screen for visual feedback of the task.

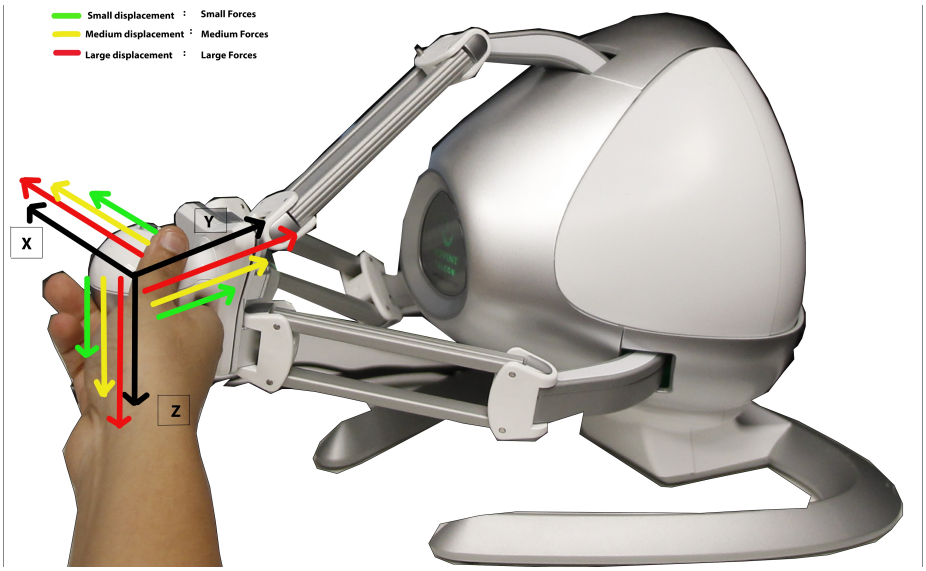


Figure 2.2: Illustration of the magnitude and direction of Forces for the NOVINT Falcon Haptic device. The displacement of the knob is directly proportional to the magnitude of Forces applied on the knob.

2.1.2 Software Setup

The communication of data between the haptic device, the scripts for the objects in the Unity scenes, and the python program to record the data was achieved using a robust IoT (internet of things) library available in all the languages used, 'zero MQ'. To avoid delay in data communication and saturation of the topics to which the data is published and subscribed, multi-threading was implemented for all these processes that run in parallel.

The software for the visual feedback was implemented in the UNITY environment. The physics engine would receive the discrete (clipped) input values (velocity setpoints) from the Falcon over a server-client protocol and apply them to the object to simulate the physical dynamics. The control algorithm for the system was run at a frequency of $1kHz$.

Data Collection

A python script was used for data collection. Input values from both the participants, states, and timestamps were collected for each trial. The frequency for data collection was set as $25Hz$.

2.2 Experiment Scenarios

To analyze interaction patterns, different scenarios for collaborative physical interaction were designed. The task demands two people to move an object to a goal location. The task was designed such that one agent could not complete the task alone. The goal location consisted of a pre-determined position and orientation. The tasks can be categorized into two broad categories, complete information, and information asymmetry.

For complete information, both the participants were provided with the target location. For information asymmetry, the potential target locations were displayed for both participants, however, only one participant was aware of the exact goal from these options. The

design ensured that for all the scenarios, participants had to collaborate to complete the task. In the case of information asymmetry, the participant unaware of the exact goal had to learn it and the participant aware of it had to teach it to their partner during the task. 10 trials were conducted for each task. Following is a summary of the different scenarios.

2.2.1 Complete Information

In these scenarios, both participants are provided with the target location (goal). Hence, no conflict in terms of the final goal/ target is expected.

- **Translation**: Figure 2.3(a) represents the screen visual shown to each participant where they have to find the best combination inputs applied at a given time to reach the goal accurately in a time-frame of 10 seconds. This scenario is designed to have a goal that can be reached by a collaborative translation of the object.

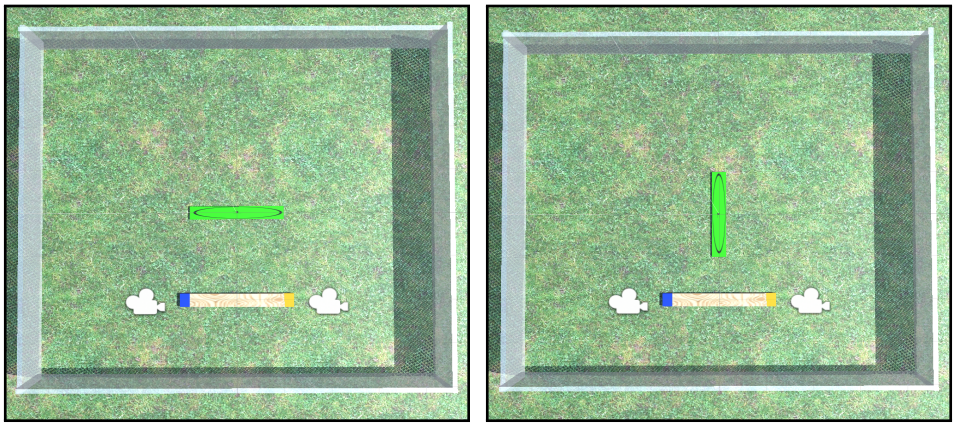
- **Translation and Rotation**: This scenario is designed to have a combination of translation and rotation goals. Here the participants have to collaboratively translate and rotate to reach the end position as seen in Figure 2.3(b), in a time-frame of 10 seconds.

2.2.2 Information Asymmetry

Here, both the participants are not presented with complete information about the goal. Possible end positions are presented in the scenario however only one participant is made aware of the end goal. The participant who knows the goal remains constant in all the experiments. As a result, the participant without the knowledge of the exact goal must learn it while the two collaborate to accomplish the target.

- **Information Asymmetry Translation**: Figure 2.4(a) represents the screen shown to each participant where the possible goal positions are presented in green, white, and orange blocks. One of the agents is given information about the exact goal location. The pair is asked to reach the goal in the time frame of 10 seconds.

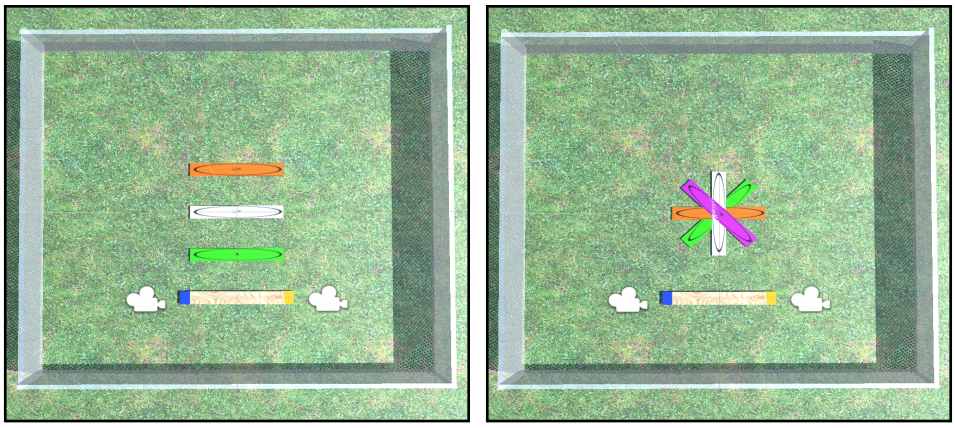
- **Information Asymmetry Translation and Rotation**: A translation and orientation-based possible goal positions are presented in green, white, pink, and orange blocks as seen in Figure 2.4(b). The same participant as for the pure translation case is given the information about the exact goal-location. Again, the experiment must be completed in a 10 seconds time-frame.



(a) Scenario for translation goal with complete information distribution among agents

(b) Scenario for translation as well as rotation goal with complete information distribution among agents

Figure 2.3: Experiment scenarios: Complete Information



(a) Scenario for translation goal with information asymmetry among agents

(b) Scenario for translation as well as rotational goal with information asymmetry among participants

Figure 2.4: Experiment scenarios: Information asymmetry (Only one participant knows the exact goal option while both the participants know the goal options or the goal candidate set as shown above)

2.3 Procedure and Participation

The study was conducted after a successful application and approval from the Institutional Review Board (IRB) at Arizona State University. The study was registered with the IRB ID: *STUDY*00011502. A total of 30 participants (paired to form 15 teams) volunteered to take part in the experiment. It was ensured that the recruited participants were adults and students at Arizona State University. The participation was voluntary and the participants were permitted to leave in case they experienced discomfort. On successful completion of the experiment, they were rewarded for their participation in the form of gift cards to a local coffee-shop. The participants were informed about this reward before the task to ensure their complete attention during the experiment. The collected data was re-named to ensure that it remained de-identified. The entire experiment for a pair was conducted on the same day in an hour.

The participants were first given detailed instructions regarding experiment conduction. They were then allowed to individually familiarize with the setup. This allowed them to understand the working of the hardware. To avoid over-familiarization, they were only allowed to spend a few minutes with the setup. They were then instructed to avoid any form of verbal communication with their partners. At least two instructors were present at every trial to conduct the trial effectively.

Then, the participants were briefed about the complete information scenarios. After the briefing, the trials for complete information scenarios were conducted. Each scenario was run for 10 trials. The trial number was verbally announced before starting the trial. After every trial, the participants were asked to fill out a form, evaluating their collaborative performance for the trial. After completing the complete information case scenarios, the participants were briefed about the information asymmetry scenarios. In this section of the experiment, the participant aware of the goal position remained constant to avoid confusion. Again, 10 trials were conducted for each scenario and both the participants were asked to fill out an evaluation sheet after every trial.

Chapter 3

PRE-PROCESSING THE DATA FOR ANALYSIS

To model collaborative behavior, an inverse modeling algorithm must be formulated. The inverse algorithm requires the dynamics of the system. Therefore, system identification of the virtual physics engine in UNITY was necessary.

3.1 System Identification of the Virtual Physics Engine

The system consists of two inputs on either side of the object (the couch) that were applied orthogonal to it. The inputs were identified to be in the form of velocity setpoints (u_1 and u_2). The two human participants would apply forces on the knob of their respective haptic devices, which were used to compute their respective inputs to be published (sent) to the UNITY physics engine as input velocity set-points for the two ends of the object (as described in section-2.1.1). The linear positions and velocities (for x and y) and the angular position and velocity of the midpoint of the object were dependants on the combination of inputs applied by the participants to the object.

3.1.1 Derivations for the Velocity Set-Points

The equations describing the relationship between the input velocity setpoints (v_1 and v_2) applied to the two ends of the object and the resultant angular and linear velocities of the center of the object (v_x, v_y and ω) can be derived as follows.

Derivation for the angular velocity set point

First, the equation for the angular velocity at the center of the object is derived, given the input velocity set points at both the ends of the object. The problem is explored in two possible cases.

When Inputs Are Applied in the Opposite Direction

Please refer to figure 3.1(a). The red dot is the center of rotation. First, the center of rotation is calculated, followed by calculation of the radii of the rotation (r_i) for both the inputs, v_i ($i = 1, 2$). The radius of rotation is the distance between the center of rotation and the point at which the input is applied. Let the radius of rotation for the end at which v_2 is applied, be denoted by x (refer to the figure-3.1(a)). Therefore, $x = r_2$. The angular velocity is defined as the linear velocity applied orthogonal to the radius divided by the radius of rotation, ($\frac{v_2}{x}$). It can be stated that if unequal input velocities are applied to the object in opposite direction, the sum of the radii of rotation is equal to the length of the object (l) and that the ratio between the radii of rotation would be equal to the ratio of the applied velocities, as the angular velocity remains constant throughout the rigid body.

Hence, $\frac{v_i}{r_i}$ remains constant for both v_1 and v_2. Therefore, $\frac{v_1}{r_1} = \frac{v_2}{r_2}$, where r_1 and r_2 are the radii of rotation for the points where the velocities v_1 and v_2 are applied respectively. (In this case, the center of rotation is present on the object).

Considering that the anticlockwise direction is positive, the formulation could be described as follows.

$$\frac{v_1}{l-x} = \frac{-v_2}{x} \tag{3.1}$$

... as r_2 = x, $r_1 = l - x$. Therefore,

$$\frac{v_1}{-v_2} = \frac{l-x}{x} \tag{3.2}$$

$$\frac{v_1 - v_2}{-v_2} = \frac{l-x+x}{x} = \frac{l}{x} \tag{3.3}$$

... by adding 1 on both sides.

Therefore,

$$x = \frac{l \times -v_2}{v_1 - v_2} \tag{3.4}$$

Therefore, angular velocity (constant throughout the rigid body) can be calculated as,

$$w = \frac{v_i}{r_i} = \frac{-v_2}{r_2} = \frac{-v_2}{x} = \frac{-v_2 \times (v_1 - v_2)}{l \times -v_2} = \frac{(v_1 - v_2)}{l} \tag{3.5}$$

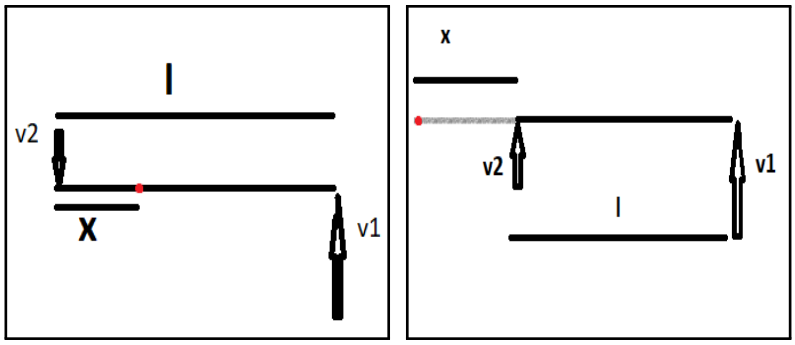
When the inputs are applied in the same direction

When the inputs are applied in the same direction but have different magnitudes. Please refer to figure 3.1(b). The red dot is the center of rotation. In this case, the center of rotation is present outside the object. Again, anticlockwise direction is considered as positive. The ratio of the radii is equal to the ratio of the input velocities, as $\frac{v_1}{r_1} = \frac{v_2}{r_2}$, where r_1 and r_2 are the radii of rotation for the points where the velocities v_1 and v_2 are applied respectively, and let $r_2 = x$.

$$\frac{v_1}{l-x} = \frac{-v_2}{x} \tag{3.6}$$

... as r_2 = x, $r_1 = l + x$. Therefore,

$$\frac{v_1}{v_2} = \frac{x+l}{x} \tag{3.7}$$



(a) Inputs applied in opposite direction

(b) Inputs applied in same direction with unequal magnitude

Figure 3.1: Visualization of the center and radii of rotation for the different cases of inputs. l is the length of the object and x is the distance between the center of rotation and the point on the object where the input-2 is applied.

$$\frac{v_1 - v_2}{v_2} = \frac{x + l - x}{x} = \frac{l}{x} \tag{3.8}$$

...subtracting 1 on both sides of equation-3.7.

Therefore,

$$x = \frac{l \times v_2}{v_1 - v_2} \tag{3.9}$$

Therefore, angular velocity (constant throughout the rigid body) can be calculated as,

$$w = \frac{v_i}{r_i} = \frac{v}{r} = \frac{v_2}{x} = \frac{v_2 \times (v_1 - v_2)}{l \times v_2} = \frac{(v_1 - v_2)}{l} \tag{3.10}$$

Therefore, it can be concluded that irrespective of the direction of the inputs, the equation for angular velocity remains $\omega = \frac{(v_1 - v_2)}{2}$.

As the angular velocity of every point in a rigid body remains same, according to the properties of a rigid body, the above derivation proves that,

$$\omega_{mp} = \frac{(v_1 - v_2)}{2} \tag{3.11}$$

where, ω_{mp} is the angular velocity at the mid point of the object. The value of ω_{mp} is determined, given the applied inputs, v_1, v_2.

Derivation for the linear velocity set point

Next, the equations for the linear velocities along x and y directions are derived for the center of the object, given the input velocity set points at the two ends of it. Refer to figure 3.2. From equation 3.11, the angular velocity at the midpoint of the object is known.

Considering properties of a rigid body with respect to velocities, it can be stated that if P and Q are two points on a rigid body, the the instantaneous linear velocity at point Q, given the instantaneous linear velocity at point P (v^P), the vector distance between the points P and Q (r^{PQ}) and the angular velocity of the rigid body (ω_{rb}) can be defined as,

$$v^Q = v^P + \omega_{rb} \times r^{PQ} \tag{3.12}$$

Let v_{mp} be the instantaneous linear velocity at the mid point of the object. Therefore,

$$v_{mp} = v_2 + \frac{(v_2 - v_1)}{l} \times \frac{l}{2} = v_2 + \frac{(v_2 - v_1)}{2} = \frac{2v_2 + v_1 - v_2}{2} = \frac{v_1 + v_2}{2} \tag{3.13}$$

However, calculation of the components of the linear velocity in x and y directions is required. The components can be defined as $v_x = vcos(\gamma)$ and $v_y = vsin(\gamma)$, where γ is the angle between the velocity vector and the x-axis. Here, the velocities are applied orthogonal to the horizontal as the object is placed along the horizontal. Therefore, the angle, $\gamma = \frac{\pi}{2} + \theta$, where θ is the angular position of the object (which can be calculated using the angular velocity ω_{mp} of the object).

Therefore, the components of instantaneous linear velocity of the mid point of the object, along the x and y directions can be calculated as,

$$v_x = v_{mp} cos\left(\frac{\pi}{2} + \theta\right) = v_{mp}\left[cos\left(\frac{\pi}{2}\right) cos(\theta) - sin\left(\frac{\pi}{2}\right) sin(\theta)\right] = -v_{mp} sin(\theta) \tag{3.14}$$

$$v_y = v_{mp} sin\left(\frac{\pi}{2} + \theta\right) = v_{mp}\left[sin\left(\frac{\pi}{2}\right) cos(\theta) + cos\left(\frac{\pi}{2}\right) sin(\theta)\right] = v_{mp} cos(\theta) \tag{3.15}$$

As $v_{mp} = \frac{v_1+v_2}{2}$,

$$v_x = -\left(\frac{v_1+v_2}{2}\right) sin(\theta) \tag{3.16}$$

$$v_y = \left(\frac{v_1+v_2}{2}\right) cos(\theta) \tag{3.17}$$

The equations 3.16 and 3.17 denote values of the linear velocities of the object along x and y, given the applied inputs v_1 and v_2.

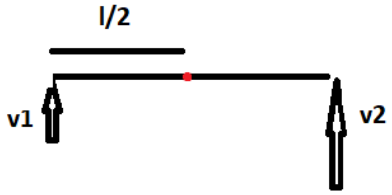


Figure 3.2: Visualization of the inputs and calculation of linear velocity at the center

From the above description and explanation, it can be stated that the set points for linear velocities along x and y and the angular velocity of the mid point of the object (according to the UNITY environment) are determined by the equations 3.16, 3.17 and 3.11. However, it was observed that the set points were not directly applied to the object at every time step. They were mapped through a first order dynamics equation before and then applied to the object (rigid-body in case of the UNITY environment).

3.1.2 System Identification

As described above, the system consists of two inputs on either sides of the object that were applied orthogonal to it. The inputs were in the form of velocity set points (u_1 and u_2). However, it was observed that these input velocities were not applied directly to the system, as mentioned above. They were mapped by a first order transfer function in the UNITY environment. The states can be defined as

$$s(k) = \begin{bmatrix} x(k) \\ y(k) \\ v_x(k) \\ v_y(k) \\ \theta(k) \\ \omega(k) \end{bmatrix} \tag{3.18}$$

The inputs (velocities) were denoted as $u_1(k)$ and $u_2(k)$. Here, $x(k)$, $y(k)$ are the x and y co-ordinates of the center of the object and $\theta(k)$ is the angular position of the object (rigid body) with respect to the horizontal, at the time step, k. $v_x(k)$, $v_y(k)$ are the velocities in the x and y directions and $\omega(k)$ is the angular velocity of the rigid body, at the time step, k. Please refer to figure-3.3 for the illustration of the states.

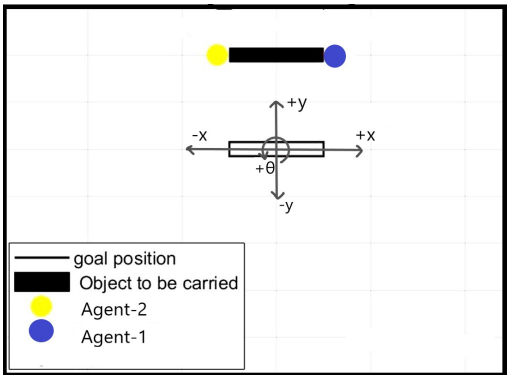


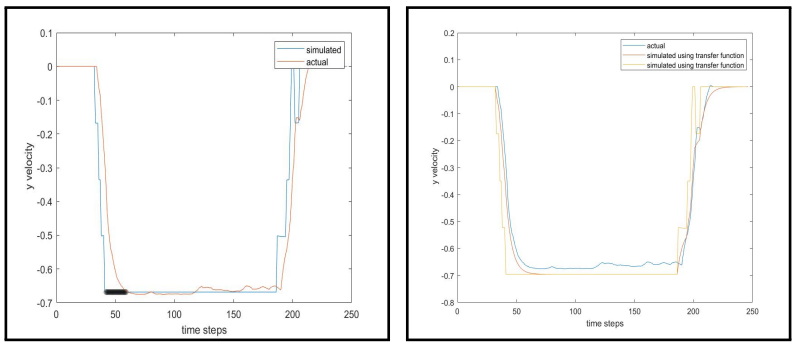
Figure 3.3: Illustration of states of the object

System identification was performed by collecting the data for all patterns of inputs. The system could be visualized in the state space as follows,

$$s(k+1) = As(k) + B_1(k)u_1(k) + B_2(k)u_2(k) \tag{3.19}$$

The value of A, B_1 and B_2 could only be computed after identifying this first order transfer function.

Several data sets were collected for different pairs of inputs from the available input candidate set. The inputs were applied at a stretch for 20 seconds. The figure 3.4(a) and 3.5(a) show the effect of the first-order transfer function, highlighted by a black line. The figure was plotted with the actual data and the calculated velocity value considering only a proportional mapping.

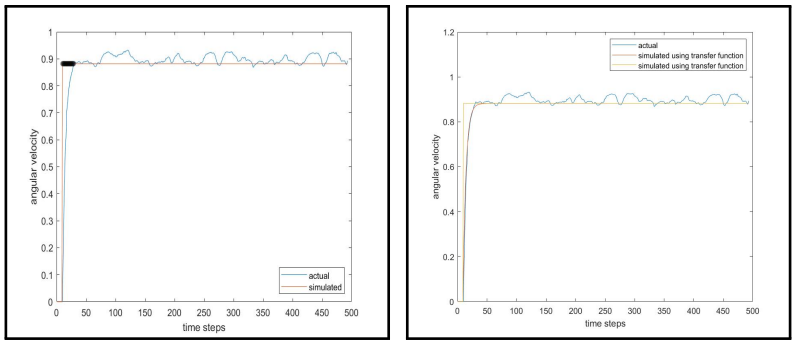


(a) The effect of first order transfer function on y- velocity

(b) Comparison between actual y-velocity and models with and without considering the first order transfer function

Figure 3.4: Illustration to show that linear velocity set-points are mapped through a first order transfer function. The inputs to the system are $u_1 = -1.4$, $u_2 = -1.4$

The structure of A, B_1 and B_2 was specified from the state transition equations (dependant on the equations-3.20, 3.21 and 3.22), but the values of the terms in the matrices could only be calculated after identifying the first-order transfer function. The identification of this first-order transfer function was carried out using the System Identification application,



(a) The effect of first order transfer function on angular velocity

(b) Comparison between actual angular velocity and models with and without considering the first order transfer function

Figure 3.5: Illustration to show that angular velocity set-points are mapped through a first order transfer function. The inputs to the system are $u_1 = 0$, $u_2 = -2.1$

from the system identification toolbox in MATLAB. The application requires an input vector and an output vector to be fed in for identifying a transfer function (this portion was black-box modeling). The input was given as the vector of the values of angular velocity, calculated according to the kinematic equation for angular velocity which has been derived in section-3.1.1 and the output vector fed to the application was recorded values of angular velocity (from the collected data). The values of angular velocity were calculated considering the inputs, $u_1(t)$ and $u_2(t)$, which are the velocity setpoints applied to the two ends of the object.

$$\omega(t) = \frac{u_1(t) - u_2(t)}{2} \tag{3.20}$$

While the basic structure of A, B_1 and B_2 was formulated considering the dynamics, with inputs as velocity set-points mapped by a first-order transfer function. As the identification of this first-order transfer function was done using black-box modeling, the system identification can be described as grey-box modeling.

The application returned the proportional term (k) and the pole (p) for the first-order transfer function, $H(s) = \frac{k}{s+p}$. This process was carried out for all the collected samples. An average value for k and p was calculated. It was observed that the transfer functions were different for linear velocity and angular velocity mappings. Also, the value for k in the case of the angular velocity calculation (k_r) was different for a few pairs of control inputs. While the difference between values of k was in the range of 10^{-2}, the response affected the value of the output. Therefore, for such exceptional cases, the value of k was set separately.

A similar test was performed for the terms in the linear velocity vectors, ($v_x(t)$ and $v_y(t)$) as well, considering the following equations as velocity setpoints for values of linear velocities along x and y. (Refer to section-3.1.1 for derivation of the equations)

$$v_x(t) = -\frac{(u_1(t) + u_2(t)) \times sin(\theta(k))}{2} \tag{3.21}$$

$$v_y(t) = \frac{(u_1(t) + u_2(t)) \times cos(\theta(k))}{2} \tag{3.22}$$

Refer to the table 3.1 and 3.2, for the computed k and p values and the efficiency of model fitting obtained for each of them. The table shows the efficiency of the fitted model for the particular input pair and the generalized model (considering the mean of all the k and p values), for the velocities in x and y directions and the angular velocity. The value of efficiency of the model is obtained from the system identification application's feature 'Fit to estimation data' for each of the fitted models. The generalized model is imported in the system identification application and and it's efficiency is obtained from the 'best fit' value in the 'model output' feature from the application. The efficiencies of the fitted model and generalized model for all types of input pairs has been presented in tables 3.1, 3.2.

The internal first order transfer functions in UNITY environment which maps the input set points for the velocities were therefore realized as follows. Let, v_x^{sp} and v_y^{sp} be vectors of calculated linear velocity set-points using equations-3.21, 3.22 and v_x^{data},v_y^{data} be the linear velocity vectors from the collected data. Similarly, let ω^{sp} be the vector of calculated angular velocities from the collected data and ω^{sp} be the angular velocity vectors from the collected data. Then,

1. For rotation,

$$H(s) = \frac{\omega^{data}(s)}{\omega^{sp}(s)} = \frac{k_r}{s+0.1866} \tag{3.23}$$

   The value of $k_r = 0.1553$ except for the exception cases, for which their respective k_r is considered.

2. For translation,

$$H(s) = \frac{v_x^{data}(s)}{v_x^{sp}(s)} = \frac{v_y^{data}(s)}{v_y^{sp}(s)} = \frac{0.0982}{s+0.2018} \tag{3.24}$$

Table 3.1: Values of k and p, obtained for corresponding pair of inputs using the system identification toolbox (MATLAB) for linear velocity from collected data for the system identification. Average of the k and the p values calculated ($k = 0.0982$ and $p = 0.2018$). Values used for formulation of generalized transfer function. Table contains efficiencies of the model fitted by system identification toolbox and the generalized transfer function.

Input (Value and direction)	Transfer function parameters		Model efficiency	
	k	p	fitted	generalized
small and small (same)	0.076	0.1679	92.31	83.63
medium and medium (same)	0.096	0.2	95.23	94.64
large and large (same)	0.093	0.19	96.75	96.48
zero and small	0.1189	0.211	86.77	83.2
zero and medium	0.1372	0.2651	92.6	89.5
zero and large	0.1314	0.258	95.4	92.5
small and medium (opposite)	0.1235	0.1445	88.13	85.58
small and large (opposite)	0.1155	0.2237	92.63	95.18
medium and small (opposite)	0.077	0.1613	86.25	85.54
medium and large (opposite)	0.1271	0.2406	89.45	85.49
large and small (opposite)	0.098	0.1977	93.52	93.17
large and medium (opposite)	0.093	0.1944	83.78	83.82
small and medium (same)	0.1	0.2	92.26	92.21
small and large (same)	0.08	0.1877	94.33	96.01
medium and small (same)	0.07	0.15	96.86	92.38
medium and large (same)	0.0877	0.1819	98.07	97.02
large and small (same)	0.093	0.193	98.22	98.09
large and medium (same)	0.076	0.1578	96.36	95.06

Table 3.2: Values of k and p, obtained for corresponding pair of inputs using the system identification toolbox (MATLAB) for linear velocity from collected data for the system identification. Average of the k and the p values calculated ($k = 0.1553$ and $p = 0.1866$). k value for the three exception cases as described above are set separately. For the rest of the cases, average values used for formulation of generalized transfer function. Table contains efficiencies of the model fitted by system identification toolbox and the generalized transfer function.

Input (Value and direction)	Transfer function parameters		Model efficiency	
	k	p	fitted	generalized
medium and medium (opposite)	0.1108	0.1694	84.27	60.64
large and large (opposite)	0.2195	0.2755	87.34	72.67
zero and small	0.1237	0.2	57.66	40.29
zero and medium	0.1404	0.1757	87.34	77.94
zero and large	0.1706	92.42	0.1241	85.7
small and medium (opposite)	0.1235	0.1445	93.4	85.3
small and large (opposite)	0.1871	0.2128	95	70
medium and small (opposite)	0.1313	0.1544	94.03	88.93
medium and large (opposite)	0.1487	0.1659	94.33	70.21
large and small (opposite)	0.1591	0.1811	94.59	78.06
large and medium (opposite)	0.1504	0.681	95.03	69.09
small and medium (same)	0.1475	0.171	82.8	77.38
small and large (same)	0.1168	0.1331	88.2	75.35
medium and small (same)	0.1421	0.169	79.36	78.9
medium and large (same)	0.1823	0.2103	77.78	88.72
large and small (same)	0.1667	0.1938	90.51	83.3
large and medium (same)	0.2199	0.2488	77.2	62.7

However, the above transfer functions were calculated in the Laplace domain. To formulate the equations for the propagation of the states, these transfer functions had to be converted to the z-domain, to match the discrete state space of the system.

The transfer functions are converted to the z-domain using the inbuilt MATLAB function, [transfer function in z] = c2d([transfer function in s, sampling time)]. The sampling time is considered as 1. This is because the continuous-time transfer function was computed using the already sampled data.

1. For rotation,

$$P[z] = \frac{m_r}{z - 0.8298} \tag{3.25}$$

The value of $m_r = 0.1417$ for all the input pairs, except for the exceptional cases for which their respective m_r is considered. The 4 exceptional cases where the value of m_r is different than 0.1417 are as follows,

(a) when one input is small (± 0.7) and the other input is $= 0$. Here, $m_r = 0.1027$.

(b) When the two inputs are small and opposite, e.g. ($u_1 = 0.7$ and $u_2 = -0.7$) or ($u_1 = -0.7$ and $u_2 = 0.7$). Here, $m_r = 0$.

(c) When the two inputs are medium and opposite, e.g. ($u_1 = 1.4$ and $u_2 = -1.4$) or ($u_1 = -1.4$ and $u_2 = 1.4$). Here, $m_r = 0.1019$.

(d) When one input is medium (± 1.4) and the other input is $= 0$. Here, $m_r = 0.1287$.

2. For translation,

$$P[z] = \frac{0.08874}{z - 0.8173} \tag{3.26}$$

From the kinematic equations of the system, the states, x, y and θ at every time step could also be written in the form of a transfer function in the z-domain, where the state $v_x(t)$, $v_y(t)$ or $\omega(t)$ from the previous time step is the input to the transfer function and the position x, y or θ at the next time step $(k+1)$ is the output of the transfer function.

$$P[z] = \frac{0.04}{z - 1} \tag{3.27}$$

The above transfer functions could now be used to formulate the dynamic equations of each of the states after calculating the inverse z-transform and writing the dynamic equations in discrete form. These equations were then used to formulate the state space of the system as shown above in equations, 3.19, 3.28, 3.29 and 3.30.

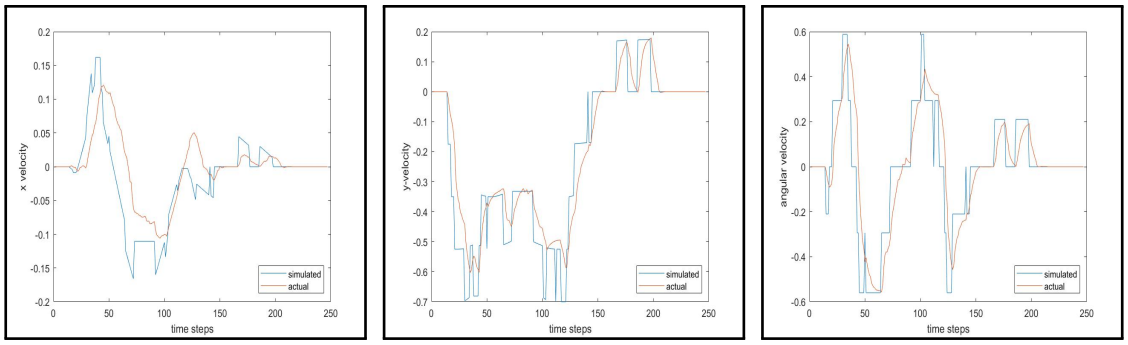
The plots in figure 3.6 show the comparison between collected data and computed data, considering the equations derived using the above transfer functions. The data for Pair-8, trial-2 with translation goal was considered for the analysis.

It could be observed that the root mean squared error was reduced significantly when the first-order transfer function was considered instead of just a proportional mapping or 0^{th} order transfer function. Refer to figure to figure 3.6. The root mean squared errors (RMSE) observed between the actual and computed states all the pairs' trial 2's have been shown in table-3.3. For 4 of the pairs, data with complete information, translation goal has been considered. For other 4 of the pairs, data with complete information, translation+rotation goal has been considered.

Table 3.3: Root mean squared errors between the actual and computed states for 7 of the pairs' trial-2's

Pair/goal type	x	y	v_x	v_y	θ	ω
7/t	0.1529	0.1099	0.0378	0.0358	0.1253	0.0582
8/t	0.0881	0.0935	0.0218	0.0311	0.0666	0.0538
9/t	0.0080	0.1157	0.0070	0.0308	0.0162	0.0349
10/t	0.0316	0.1116	0.0181	0.0277	0.0786	0.0393
11/t+r	0.0624	0.0705	0.0231	0.0334	0.0681	0.0512
12/t+r	0.0705	0.1075	0.0420	0.0316	0.1214	0.1040
13/t+r	0.1633	0.1117	0.0536	0.0590	0.1741	0.0764
14/t+r	0.0420	0.0660	0.0371	0.0431	0.0860	0.0765

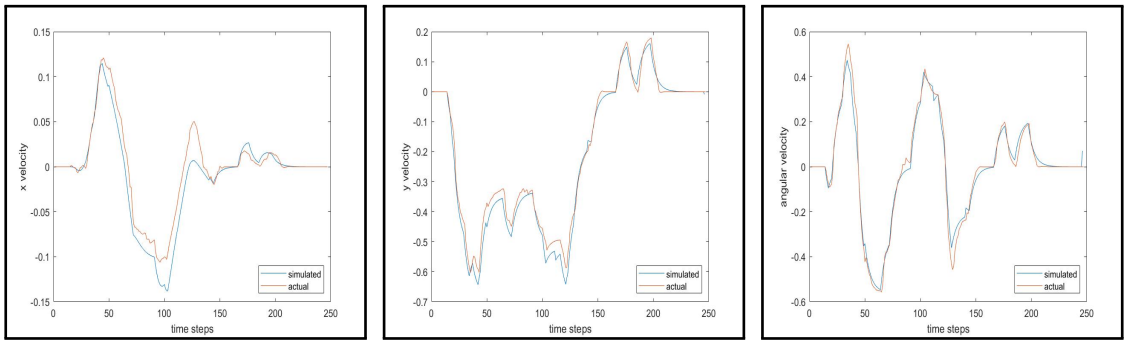
Figure 3.6 shows the velocity plots for trial-2 of the translation goal in case of Pair-8. The sub-figures 3.6(a), 3.6(b), 3.6(c) show the actual and simulated angular velocities when the simulated velocities were computed by mapping the input set points with a proportionality constant. The sub-figures 3.6(d), 3.6(e), 3.6(f) show the actual and compared angular velocities calculated by mapping the input set points using a first order transfer function.



(a) Velocity of x without using the first order mapping

(b) Velocity of y without using the first order mapping

(c) Angular velocity without using the first order mapping



(d) Velocity of x using the first order mapping

(e) Velocity of y using the first order mapping

(f) Angular velocity using the first order mapping

Figure 3.6: Visualization for comparison of the velocities with and without using the first order mapping between the computed set point and the observed states for one of the cases from table-3.3, translation data for pair-8 trial-2

Therefore, A, B_1 and B_2 from the discrete-time state space equations-3.19 were identified as follows.

$$A = \begin{bmatrix} 1 & 0 & t_s & 0 & 0 & 0 \\ 0 & 1 & 0 & t_s & 0 & 0 \\ 0 & 0 & 0.8173 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.8173 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & t_s \\ 0 & 0 & 0 & 0 & 0 & 0.8298 \end{bmatrix} \tag{3.28}$$

$$B_1(k) = \begin{bmatrix} 0 \\ 0 \\ -\frac{0.08874 \times sin(\theta(k))}{2} \\ \frac{0.08874 \times cos(\theta(k))}{2} \\ 0 \\ \frac{m_r(k)}{2} \end{bmatrix} \tag{3.29}$$

$$B_2(k) = \begin{bmatrix} 0 \\ 0 \\ -\frac{0.08874 \times sin(\theta(k))}{2} \\ \frac{0.08874 \times cos(\theta(k))}{2} \\ 0 \\ -\frac{m_r(k)}{2} \end{bmatrix} \tag{3.30}$$

Here, $t_s = 0.04$ seconds is the sampling time. m_r stands for the effect of applied inputs on the rotational dynamics. As mentioned above, the proportionality constant in the transfer function for mapping angular velocity, k had a constant value except for a few cases of input pairs, for which it was set separately. Subsequently, after converting the transfer function to the discrete domain, as mentioned above (equation-3.25), the value of m_r was set separately for a few of these input pairs.

Chapter 4

MODELING HUMAN-HUMAN COLLABORATIVE BEHAVIOUR

To show the improvement in the results over the updates made to the baseline algorithm, the results from Pair-9 have been displayed. The entire analysis has been performed on the Trial-1 with purely translation goal.

The baseline algorithm was developed by referring to [36]. The forward optimal control model in the baseline model is different than the one used in the literature, as the forward problem, in this case, is to model the collaborative behavior. In the forward model, the baseline algorithm uses Nash equilibrium solutions to calculate control inputs separately for all the agents. The genetic algorithm was used as the solver used for the inverse optimal control, with Frobenius norm between the generated and expert (human) trajectories as the fitness function. The first update was made to the fitness function of the genetic algorithm. The new fitness function, DTW was able to generate trajectories that reached the goal position which was not possible by the baseline model. The second update was built upon the first update. The update was made to incorporate the realistic assumption that each agent would predict the optimal inputs for themselves and their partner considering that their partner's cost function would be different than their own (refer to section-4.3). The resulting trajectories from this update were closer to the human trajectories than before. The third update was built upon the previous (second) update, considering stochasticity in input application for each agent (refer to section-4.4). However, due to account for non-unique solutions obtained at each run of the inverse optimal control, an average result from 5 runs of the algorithm was considered for comparison. The fourth update was made by changing the fitness function from the third update back to the original fitness function from the baseline algorithm, Frobenius norm (refer to section 4.5). The results from this update

were compared with the baseline algorithm due to the similarity in their fitness functions. Again, an average performance from 5 runs of the algorithm was considered to ensure that the results were generalized. Results from the fourth update look quite promising and have a remarkable resemblance to the human trajectory.

4.1 Development of Baseline Model

This section aims to formulate a collaborative model that can describe human behavior. Therefore, the model must be able to achieve a response, most similar to human trajectories (from the collected data). A hypothesized structure of the collaborative model, which is an extended optimal control formulation was developed (refer to section-4.1.2). Then,an inverse optimal control algorithm was formulated to find the best weights for the optimal control framework, such that the resulting response is as close as possible to the human trajectory (refer to section-4.1.1).

4.1.1 Inverse Optimal Control to Model the Collaborative Human Behavior

An inverse optimal control framework to model the cost function for every pair's data was formulated [36; 35]. Let $u_i^i(t)$, $u_i^{-i}(t)$ be the input predictions by agent i for themselves and their partner, respectively. Then cost functions for i during the game would be as follows, Let,

$$f_1(i) = \begin{bmatrix} x_g - x(i) \\ y_g - y(i) \\ \theta_g - \theta(i) \\ u_j(i) \end{bmatrix} \tag{4.1}$$

where, $u_j(i) = u_1^1(i)$ or $u_j(i) = u_1^2(i)$ is the feature vector for agent-1.

$$f_2(i) = \begin{bmatrix} x_g - x(i) \\ y_g - y(i) \\ \theta_g - \theta(i) \\ u_j(i) \end{bmatrix} \tag{4.2}$$

where, $u_j(i) = u_2^1(i)$ or $u_j(i) = u_2^2(i)$ is the feature vector for agent-2.

Therefore, the cost functions of the two agents are,

$$J_1 = \sum_{i=1}^{h} (f_1(i))^T W_1^{diag} (f_1(i)) \tag{4.3}$$

$$J_2 = \sum_{i=1}^{h} (f_2(i))^T W_2^{diag} (f_2(i)) \tag{4.4}$$

where $h \in \mathbb{N}^+$ is the prediction horizon, $g = \begin{bmatrix} x_g & y_g & \theta_g \end{bmatrix}'$ is the goal and $\begin{bmatrix} x(i) & y(i) & \theta(i) \end{bmatrix}'$ is the position at iteration i, which is dependant on the hypothesized inputs for the agent and their partner.

The weight vectors for the two agents working together are defined as,
$W_1 = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \end{bmatrix}$, $W_1 \in \mathbb{R}^{4\times 1}$ and $W_2 = \begin{bmatrix} w_{21} & w_{22} & w_{23} & w_{24} \end{bmatrix}$, $W_2 \in \mathbb{R}^{4\times 1}$.

The weighting matrix for the cost function of agent-1 is $W_1^{diag} \in \mathbb{R}^{4\times 4}$. It is a diagonal matrix such that, $W_1^{diag} = diag(\begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \end{bmatrix})$. Similarly, for agent-2, the diagonal weighting matrix $W_2^{diag} \in \mathbb{R}^{4\times 4}$ and $W_2^{diag} = diag(\begin{bmatrix} w_{21} & w_{22} & w_{23} & w_{24} \end{bmatrix})$.

The inverse optimal control algorithm considers $W_1 \neq W_2$.

(Please refer to section-4.1.2 for the forward algorithm).

$p^{(i)}$ such that $p^{(i)}[0:3] = W_1$ and $p^{(i)}[4:7] = W_2$ is defined. The objective function in the formulation is a function of $p^{(i)}$ at iteration, i. (Please refer to the algorithm 1). *Genetic algorithm* is used as a solver in this case for finding the optimal weights [36]. (refer to section-4.1.3). The fitness function is calculated using frobenius norm of the

difference between the human and predicted trajectories [36]. The description of fitness value calculation using frobenius norm has been given in the section-4.1.4

4.1.2 Forward Algorithm to Generate the Trajectories

The task can be described as a standard collaborative couch-pushing task to achieve a defined goal. The goal demands a dominant movement in the y-direction. The application of inputs is also predominantly in the y-direction. Therefore the task involves collaboratively reaching the goal. As the agents start applying input velocities at different time instances, it was observed that there is an initial disturbance generated in x and θ (angular) dimensions. This deviation can be regarded as a noisy initial condition for the agents' model predictive control. However, to ensure that interaction behavior between the agents is not biased in any way and doesn't over-fit, constant (zero) initial conditions were considered for all the pairs. The following sections contain the analysis between simulation results using model predictive control (considering the zero initial conditions) and the actual human-human trajectories.

The feature vector, $f_i \in \mathbb{R}^{1\times n}$ and the weighting matrix, $W_i^{diag} \in \mathbb{R}^{n\times n}$ are defined for an agent, i. W_i^{diag} is a diagonal matrix whose diagonal elements are the weights allocated to the features from the feature vector, f. For example, the diagonal element $W_i^{diag}(m,m)$ is the weight allocated to the m^{th} feature, $f_i(1,m)$.

A quadratic cost function with finite horizon is chosen, that accounts for cost of applying the inputs, their outcomes on the future time-steps and the reward for proceeding towards the goal. Let, $h \in \mathbb{N}^+$ be the prediction horizon.

Assume that, the trajectory for $u_i^i(k)$ is $\xi_i^i(k) = \{u_i^i(k), u_i^i(k+1), ..., u_i^i(k+h-1)\}$ and $u_{-i}^i(k)$ is $\xi_{-i}^i(k) = \{u_{-i}^i(k), u_{-i}^i(k+1), ..., u_{-i}^i(k+h-1)\}$. These are the predicted input trajectories (according to agent, i) for the agent and their partner at the k^{th} step. Now the cost function to be minimized by the agent would be as follows,

$$J_i(\xi_i^i(k), \xi_{-i}^i(k), \bar{y}(k), g) = \sum_{k=1}^{h} (f_i(k))^T W_i^{diag} (f_i(k)) \tag{4.5}$$

Here, $h \in \mathbb{N}^+$ is the prediction horizon. $g = \begin{bmatrix} x_g & y_g & \theta_g \end{bmatrix}'$ is the goal position and $\bar{y}(k) = \begin{bmatrix} x(k) & y(k) & \theta(k) \end{bmatrix}'$ is position at time step k.

As the states considered in the output are only the position terms, the output of the state space can be denoted as,

$$\bar{y}(k) = Cs(k) \tag{4.6}$$

where,

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{4.7}$$

At each time step k, the agent, i chooses a Nash equilibrium solution $< \xi_i^{i*}, \xi_{-i}^{i*} >$ using the condition,

$$\xi_i^{i*} = \underset{\xi_i^i \in \Xi_i}{\arg\min} J_i(\xi_i^i, \xi_{-i}^{i*}, \bar{y}, g) \tag{4.8}$$

$$\xi_{-i}^{i*} = \underset{\xi_{-i}^i \in \Xi_{-i}}{\arg\min} J_i(\xi_i^{i*}, \xi_{-i}^i, \bar{y}, g) \tag{4.9}$$

where Ξ_i and Ξ_{-i} are the action trajectory sets for the agent i and their partner. Then they define a set Ξ^* which contains all the Nash Equilibrium pairs. The agent then picks a solution which minimize the cost, from Ξ^*. This optimal solution is denoted as follows,

$$\xi_i^o = \underset{\xi_i^i, <\xi_i^i, \xi_{-i}^i> \in \Xi^*}{\arg\min} J_i(\xi_i^i, \xi_{-i}^i, \bar{y}, g) \tag{4.10}$$

$$\xi^{o}_{-i} = \underset{\xi^{i}_{-i}, <\xi^{i}_{i}, \xi^{i}_{-i}> \in \Xi^{*}}{\arg\min} \; J_i(\xi^{i}_{i}, \xi^{i}_{-i}, \bar{y}, g) \tag{4.11}$$

By solving (4.10) and (4.11), the agent i obtains the optimal control trajectories. The agent then chooses the first element from each trajectory as the predicted control inputs for current time step (k). Hence $u^{i}_{i} = \xi^{o}_{i}[0]$ and $u^{i}_{-i} = \xi^{o}_{-i}[0]$ which are the first elements from the sets, ξ^{o}_{i} and ξ^{o}_{-i} respectively, are chosen. The agent applies the input, u^{i}_{i}.

Each agent i, (in our case, $i = \{1,2\}$) has their own weight vector W_i and therefore a different cost function, J_i. Hence, each of them applies inputs based on the result of their own cost function. This results in a trajectory that deviates from the most optimal path in case both the agents followed the same cost function. This can be observed due to the variations in strategies (cost functions) of the involved agents.

4.1.3 Optimization Solver: Genetic Algorithm

The optimizer or solver used in the proposed optimization formulation is a genetic algorithm. The ga() function from the Global Optimization toolbox in MATLAB has been used.

Outline of the Genetic Algorithm

Genetic algorithm is an optimization solver that calculates the global optimum in most cases. The difference between a classical solver and the genetic algorithm can be explained as follows. A classical solver generates only a single point at each iteration and updates it to reach the optima. The genetic algorithm generates a group or population of points at each iteration. The population at each iteration is generated by selecting and updating the best points from the previous iterations. This new population is referred to as the new generation. Only the best points in the previous generations contribute to the next generations and the optimal solution evolves over multiple generations.

In the proposed algorithm (1), the vector $p \in \mathbb{R}^{8\times 1}$, is generated by appending the weight vectors of both the agents, W_1 and W_2. The objective function in this case is a combination of the lines 4 and 5 in the algorithm (1). The variables other than p (or $W_1^{(i)}$, $W_2^{(i)}$) are all constants. This is because the physics of the system, initial conditions, prediction horizon, the position of the goal, and expert (human) trajectory is all constant for a trial. Therefore, the lines 4 and 5 together can be visualized as the objective function, which is a function of the variable, p^i and returns the fitness value e_i (refer to the section 4.1.4). The genetic algorithm finds the best value of p such that a minimum e is reached.

The steps followed by the solver (ga() function in MATLAB) are as follows;

1. An initial population is chosen randomly. In our case, the initial population consists of multiple options of $p \in \mathbb{R}^{1\times 8}$. This is because, p is the decision variable and is adjusted to minimize the value of fitness function, e. These options of p are referred to as members of the generation. Each generation consisted of 200 members for the current application.

2. The raw fitness scores for each option of p (member) in this population are calculated using the fitness function.

3. These raw scores are scaled to a new range of fitness values called expectation values.

4. The members with lower fitness values are chosen as parents for the next generations.

5. Three types of children, elite, crossover, and mutation are generated from the selected parents from the previous generation and form the next generation.

6. The members with the best (lowest) fitness values are passed directly to the new generation and are called **elite children**.

7. A combination of multiple members from the previous generation which are selected as parents produce the **crossover children**.

8. Updating the values of a single member among the chosen parents from the previous generation produces **mutation children**.

9. Each generation consists of a population of 8×1 vectors and a new generation is created at each iteration of the algorithm.

10. The algorithm keeps running until the relative change in the fitness value of the best member p^i in a generation i is lesser than the threshold of convergence for more than a certain number of generations.

The objective function for the ga() solver is a function of p and returns the fitness value which is calculated using the fitness function. (Refer to section 4.1.4). Therefore, every point in the generations of the genetic algorithm is of the dimension 8×1.

Threshold of convergence was chosen as $\varepsilon = 0.01$ and the maximum stalling generations, $max\_stall\_gen = 20$ generations. This means when the change in the average fitness value $< \varepsilon$ for more than max_stall_gen generations, the solver stops the optimization process. Every generation consists of a population of 200 hypothesized $p's$.

4.1.4 Fitness Function

Frobenius norm between simulated and expert trajectories is chosen to calculate the fitness function in the inverse optimal control formulation by [36]. The squared of the Frobenius norm value is used as the fitness function. Frobenius norm is the matrix norm. It compares every term from the first matrix to its corresponding term in the second matrix.

Let the output from the human trajectory (consisting of the states x, y, θ from the first to the last time step) be $y(i, j)$ and the output from the simulated trajectory to be $\bar{y}(i, j)$. Let the dimensions of these two matrices be $m \times n$. Frobenius norm of a matrix the difference between simulated and human trajectory can be defined as,

$$frob\_norm(y, \bar{y}) = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} \|y(i, j) - \bar{y}(i, j)\|^2} \tag{4.12}$$

As fitness function, e_{frob_norm} is defined as the square of the frobenius norm of difference between the actual and simulated trajectories,

$$e_{frob\_norm}(y, \bar{y}) = \sum_{i=1}^{m} \sum_{j=1}^{n} \|y(i, j) - \bar{y}(i, j)\|^2 \tag{4.13}$$

This method tries to generate a trajectory, which has a time-step wise closest resemblance to the human trajectory.

4.1.5 Re-Sizing the Data Before Analysis

It was observed from the collected data that a dyad would not start at the first time step or end at the last time step. There was a region between their start time and their end time of the task. The rest of the section of the trajectory was observed to have no actions by the agents. For analyzing the interaction behavior, it is important to consider the effective trajectory only. It was also observed that each pair had a different start time and an end time for the task, resulting in different lengths of effective trajectories for each of them. Therefore, to have the trajectories of equal lengths for the sake of comparison, the trajectories were resampled to a common length. This was done using the MATLAB function, resample(). The inputs required for this function are, the time-series to be re-sampled, desired size of the series after re-sampling and the current size of the time-series. The sampling time (t_s), used for simulating the low level dynamics for the trial, according to equations 3.19, 3.28, 3.29, 3.30 is also modified. The new sampling time is defined as

$t_s^{new} = \frac{0.04 \times t_{new}}{t_{old}}$, where t_{new} is the new length of the time-series and t_{old} is the length before re-sampling. The re-sampled trajectories would now be used for fitting the human models using the inverse optimal control algorithm - 1.

4.1.6 The Proposed Optimization Formulation

The inverse optimal control framework is formulated as an optimization with the fitness function (refer to section-4.1.4) as the objective function. The objective function is a function of the decision variable, p which is an 8×1 vector (explained in the section 4.1.3). The genetic algorithm, a global optimization solver is used for the process.

The genetic algorithm hypothesizes a population of 200 decision variables at every generation. For all generations, the algorithm- 2 is used to calculate the trajectory corresponding to every hypothesized vector p^i in the generation, which is used to calculate it's corresponding fitness value, e_i. The fitness value is calculated as the squared of the frobenius norm (refer to section-4.1.4), which has been denoted as $\|\cdot\|_F^2$ in the algorithm-1. These fitness values are used for selecting parent members for the next generation. As explained in section- 4.1.3.

The genetic algorithm keeps running until the change in average fitness value for 20 consecutive generations is less than 0.01. The best fitness value at the final generation and its corresponding decision vector is returned as the best fit. In the algorithm-1, P^i is denoted as the i^{th} generation of p's.

Algorithm 1: Algorithm to fit W_1, W_2 given the system dynamics, threshold of convergence and expert demonstrations

input : Expert demonstrations y[1:N], state space $\{A, B_1, B_2, C\}$, initial condition $s(0)$, goal $\mathbf{g} = [x_g, y_g, \theta_g]$, prediction horizon h, input candidate set I, threshold of convergence ε, number of time steps N

output: Optimal decision vector, p

Initialize $P^{(0)}$ = random() and $i = 0$

while $\Delta e_i > \varepsilon$ **do**

  For all $p \in P^i$, generate

  $\bar{y}(t = 0 : N; p) = generate\_trajectory(A, B_1, B_2, C, p, s(0), h, goal)$

  For all $p \in P^i$, calculate $e_i(p) = \|\bar{y}(t = 0 : 245; p) - y(t = 0 : 245)\|_F^2$

  Find e_i^{avg}, e_i^{best} from $e_i(p)$, $p \in P^i$

  $\Delta e_i^{avg} = e_i^{avg} - e_{i-1}^{avg}$

  Update $p_{best}^{(i)}$ based on $e_{(i)}^{best}$

  Increment i

  Generate new generation P^i using the genetic algorithm solver

end

$p = p_{best}^{(i)}$

return p

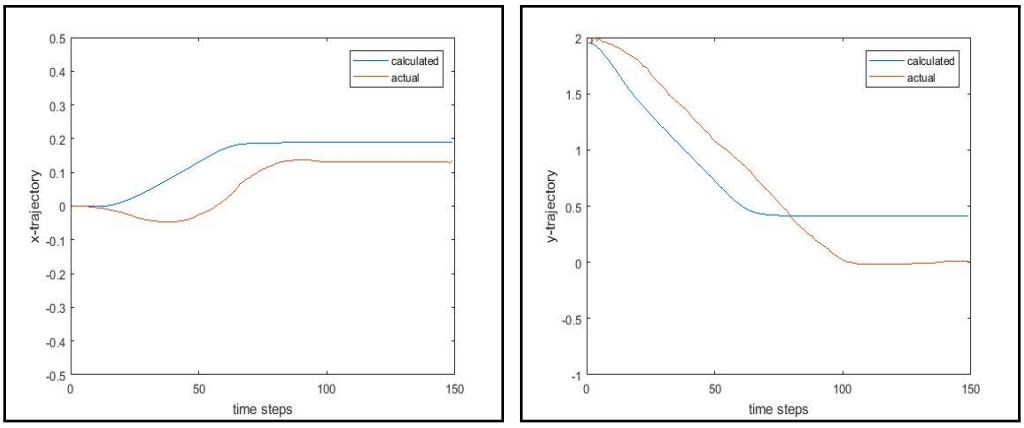
Algorithm 2: generate_trajectory(): The forward algorithm in the inner loop of the algorithm 1

input : State space, $\{A, B_1, B_2, C\}$, updated vector, p, initial condition, $s(0)$, the goal, $\mathbf{g} = [x_g, y_g, \theta_g]$, prediction horizon, h, input candidate set, I

output: Generated optimal trajectory using determined weights, $\bar{y}(t = 1 : N; p)$

for $t = 1 : N$ **do**
- Extract W_1 and W_2 from p
- Update B_1 and B_2 based on (3.29) and (3.30)
- **for** $(\xi_1^1, \xi_2^1) \in \Xi_1 \times \Xi_2$ **do**
  - compute $J_1(\xi_1^1, \xi_2^1, p, g)$ using W_1
- **end**
- Find the Nash Equilibira set Ξ^* based on (4.8) and (4.9)
- Compute ξ_1^o and ξ_2^o with (4.10) and (4.11)
- Choose the first action in ξ_1^o as the optimal control, $u_1^o = \xi_1^o[0]$.
- **for** $(\xi_1^2, \xi_2^2) \in \Xi_1 \times \Xi_2$ **do**
  - compute $J_2(\xi_1^2, \xi_2^2, p, g)$ using W_2
- **end**
- Find the Nash Equilibira set Ξ^* based on (4.8) and (4.9)
- Compute ξ_1^o and ξ_2^o with (4.10) and (4.11)
- Choose the first action in ξ_2^o as the optimal control, $u_2^o = \xi_2^o[0]$.
- Update the value of $\bar{y}(t)$ using u_1^o and u_2^o based on (4.6)

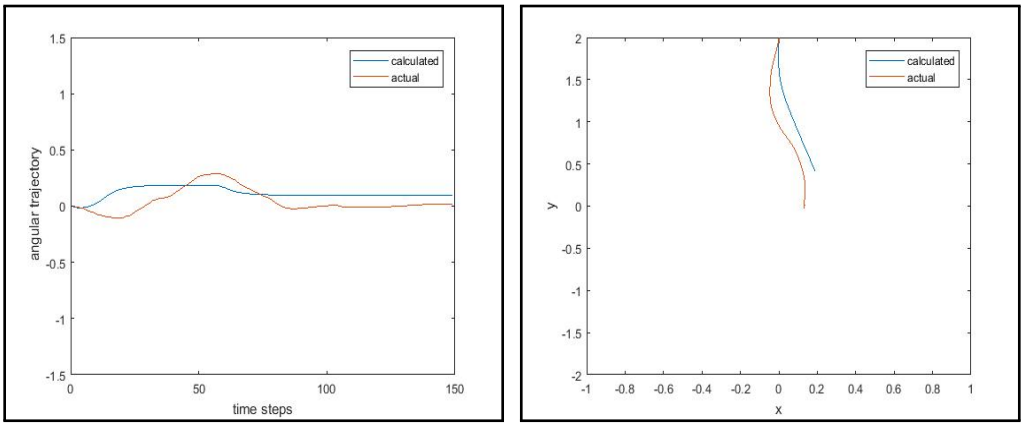
end



(a) Comparison of simulated and actual x-trajectory, tuned using the optimization algorithm

(b) Comparison of simulated and actual y-trajectory, tuned using the optimization algorithm

Figure 4.1: Trajectories for the actual and simulated trajectories for pair-9, trial-1 tuned using the algorithm



(a) Comparison of simulated and actual angular trajectory, tuned using the optimization algorithm

(b) Comparison of simulated and actual x vs y trajectory, tuned using the optimization algorithm

Figure 4.2: Trajectories for the actual and simulated trajectories for pair-9, trial-1 tuned using the algorithm

4.1.7 Preliminary Analysis

Observe Figures 4.1 and 4.2. It can be observed that humans would not try to reach the goal as soon as possible, as long as they can reach in the defined time-limit. The algorithm would find a trajectory that has a minimum Euclidean distance from the human trajectory for x, y, and θ. Suitable weights determining cost functions of this trajectory are simulated. Higher costs are allotted to the control inputs to generate a slower trajectory (that resembles the human trajectory, time-step wise). At a certain point in the simulation, the input becomes costlier than the distance from the goal, thus not reaching the goal ($y = 0$). If lower costs were selected for the control inputs, the trajectory would become faster than the human trajectory. While the goal would be reached in this case, the euclidean distance between the simulated and human trajectories would be higher than before, resulting in the rejection of this solution.

Therefore, a method that could capture the trend of human trajectories, rather than time-step wise comparison was required.

4.2 Update 1: Dtw Distance As New Fitness Function

Human behavior is too complex to model perfectly. Sometimes, people tend to pause during the task. Sometimes, they act very slowly and are not concerned about the elapsed time if they ultimately reach the goal, although there is a time constraint.

Due to these factors, it is difficult to replicate the exact human trajectory. However, to analyze the humans' strategies, trajectories that follow similar trends to human trajectories can be considered. These trajectories do follow similar trends to the human trajectories but are not synchronized with them. Most of the time, human trajectories are slightly delayed. Dynamic time warping was used as the fitness function, to find a trajectory that follows the most similar trend to the human trajectory.

4.2.1 Fitness Function: Dynamic Time Warping

Now, the fitness function in the algorithm-1 is calculated using $DTW()$ (dynamic time warping distance) between simulated and expert (human) trajectories.

Dynamic time warping is an algorithm that is used to measure the similarity among two chronological time series sequences, which may vary in speeds. It has be used to align sequences which may be misaligned, but follow a similar trend and can also be used to calculate similarity between them [50]. A few applications of dynamic time warping are, to calculate the closest predictions for sales using trends followed in the past observations [51], analysis of walking sequences using data from different individuals [52] and speech analysis [53]. The similarity between two chronological time-series sequences and the DTW() distance between them are inversely proportional. Therefore, lower DTW distance value implies more similarity between the sequences and vice versa.

4.2.2 Mathematical Description of the Dynamic Time Warping algorithm

Let the two sequences in comparison be $Y = \{y(1), y(2), ..., y(n)\}$ and $\bar{Y} = \{\bar{y}(1), \bar{y}(2), ..., \bar{y}(n)\}$

The sequences are first aligned. In order to align the sequences, they are arranged in an $n \times n$ grid. One of the axes in the grid corresponds to the indices of the sequence, Y and the other axis corresponds to the indices of the sequence, $\bar{Y}$. Therefore, each term in the sequence Y is aligned with each term in the sequence $\bar{Y}$.

A set of warping paths ($W = \{w_1, w_2, ..., w_m\}$), containing different alignments of the sequences is generated. The elements in the paths are tuples, $< i, j >$. Where i is the index of an element from Y and j is the index of an element from $\bar{Y}$.

The value of a member $w(k) =< i, j >$ such that $w \in W$ is

$$dist(w(k)) = \|Y(i) - \bar{Y}(j)\| \tag{4.14}$$

The function $DTW(Y,\bar{Y})$ returns the most optimal path W such that

$$W(Y,\bar{Y}) = \arg\min_{W} \sum_{k=1}^{n} dist(w(k)) \tag{4.15}$$

The final numeric value returned by the function $DTW(Y,\bar{Y})$ is

$$DTW(Y,\bar{Y}) = \sum_{k=1}^{n} dist(W(k)) \tag{4.16}$$

where W is the optimal warping path from (4.15).

A few rules for generating the warping paths are as follows.

- Every point in the first sequence must be paired with one or more points in the second sequence and vice versa.
- The first points in the two sequences must be matched with each other, but they may not be the only matches for these two points.
- The last points in the two sequences must be matched with each other, but they may not be the only matches for these two points.
- The paired indices of the points in the first sequence and the points in the second sequence must be monotonically increasing.

The output vectors y and $\bar{y}$ in our case are not 1 dimensional vectors. They are 3 dimensional vectors consisting of the trajectories of x, y and θ. The $DTW()$ function treats each of these columns (trajectories of x, y and θ) from the output matrix as independent vectors and returns the sum of the three final values.

The line 5 in the algorithm- 1 is changed to dynamic time warping distance instead of the frobenius norm.

4.2.3 Comparison of the Fitness Functions: Dtw distance and Frobenius Norm

Let the generated trajectory from the algorithm be $\bar{Y} \in \mathbb{R}^{m \times n}$ and the human trajectory according to the collected data be, $Y \in \mathbb{R}^{m \times n}$. The value of $m = 3$ corresponding to the three dimensions of output trajectory, x, y and θ and n = length of the trajectory in time-steps. Therefore,

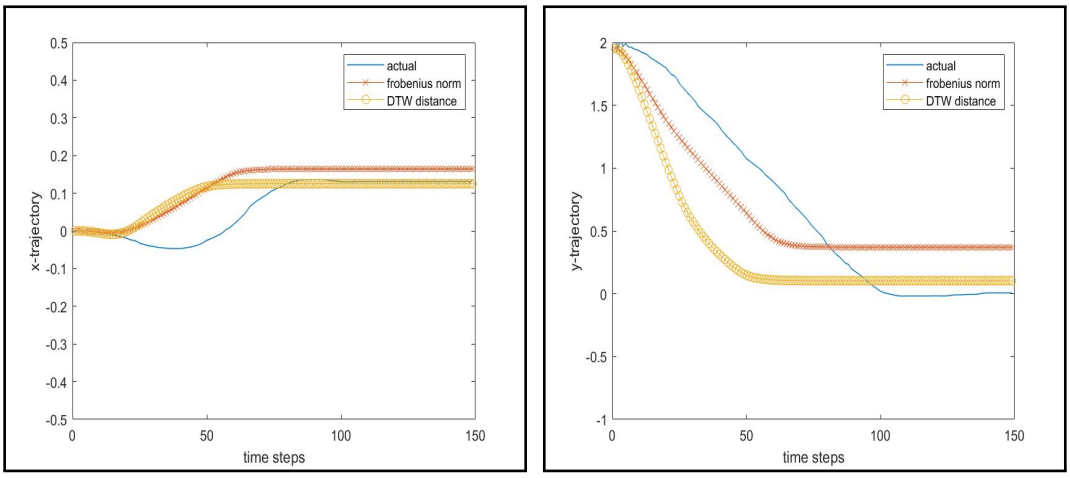
$$e_{frob\_norm}(Y, \bar{Y}) = \sum_{i=1}^{m} \sum_{j=1}^{n} \|Y(i,j) - \bar{Y}(i,j)\|^2 \tag{4.17}$$

$$e_{DTW}(Y, \bar{Y}) = \left[\sum_{i=1}^{m} \sum_{j=1}^{n} dist(W) \right]^2 \tag{4.18}$$

where W is the optimal warping path described in section-4.2.2. The optimal warping path is calculated along the time-steps, $(j = 1 : n)$ by comparing every point in $Y(i, j = 1 : n)$ with every point in $\bar{Y}(i, j = 1 : n)$. The comparison is carried out using the euclidean distance measure, or norm (refer to section-4.2.2).

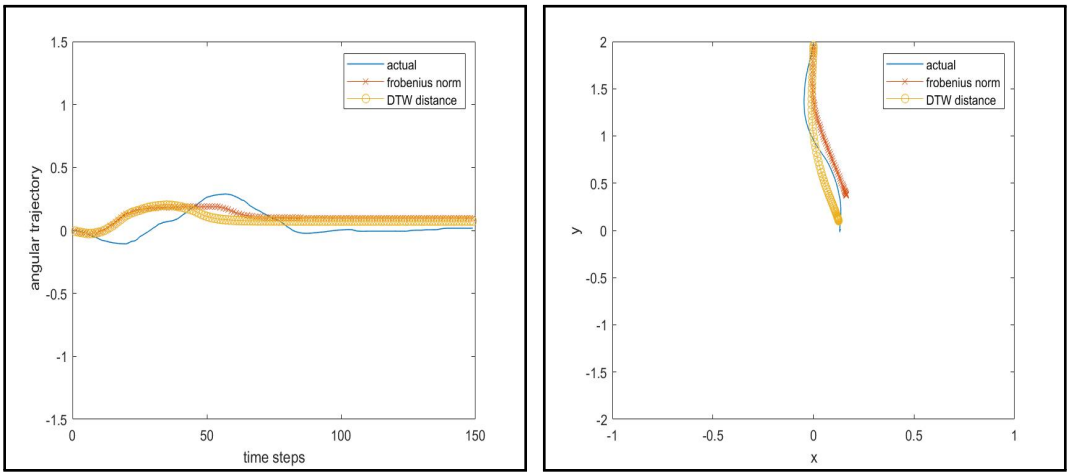
The major difference between the nature of the two fitness functions is the calculation of the warping path in case of DTW distance. This step accounts for the misalignment in the two sequences/trajectories and therefore estimates the similarity in the trend of the trajectories rather than the time-step wise comparison carried out using the Frobenius norm.

Another difference between the calculations of the fitness functions is as follows. The fitness function using Frobenius norm is calculated as the sum of squares of the terms in the difference matrix, while DTW distance is calculated using the square of the sum of the terms in the warping path. This is the reason, why the fitness values obtained using DTW distance were larger than the ones obtained using Frobenius norm.



(a) Comparison of x vs $timestep$ trajectories (b) Comparison of y vs $timestep$ trajectories

Figure 4.3: Comparison of trajectories generated from baseline model (frobenius norm as fitness function) and update-1 model (DTW distance as fitness function) with actual human trajectory



(a) Comparison of θ vs $timestep$ trajectories (b) Comparison of x vs y trajectories

Figure 4.4: Comparison of trajectories generated from baseline model (frobenius norm as fitness function) and update-1 model (DTW distance as fitness function) with actual human trajectory

4.2.4 Analysis

An improvement in the completeness of the task (reaching the goal) could be observed in the trajectory obtained from this update.(Refer to figures 4.3, 4.4). However, using root mean squared error as an analysis metric was no longer possible. This is because, while the shape of the trajectory (trend) was similar to the human trajectory, the exact trajectory (of each co-ordinate at every time step) was not achieved using this fitness function. Therefore, the metric used to compare the trajectories had to be changed to be DTW distance (dynamic time warping distance). The description of evaluating the value of this metric has been explained in 4.2.1.

4.3 Update 2: Assume Different Cost Function for Partner

To improve the performance of the forward algorithm further, a level of complexity was added to the optimal control (algorithm-2).

Consider an agent, i. Until now, i is assumed to evaluate the input trajectories of itself and the partner according to equations 4.8, 4.9, 4.10 and 4.11. The agent uses the same cost function, J_i which corresponds to the weighting vector W_i for the evaluation. However, a more realistic scenario would be, that humans would not assume that their partner is identical (has the same strategy) to them during collaborative tasks.

Therefore, i is now modeled to consider different weighting vectors, $W_{i,i}$ and $W_{i,-i}$ and subsequently the cost functions $J_{i,i}$ and $J_{i,-i}$ to evaluate the control trajectories of itself and its partner. Consequently, at each time step k,

$$J_{i,i}(\xi_i^i(k), \xi_{-i}^i(k), \bar{y}(k), g) = \sum_{k=1}^{h} (f_i(k))^T W_{i,i}^{diag} (f_i(k)) \tag{4.19}$$

$$J_{i,-i}(\xi_i^i(k), \xi_{-i}^i(k), \bar{y}(k), g) = \sum_{k=1}^{h} (f_i(k))^T W_{i,-i}^{diag} (f_i(k)) \tag{4.20}$$

Here $W^{diag} = diag([W])$ is a diagonal matrix which has diagonal elements equal to the elements in the vector W, and $f_i(k)$ is the feature vector at time-step k (the definitions are the same as described in section-4.1.2). Therefore, i chooses the Nash equilibrium solution, $< \xi_i^{i*}, \xi_{-i}^{i*} >$ using the condition,

$$\xi_i^{i*} = \underset{\xi_i^i \in \Xi_i}{\arg\min} J_{i,i}(\xi_i^i, \xi_{-i}^{i*}, \bar{y}, g) \tag{4.21}$$

$$\xi_{-i}^{i*} = \underset{\xi_{-i}^i \in \Xi_{-i}}{\arg\min} J_{i,-i}(\xi_i^{i*}, \xi_{-i}^i, \bar{y}, g) \tag{4.22}$$

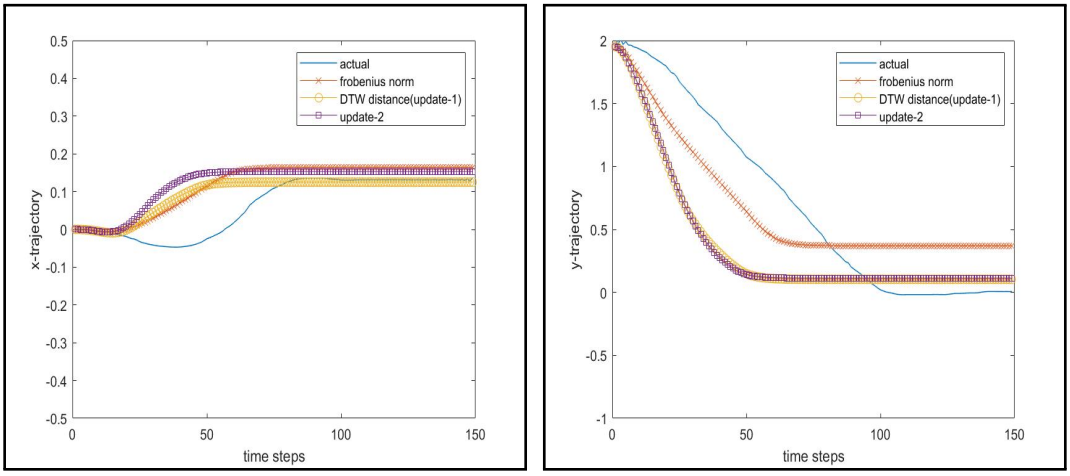
where Ξ_i and Ξ_{-i} are the action trajectory sets for the agent i and their partner.Further, i defines the set Ξ^* containing all the Nash Equilibrium pairs and then picks a solution from it which minimize the cost. This optimal solution is denoted as follows,

$$\xi_i^o = \underset{\xi_i^i, <\xi_i^i, \xi_{-i}^i> \in \Xi^*}{\arg\min} J_{i,i}(\xi_i^i, \xi_{-i}^i, p, g) \tag{4.23}$$

$$\xi_{-i}^o = \underset{\xi_{-i}^i, <\xi_i^i, \xi_{-i}^i> \in \Xi^*}{\arg\min} J_{i,-i}(\xi_i^i, \xi_{-i}^i, p, g) \tag{4.24}$$

which is used to further calculate, $u_i^i = \xi_i^o[0]$ and $u_{-i}^i = \xi_{-i}^o[0]$ and therefore the agent's input, u_i^i.

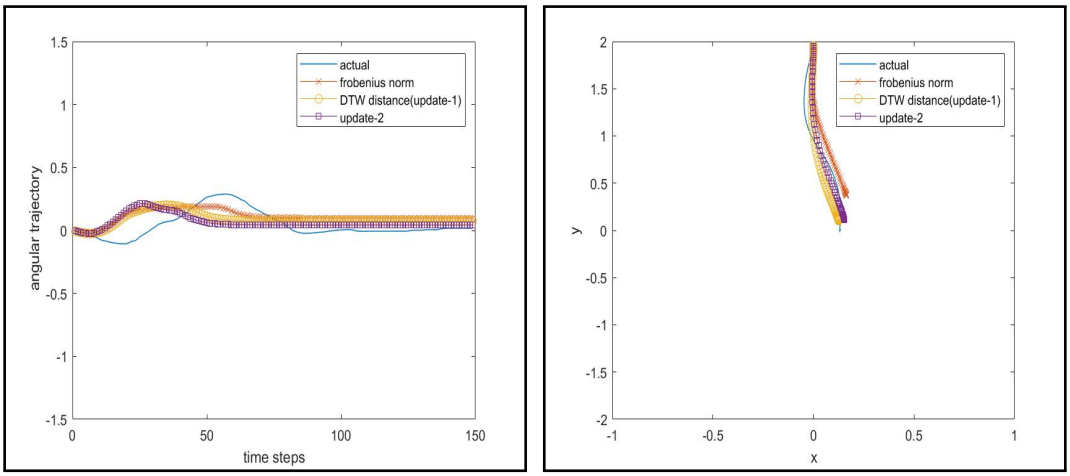
Applying this update to both the agents, the weight vectors to be fitted now are, W_{11}, W_{12} for agent-1 and W_{21}, W_{22} for agent-2. When these weight vectors are concatenated together to form a single decision variable for the genetic algorithm, the decision variable would now be of the size, 16×1 instead of 8×1 as it was before (refer to section-4.1.1).



(a) Comparison of trajectories generated from baseline and updated models with actual human trajectory (x vs $timestep$)

(b) Comparison of trajectories generated from baseline and updated models with actual human trajectory (y vs $timestep$)

Figure 4.5: Comparison of trajectories generated from baseline and updated models with actual human trajectory



(a) Comparison of trajectories generated from baseline and updated models with actual human trajectory (θ vs $timestep$)

(b) Comparison of trajectories generated from baseline and updated models with actual human trajectory (x vs y)

Figure 4.6: Comparison of trajectories generated from baseline and updated models with actual human trajectory

4.3.1 Analysis

An improvement can be observed in the results.(Refer to figures 4.5, 4.6). A reduction in the error metric can also be observed. The conclusion from these improvements is, even in a mundane task such as this, humans tend to assume the partner's behavior/strategy (different from their own) while choosing their actions. The improvement in the fitness value corresponding to optimal weights returned by the genetic algorithm solver has been displayed in table-4.1. As mentioned in the previous update, the comparison metric is DTW() distance.

Table 4.1: Comparison of fitness values from updates 1 and 2 with the baseline algorithm.

Algorithm	**Fitness value**
baseline	31.1563
Update-1	20.0257
Update-2	16.4604

4.4 Update 3: Stochastic Inputs Using Bounded Rationality

Updates made to the collaborative model were able to obtain more similarity between the trends of generated trajectory and the human trajectory. However, the model lacks an important aspect of human behavior, stochastic nature. When a human predicts the optimal input at a particular time step, there is just a higher probability that the agent would take the action. However, there is also a probability that they would take an action other than the predicted one. This behavior would be dependant on how rational or random the human is during the task.

4.4.1 Incorporating Stochastic Inputs

While modeling the optimal control problem with stochastic inputs, the stochastic nature can be incorporated by generating a probability distribution (P) over the input candidate set. P is determined by choosing a coefficient of rationality, which would be called α. The procedure is adapted from [8].

Consider the action candidate set to be U. With the algorithm from the update- 2, (section - 4.3), an agent i computes the optimal control input pair (u_i, u_{-i}). Assuming that their partner is completely optimal, the agent builds a probability density function over the action candidate set, U. The action is selected from U based on this probability density function (PDF) over U.

How Is the PDF Generated?

Let $u_i \in U$ where, U is the action candidate set. Referring to algorithm- 2, for each u_i, a cost is calculated considering that the predicted input of the partner, u_{-i} is optimal and deterministic. Let the cost candidate set, C be the set of costs corresponding to each $u_i \in U$ calculated using equation-4.3/4.4. Based on the cost candidate set, the probability density function should be determined such that, the higher the cost of input; the lower is the probability of applying that input.

Therefore, the probability density for each member $u_i \in U$, was calculated based on its corresponding value of cost as follows,

$$P(u_i|u_{-i},\alpha,p,g) = \frac{e^{\alpha(c_{max}-c_i)}}{\sum_{c\in C} e^{\alpha(c_{max}-c)}} \tag{4.25}$$

Here, $c_{max} \in C$ is the maximum cost from the cost candidate set C, $c_i \in C$ is the cost for applying input u_i, α is the coefficient of rationality. The higher the value of α, the more is the rationality of the agent i's action. p are the hypothesized weights and g is the goal position. The cost function is dependant on these two quantities.

The results obtained from this update were found to reduce the final error (DTW distance between the generated trajectory and human trajectory) to a great extent.

An Explanation for Equation-(4.25)

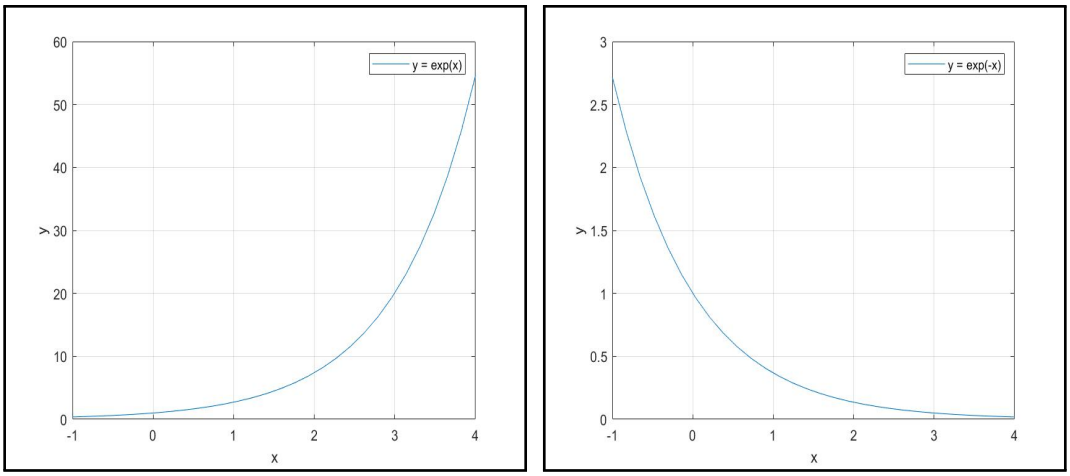
The literature [8] referred to generate the probability density function, uses the formulation, 'maximization of a reward function' to generate the control inputs. Therefore, each candidate, u from the action candidate set U has an associated reward. Consider that R is the reward candidate set which corresponds to the candidates $u_i \in U$. Therefore, the higher the value of a candidate in R, the higher should be its corresponding value in the probability density function (Figure 4.7(a)). To capture this relationship, the authors of [8] use the following equation.

$$P(u_i|u_{-i}, \theta, \alpha) = \frac{e^{\alpha R_{u_i}}}{\sum_{u \in U} e^{\alpha R_u}} \tag{4.26}$$

where, $R_u \in R$ is the reward corresponding to the input u and is a function of the trained weights, θ

However, in our case, the relationship between the cost candidate set C and the probability density function is the opposite. Therefore, the lower the value of a candidate in C, the higher should it's the corresponding value of the probability density function. To achieve this, a reducing exponential function (Figure 4.7(b)) is required in contrast to the increasing exponential function, used in the literature.

But, a function $y = e^{-x}$ decays to 0 too quickly. As the cost function in our case is quadratic, the generated cost values are positive and significantly greater than 0. Therefore,



(a) Plot for the exponential function, $y = e^{x}$ (b) Plot for the exponential function, $y = e^{-x}$

Figure 4.7: Exponential functions for pdf generation

the corresponding value of y using the equation, $y = e^{-c_i}$, $c_i \in C$ would tend to zero in almost all the cases, considering our cost function.

As a solution to this issue, the exponential curve is moved so that, value at the maximum cost candidate is 1 and the rest of the values of the costs are all > 1. The numerator in the equation (4.25) would then be exponentially larger for smaller cost values. (Observe the comparison between $y = e^{-C_{u_i}}$ and $y = e^{max(C)-C_{u_i}}$, in figure-4.8. This is just a visualization to describe the process of defining the probability density function. The plot in the illustration assumes $c_{max} = 3$, but it can be any arbitrary value.)

Once the agent has calculated the probability density function ($pdf(u_i)$) over the action candidate set U, a control action $u_i \in U$ is selected based on it. The control action may or may not be the most optimal but is not completely random either, because it is picked based on a probability density function. This process of action selection is performed at every time-step and applied between lines 13 and 14 of the algorithm-2. The rationality coefficient α is assumed to be different for both the agents, resulting in an addition of 2 terms (α_1 and α_2) in the decision variable, p. Therefore, the new decision variable now has dimensions, 18×1.

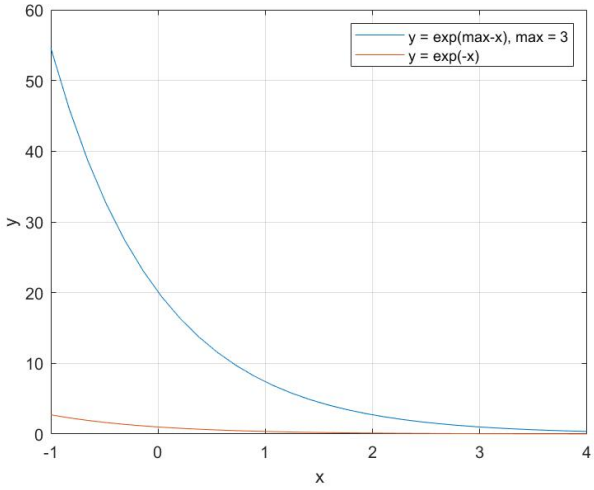
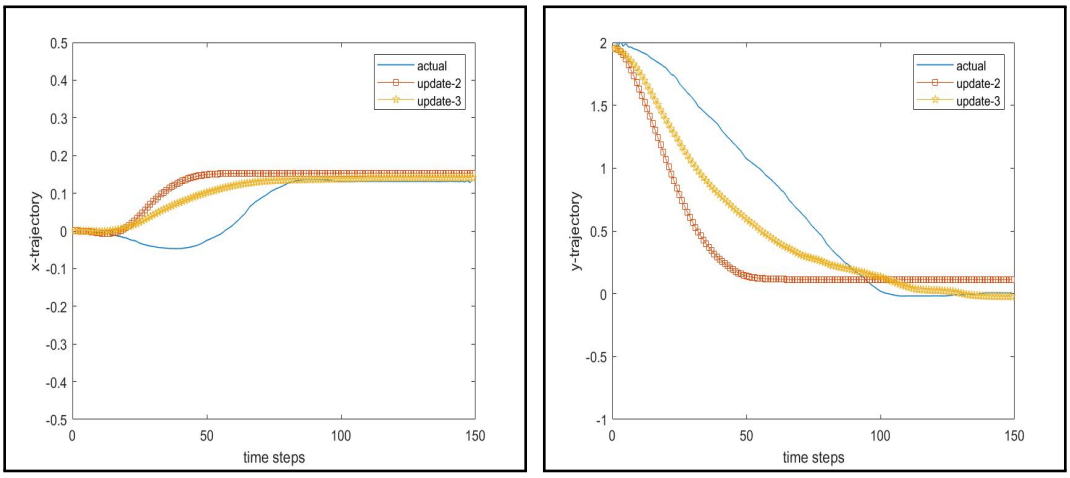


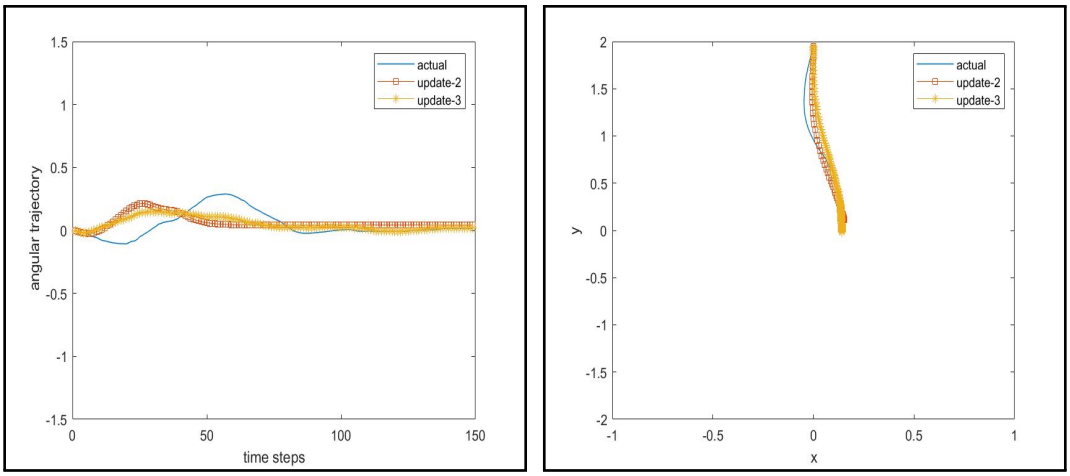
Figure 4.8: Plot for the exponential function, $y = e^{-x}$ and $y = e^{max-x}$



(a) Comparison of trajectories generated from baseline and updated models with actual human trajectory (x vs $timestep$)

(b) Comparison of trajectories generated from baseline and updated models with actual human trajectory (y vs $timestep$)

Figure 4.9: Comparison of trajectories generated from baseline and updated models with actual human trajectory



(a) Comparison of trajectories generated from baseline and updated models with actual human trajectory (θ vs $timestep$)

(b) Comparison of trajectories generated from baseline and updated models with actual human trajectory (x vs y)

Figure 4.10: Comparison of trajectories generated from baseline and updated models with actual human trajectory

4.4.2 Analysis

Inverse optimal control (Algorithm-1) is performed with this update resulting in great improvements to the $DTW()$ distance or error value. (Refer to figures 4.9, 4.10)

However, due to the presence of randomness in the forward algorithm (trajectory generation), the search-space of optimization changes completely. Multiple solutions have become possible. This is because rationality coefficients determine a probability density function. So, for a pair of them, (α_1, α_2), there would be multiple resultant trajectories. When the genetic algorithm simulates a trajectory with the pair the result is a member of a large set of results. Therefore, every time the inverse optimal control is performed, different solutions (p's) would be achieved. Multiple runs were performed to check similarities between the fitted cost function and the corresponding fitness values. To ensure no biases, the recorded fitness values and trajectories are used directly as returned by the genetic algorithm solver. This exercise is performed five times to ensure more general conclusions.

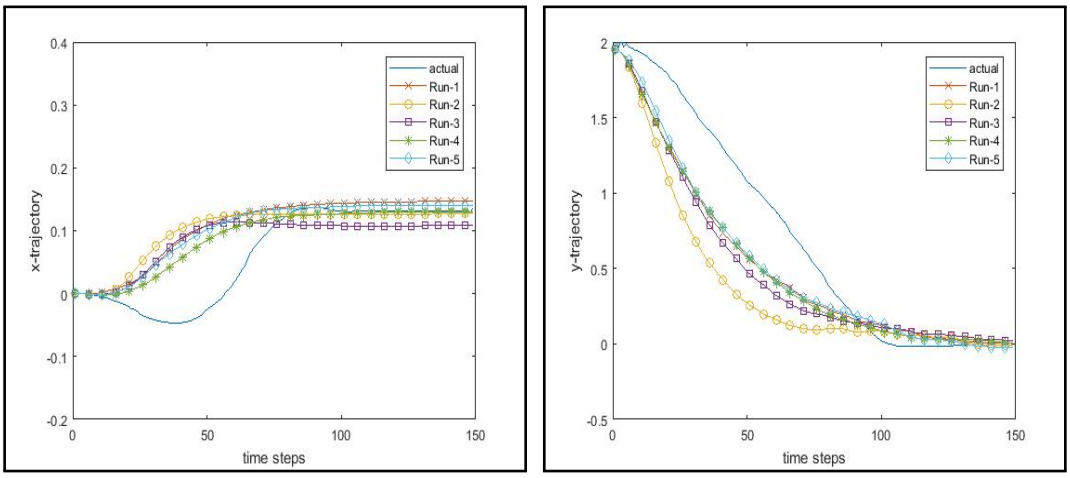
Table-4.2 shows the improvement in fitness value of all the updates including the current one. A mean of all the five fitness values (refer to table-4.3) is considered to ensure that the reduction in fitness value is not just a special case of result.

Table 4.2: Comparison of fitness values from updates 1 and 2 with the baseline algorithm.

Algorithm	**Fitness value**
baseline	31.1563
Update-1	20.0257
Update-2	16.4604
Update-3	9.915

Table 4.3: Comparison of fitness values from 5 runs of the optimization algorithm.

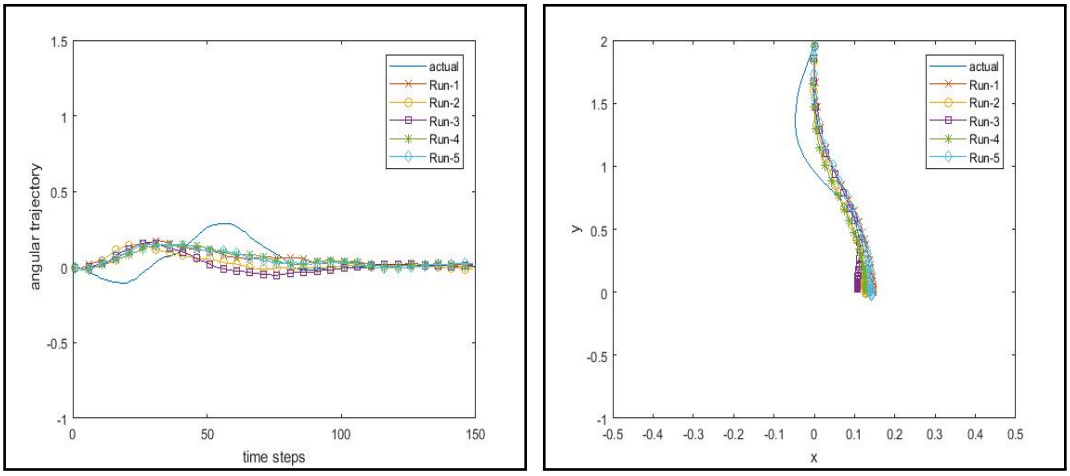
Run no.	1	2	3	4	5
fitness value	10.6035	8.7286	10.7608	9.7437	9.7384



(a) Comparison of trajectories generated from 5 runs of the stochastic input and DTW fitness function models, with actual human trajectory (x vs $timestep$)

(b) Comparison of trajectories generated from 5 runs of the stochastic input and DTW fitness function models, with actual human trajectory (y vs $timestep$)

Figure 4.11: Comparison of trajectories generated from 5 runs of the stochastic input and DTW fitness function models, with actual human trajectory



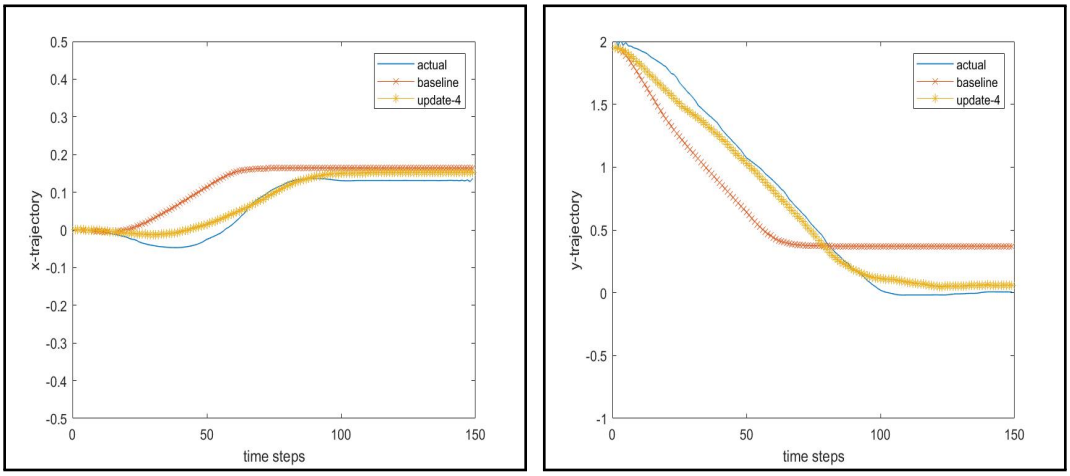
(a) Comparison of trajectories generated from 5 runs of the stochastic input and DTW fitness function models, with actual human trajectory (θ vs $timestep$)

(b) Comparison of Trajectories Generated From 5 Runs of the Stochastic Input and DTW Fitness Function Models, With Actual Human Trajectory (x vs y)

Figure 4.12: Comparison of trajectories generated from 5 runs of the stochastic input and DTW fitness function models, with actual human trajectory

4.5 Update 4: Original Fitness Function With Stochastic Inputs

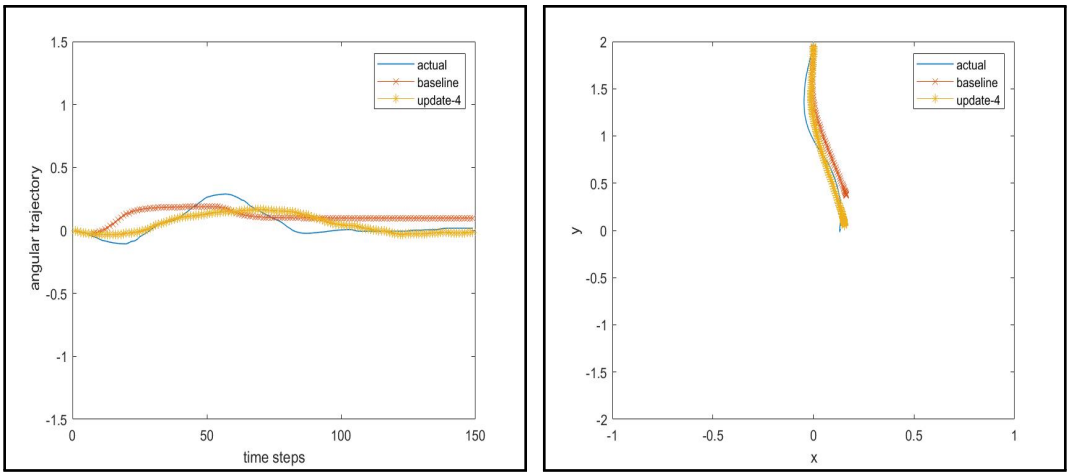
Adding a stochastic component to all the agents' input selection changes the possible behavior dynamic of the model. In the second update, the fitness function was changed to DTW to tackle the effect of stochastic/random human behavior, which is now accounted for by adding stochasticity to the agents' inputs. Therefore, the model is tested with the original fitness function, Frobenius norm between human state trajectory and simulated state trajectory. The rest of the problem setup was kept the same. i.e. dimensions of the decision variable, p remained 18×1. (Refer to figures 4.13, 4.14)



(a) Comparison of trajectories generated from baseline and updated models with actual human trajectory (x vs $timestep$)

(b) Comparison of trajectories generated from baseline and updated models with actual human trajectory (y vs $timestep$)

Figure 4.13: Comparison of trajectories generated from baseline and updated models with actual human trajectory



(a) Comparison of trajectories generated from baseline and updated models with actual human trajectory (θ vs $timestep$)

(b) Comparison of Trajectories Generated From Baseline and Updated Models With Actual Human Trajectory (x vs y)

Figure 4.14: Comparison of trajectories generated from baseline and updated models with actual human trajectory

4.5.1 Analysis

The results appear quite promising. However, in this case as well, the fitted variable p would vary every time. Therefore, similar to the previous update, the genetic algorithm was run five times, and an average of the fitness value from all 5 runs was considered to ensure a more generalized result (refer to figure-4.15, figure-4.16). The resulting error values have also been documented in table-4.5 and table-4.6.

Table 4.4: Comparison of fitness values between baseline and Update-4.

Algorithm	**Fitness value**
Baseline	4.6022
Update-4	1.5299

Table 4.5: Comparison of fitness values from 5 runs of the Update-4.

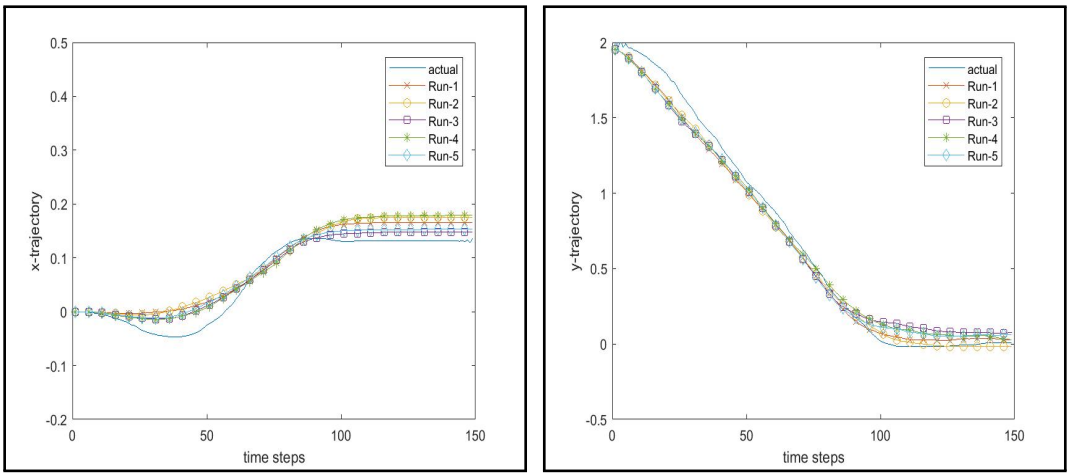
Run no.	1	2	3	4	5
fitness value(frobenius norm)	1.4372	1.5480	1.5666	1.6749	1.4229

Table-4.4 shows the comparison between the fitness values (frobenius norm in both of these cases) of the baseline algorithm and the updated algorithm. To generalize the fitness value obtained from Update-4, a mean of fitness values obtained from 5 of the algorithm has been considered. The values obtained in each run has been tabulated in table-4.5. The comparison is made only with the baseline algorithm, as the fitness function for both the algorithms was the same.

As frobenius norm was used as the fitness function in both the cases, root mean squared error (RMSE) could be used to compare the results. Table-4.6 shows the improvements obtained in the RMSE values from the first model before any updates and the new model with all the fourth update.

Table 4.6: Comparison of root mean squared error values between baseline and 5 runs of Update-4.

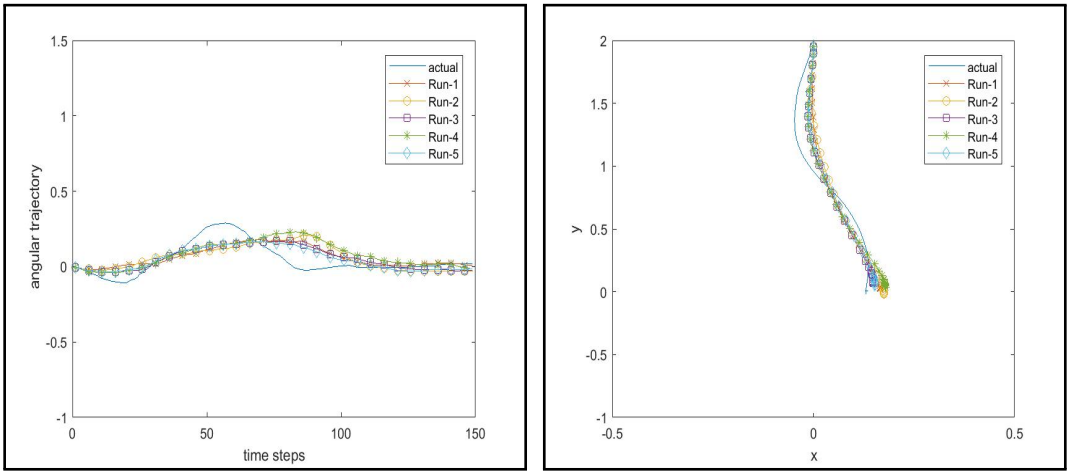
Algorithm	**RMSE (x)**	**RMSE (y)**	**RMSE (θ)**
Baseline	0.069	0.3541	0.1095
Update-4 run-1	0.0299	0.0789	0.0821
Update-4 run-2	0.0347	0.0731	0.976
Update-4 run-3	0.0199	0.1016	0.0758
Update-4 run-4	0.0324	0.09	0.0984
Update-4 run-5	0.0227	0.0903	0.0701



(a) Comparison of trajectories generated from 5 runs of the model considering stochastic input and frobenius norm as fitness function, with actual human trajectory (x vs $timestep$)

(b) Comparison of Trajectories Generated From 5 Runs of the Model Considering Stochastic Input and Frobenius Norm As Fitness Function, With Actual Human Trajectory (y vs $timestep$)

Figure 4.15: Comparison of trajectories generated from 5 runs of the model considering stochastic input and frobenius norm as fitness function, with actual human trajectory



(a) Comparison of trajectories generated from 5 runs of the model considering stochastic input and frobenius norm as fitness function, with actual human trajectory (θ vs $timestep$)

(b) Comparison of Trajectories Generated From 5 Runs of the Model Considering Stochastic Input and Frobenius Norm As Fitness Function, With Actual Human Trajectory (x vs y)

Figure 4.16: Comparison of trajectories generated from 5 runs of the model considering stochastic input and frobenius norm as fitness function, with actual human trajectory

4.6 Discussion

The results obtained from the several updates made to the baseline algorithm developed by referring to [36] have been documented for a single pair's data (Pair-9). The reason for this was to emphasize analyzing whether the proposed updates were able to generate better trajectories. The next step would be to test these models against data from the other pairs. To present that a reduction in error metric is observed in the data from other pairs have been presented as well. (tables-A.2, A.1) A comparison of the error metric (DTW distance/Frobenius norm) between the baseline algorithm and updates-1, 2, and 3 for three other pairs has been performed and presented. Another comparison between the error metrics for the Baseline model and Update-4 has also been performed and presented.

It can be observed that the first update was able to capture the trend of human trajectories and was made by changing the fitness function. This update was made to the optimization formulation and not the multi-agent collaboration model (the forward model).

The next updates were made to the multi-agent collaboration model (Algorithm-2), which runs at every iteration of the Inverse Optimal Control (Algorithm-1).

Initially, the model was built such that an agent would assume that its partner is identical to itself. The prediction of self input and the other agent's input was computed using this assumption. An amendment was made to this assumption in the second update. Therefore, each agent would have its weighting vector (cost function) and assume a partner's weighting vector (cost function) not identical to its own. This update resulted in a further reduction in the error value from 20.0257 to 16.4604.

The next update introduced a stochastic component to the control inputs of each agent. This update was able to take a big leap on error reduction, resembling the human trajectories more than the previous update. The error was reduced from 16.4604 to 9.915. However, due to the stochastic component present in the forward model of the Inverse Optimal

Control, a unique answer was not obtained repeatedly as in the previous cases. Therefore, to ensure a reliable solution, the genetic algorithm was run multiple times for the same data and the average of all the optimal fitness values returned by the genetic algorithm was considered for comparison. It was observed that all these results, the final error were reduced significantly compared to the previous updates.

The next update was made to the fitness function of the stochastic model, with every agent's assumption that the partner's cost function is not identical to self. The fitness function was changed back from DTW distance to Frobenius norm. This attempt was made to find out whether it was possible to use the new model framework, to produce a model with a time-step wise resemblance to the human model. The results were remarkable. Again, to ensure a reliable solution, the genetic algorithm was run multiple times for the same data and the average of all the optimal fitness values returned by the genetic algorithm was considered for comparison. Considering the fitness value (Frobenius norm) from the optimal weights returned by the genetic algorithm, the value was reduced from 4.6022 in the case of the baseline model with no updates to 1.5299 (mean value from all 5 runs). As Frobenius norm was used as the fitness function in both the cases, root mean squared error (RMSE) could be used to compare the results. The improvements could be significantly seen in the fitted RMSE values (table-4.6).

Chapter 5

CONCLUSIONS, SUMMARY AND FUTURE WORK

5.1 Conclusions

Human behavior in collaborative tasks can be described as an optimal control framework. However, as the task involves interaction with the other agent, the optimal inputs are predicted by every agent for themselves and their partner using a Nash equilibrium solution. Due to the underlying joint dynamics which influence the states at every time step based on both the agents' inputs, the interaction between the agents is ensured.

It can also be concluded that humans consider their partner's cost function to be different from their own. Even when the task is as mundane as collaborative couch pushing, a human agent would use its own cost function and a different cost function that they assume their partner to follow. The agent uses these cost functions to calculate the Nash equilibrium pair and predicts their own and their partner's control inputs.

It can also be concluded that human inputs are stochastic, and can be sub-optimal in many cases. This behavior has been mentioned in the literature as "bounded rationality" [54]. It can also be concluded from the resultant rationality coefficients from the inverse optimal control, that humans display fairly rational behavior resulting in sub-optimal solutions, but not the most optimal ones.

A big concern of using the stochastic input model is that, while it results in trajectories that have a remarkable resemblance to human trajectories, every time the inverse optimization (inverse optimal control) is run, a different resultant trajectory is generated. This is due to the randomness present in the optimal control generation. Even a pair of rationality coefficients (corresponding to the pair of participating humans) would result in many solutions, every time it is run. To ensure that the results are more generalized, the trajectories are run several times and an average performance from these runs is considered for comparison.

5.2 Summary

To model robotic agents that can teach, learn as well as co-operate with human subjects, they must be incorporated with a good predictive human model. Therefore, it is important to model human behavior in collaborative tasks, when they interact with other humans. While human behavior may be different based on what kind of partner they are working with, the base framework would still be similar. Human behavior in a single-agent task in itself is very difficult to model. Therefore, modeling it in collaborative multi-agent tasks is a challenging problem.

An experiment involving collaborative couch pushing was developed. A major contribution of the work is the design of a collaborative multi-agent experiment. The experiment was designed to analyze and model human-human interaction behavior. The design ensures collaboration and interaction between participating agents through the underlying joint dynamics. The experimental setup was developed from scratch. It consisted of haptic devices for human input, UNITY game development application for the physics engine and visual feedback, and a python program for data collection and experiment automation. The data collection was performed for fifteen pairs of people.

To model the high-level interaction dynamics between participating agents, it was necessary to have a low-level dynamic model (physics) of the UNITY physics engine. Therefore, system identification of the dynamic model in the UNITY engine was performed.

Another important contribution is an approach to find the behavior model of the agents using an inverse optimization formulation. A base model was developed to simulate collaborative behavior in a couch pushing-task. The model is an extended optimal control formulation. An inverse optimal control (IOC) framework was used from the literature. Several updates were made to it at different stages to obtain trajectories more similar to those of humans.

The fitness function chosen for the initial algorithm was a Frobenius norm of the between the actual and the simulated trajectories of the angular and linear position of the center of the object. However, the generated trajectories would not be able to reach the final goal. The reason is unpredictable human behavior. Humans are very intelligent entities and have more complex behavior models. It is not possible to replicate the exact human behavior, however, an attempt was made to reduce the error metric between human trajectories and simulated trajectories.

To achieve trajectories that follow similar trends as human trajectories, a new fitness function was introduced. Dynamic time warping was used as the new fitness function. The resulting trajectories were found to show more similarities with human trajectories.

An update was made to the optimal control algorithm followed by each agent individually. In the baseline algorithm, every agent predicted their own inputs and their partner's inputs under the assumption that their partner has the same cost function as them. This assumption was removed in the proposed update. In the new model, every agent would perform input predictions considering separate cost functions for themselves and their partner. An improvement in the resulting trajectories was observed.

The final update was made to the model with the addition of a stochastic component to inputs applied by the participating agents. This update resulted in trajectories much closer to the human trajectories. However, due to the stochastic or random nature of the optimal control, multiple solutions emerge for the inverse optimal control. To solve this concern, several runs of the inverse optimal control (generating trajectories and calculating fitness value) were made, and a mean of all the resulting fitness values was returned. While there was a variation in the fitted weight values, the fitness values from the final results remained closer to each other and significantly lesser than the previous update.

5.3 Future Work

Based on the results for Pair-9, the developed model showed promising results, the next step would be to apply the updated inverse optimal control model to other pairs' data.

A possible solution to solve the concern of the fitted weights considering stochastic inputs could be as follows. Instead of generating a single trajectory, a band of possible trajectories would be generated for analysis. The mean trajectory from this band could be considered.

Another solution to this issue could be a new update. In every iteration of the genetic algorithm, the trajectory would be generated using the hypothesized decision variable and the fitness value of this trajectory, compared with the human trajectory would be returned. The possible update would be to run the step of trajectory generation and fitness function calculation several times. The final fitness value would be the mean of all the fitness values generated from these different trajectories.

To begin with, emphasis will be given to modeling the behavior for tasks with pure translation goals. However, the model will also be extended to the trials with translation + rotation goals. An important aspect to consider in this case is that an additional dimension to the goal (the rotational component) may result in a more complicated higher level

dynamic. Therefore, the model must be updated further to account for these complexities in the high-level dynamics. The existing literature must be explored for more theoretical concepts that could be added to the high-level model.

The data also consists of multiple trials within a single goal-scenario (e.g. translation, translation+rotation). The developed model-fitting approach can also be used to fit inter-trial trajectories for a single pair and analyze behaviors based on the fitted decision variable. e.g. based on the rationality coefficient.

The setup can be updated from a human-human interaction to a human-robot interaction setup. The robotic agent in this case would account for human behavior considering the above model framework. There would be a few changes required in this extension. The robot may have to take a few trials runs to model the human's cost function, after which, it would be able to effectively collaborate with the human partner. The setup must also be updated to have more intuitive low-level dynamics (physics).

REFERENCES

[1] C. L. Baker and J. B. Tenenbaum, "Modeling human plan recognition using bayesian theory of mind," *Plan, activity, and intent recognition: Theory and practice*, pp. 177–204, 2014.

[2] S. Nikolaidis, J. Forlizzi, D. Hsu, J. Shah, and S. Srinivasa, "Mathematical models of adaptation in human-robot collaboration," *arXiv preprint arXiv:1707.02586*, 2017.

[3] A. De Santis, B. Siciliano, A. De Luca, and A. Bicchi, "An atlas of physical human–robot interaction," *Mechanism and Machine Theory*, vol. 43, no. 3, pp. 253–270, 2008.

[4] A. Kucukyilmaz, T. M. Sezgin, and C. Basdogan, "Conveying intentions through haptics in human-computer collaboration," in *2011 IEEE World Haptics Conference*. IEEE, 2011, pp. 421–426.

[5] N. Stefanov, A. Peer, and M. Buss, "Online intention recognition for computer-assisted teleoperation," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 5334–5339.

[6] 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, no. 1, p. 010804, 2018.

[7] Y. Li, K. P. Tee, W. L. Chan, R. Yan, Y. Chua, and D. K. Limbu, "Role adaptation of human and robot in collaborative tasks," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 5602–5607.

[8] D. Fridovich-Keil, A. Bajcsy, J. F. Fisac, S. L. Herbert, S. Wang, A. D. Dragan, and C. J. Tomlin, "Confidence-aware motion prediction for real-time collision avoidance1," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 250–265, 2020.

[9] F. Leibfried, J. Grau-Moya, and D. A. Braun, "Signaling equilibria in sensorimotor interactions," *Cognition*, vol. 141, pp. 73–86, 2015.

[10] A. Bobu, D. R. Scobee, J. F. Fisac, S. S. Sastry, and A. D. Dragan, "Less is more: Rethinking probabilistic models of human behavior," in *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, 2020, pp. 429–437.

[11] H. Wang and K. Kosuge, "Control of a robot dancer for enhancing haptic human-robot interaction in waltz," *IEEE transactions on haptics*, vol. 5, no. 3, pp. 264–273, 2012.

[12] J. Stückler and S. Behnke, "Following human guidance to cooperatively carry a large object," in *2011 11th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2011, pp. 218–223.

[13] M. Lawitzky, A. Mörtl, and S. Hirche, "Load sharing in human-robot cooperative manipulation," in *19th International Symposium in Robot and Human Interactive Communication*. IEEE, 2010, pp. 185–191.

[14] A. Bussy, A. Kheddar, A. Crosnier, and F. Keith, "Human-humanoid haptic joint object transportation case study," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 3633–3638.

[15] Y. Wang, Y. Sheng, J. Wang, and W. Zhang, "Optimal collision-free robot trajectory generation based on time series prediction of human motion," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 226–233, 2017.

[16] M. Kawato, "Internal models for motor control and trajectory planning," *Current opinion in neurobiology*, vol. 9, no. 6, pp. 718–727, 1999.

[17] D. Vogt, S. Stepputtis, S. Grehl, B. Jung, and H. B. Amor, "A system for learning continuous human-robot interactions from human-human demonstrations," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2882–2889.

[18] N. Jarrassé, T. Charalambous, and E. Burdet, "A framework to describe, analyze and generate interactive motor behaviors," *PloS one*, vol. 7, no. 11, p. e49945, 2012.

[19] S. O. Oguz, A. Kucukyilmaz, T. M. Sezgin, and C. Basdogan, "Supporting negotiation behavior with haptics-enabled human-computer interfaces," *IEEE transactions on haptics*, vol. 5, no. 3, pp. 274–284, 2012.

[20] M. Lanctot, E. Lockhart, J.-B. Lespiau, V. Zambaldi, S. Upadhyay, J. Pérolat, S. Srinivasan, F. Timbers, K. Tuyls, S. Omidshafiei *et al.*, "Openspiel: A framework for reinforcement learning in games," *arXiv preprint arXiv:1908.09453*, 2019.

[21] S. Jain and B. Argall, "Probabilistic human intent recognition for shared autonomy in assistive robotics," *ACM Transactions on Human-Robot Interaction (THRI)*, vol. 9, no. 1, pp. 1–23, 2019.

[22] ——, "Recursive bayesian human intent recognition in shared-control robotics," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3905–3912.

[23] X. Yu, W. He, Y. Li, C. Xue, J. Li, J. Zou, and C. Yang, "Bayesian estimation of human impedance and motion intention for human-robot collaboration," *IEEE transactions on cybernetics*, 2019.

[24] W. Yoshida, R. J. Dolan, and K. J. Friston, "Game theory of mind," *PLoS computational biology*, vol. 4, no. 12, 2008.

[25] N. Bard, J. N. Foerster, S. Chandar, N. Burch, M. Lanctot, H. F. Song, E. Parisotto, V. Dumoulin, S. Moitra, E. Hughes *et al.*, "The hanabi challenge: A new frontier for ai research," *Artificial Intelligence*, vol. 280, p. 103216, 2020.

[26] R. Choudhury, G. Swamy, D. Hadfield-Menell, and A. D. Dragan, "On the utility of model learning in hri," in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2019, pp. 317–325.

[27] E. Kamenica and M. Gentzkow, "Bayesian persuasion," *American Economic Review*, vol. 101, no. 6, pp. 2590–2615, 2011.

[28] Z. Wang, A. Peer, and M. Buss, "An hmm approach to realistic haptic human-robot interaction," in *World Haptics 2009-Third Joint EuroHaptics conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*. IEEE, 2009, pp. 374–379.

[29] C. Baker, R. Saxe, and J. Tenenbaum, "Bayesian theory of mind: Modeling joint belief-desire attribution," in *Proceedings of the annual meeting of the cognitive science society*, vol. 33, no. 33, 2011.

[30] Y. Wang, Y. Ren, S. Elliott, and W. Zhang, "Enabling courteous vehicle interactions through game-based and dynamics-aware intent inference," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 2, pp. 217–228, 2020.

[31] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 1.

[32] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan, "Cooperative inverse reinforcement learning," in *Advances in neural information processing systems*, 2016, pp. 3909–3917.

[33] Y. Wang, G. J. Lematta, C.-P. Hsiung, K. A. Rahm, E. K. Chiou, and W. Zhang, "Quantitative modeling and analysis of reliance in physical human–machine coordination," *Journal of Mechanisms and Robotics*, vol. 11, no. 6, 2019.

[34] C. Liu, W. Zhang, and M. Tomizuka, "Who to blame? learning and control strategies with information asymmetry," in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 4859–4864.

[35] H. El-Hussieny, A. Abouelsoud, S. F. Assal, and S. M. Megahed, "Adaptive learning of human motor behaviors: An evolving inverse optimal control approach," *Engineering Applications of Artificial Intelligence*, vol. 50, pp. 115–124, 2016.

[36] H. El-Hussieny and J.-H. Ryu, "Inverse discounted-based lqr algorithm for learning human movement behaviors," *Applied Intelligence*, vol. 49, no. 4, pp. 1489–1501, 2019.

[37] M. K. O'Malley, A. Gupta, M. Gen, and Y. Li, "Shared control in haptic systems for performance enhancement and training," 2006.

[38] Y. Li, J. C. Huegel, V. Patoglu, and M. K. O'Malley, "Progressive shared control for training in virtual environments," in *World Haptics 2009-Third Joint EuroHaptics conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*. IEEE, 2009, pp. 332–337.

[39] D. J. Block, M. B. Michelotti, and R. S. Sreenivas, "Application of the novint falcon haptic device as an actuator in real-time control," *Paladyn, Journal of Behavioral Robotics*, vol. 4, no. 3, pp. 182–193, 2013.

[40] S. S. Nudehi, R. Mukherjee, and M. Ghodoussi, "A shared-control approach to haptic interface design for minimally invasive telesurgical training," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, pp. 588–592, 2005.

[41] K. B. Reed and M. A. Peshkin, "Physical collaboration of human-human and human-robot teams," *IEEE Transactions on Haptics*, vol. 1, no. 2, pp. 108–120, 2008.

[42] Y. Li, V. Patoglu, and M. K. O'malley, "Shared control for training in virtual environments: Learning through demonstration," in *Proceedings of EuroHaptics*, 2006, pp. 93–99.

[43] D. Powell and M. K. O'Malley, "The task-dependent efficacy of shared-control haptic guidance paradigms," *IEEE transactions on haptics*, vol. 5, no. 3, pp. 208–219, 2012.

[44] N. Stefanov, A. Peer, and M. Buss, "Role determination in human-human interaction," in *World Haptics 2009-Third Joint EuroHaptics conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*. IEEE, 2009, pp. 51–56.

[45] C. E. Madan, A. Kucukyilmaz, T. M. Sezgin, and C. Basdogan, "Recognition of haptic interaction patterns in dyadic joint object manipulation," *IEEE transactions on haptics*, vol. 8, no. 1, pp. 54–66, 2014.

[46] R. Groten, D. Feth, R. L. Klatzky, and A. Peer, "The role of haptic feedback for the integration of intentions in shared task execution," *IEEE Transactions on Haptics*, vol. 6, no. 1, pp. 94–105, 2012.

[47] A. Melendez-Calderon, V. Komisar, G. Ganesh, and E. Burdet, "Classification of strategies for disturbance attenuation in human-human collaborative tasks," in *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2011, pp. 2364–2367.

[48] D. Feth, R. Groten, A. Peer, S. Hirche, and M. Buss, "Performance related energy exchange in haptic human-human interaction in a shared virtual object manipulation task," in *World Haptics 2009-Third Joint EuroHaptics conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*. IEEE, 2009, pp. 338–343.

[49] S. M. LaValle and S. A. Hutchinson, "Path selection and coordination for multiple robots via nash equilibria," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, 1994, pp. 1847–1852 vol.3.

[50] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series." in *KDD workshop*, vol. 10, no. 16. Seattle, WA, USA:, 1994, pp. 359–370.

[51] S. H. Kim, H. S. Lee, H. J. Ko, S. H. Jeong, H. W. Byun, and K. J. Oh, "Pattern matching trading system based on the dynamic time warping algorithm," *Sustainability*, vol. 10, no. 12, p. 4641, 2018.

[52] N. L. Olsen, B. Markussen, and L. L. Raket, "Simultaneous inference for misaligned multivariate functional data," *arXiv preprint arXiv:1606.03295*, 2016.

[53] K. Yu, J. Mason, and J. Oglesby, "Speaker recognition using hidden markov models, dynamic time warping and vector quantisation," *IEE Proceedings-Vision, Image and Signal Processing*, vol. 142, no. 5, pp. 313–318, 1995.

[54] R. Nakahashi and S. Yamada, "Modeling human inference of others' intentions in complex situations with plan predictability bias," *arXiv preprint arXiv:1805.06248*, 2018.

APPENDIX A

MODEL ANALYSIS FOR 4 DIFFERENT PAIRS

Table A.1: Comparison of fitness values between the baseline algorithm and updates 1,2,3 (comparison metric is DTW distance).

Pair	Baseline	Update-1	Update-2	Update-3
9	31.1563	20.0257	16.4604	9.915
10	38.09	19.89	13.91	9.6132
11	45.49	11.7	10.36	9.7163
14	30.26	23.32	22.98	8.8355

Table A.2: Comparison of fitness values between the baseline algorithm and update-4 (comparison metric is Frobenius norm).

Pair	Baseline	Update-4
9	4.6022	1.5299
10	5.3417	2.3032
11	4.3412	1.8355
14	3.8503	2.0598

APPENDIX B

IRB APPROVAL FOR DATA COLLECTION INVOLVING HUMAN SUBJECTS

ASU Knowledge Enterprise Development

EXEMPTION GRANTED

Wenlong Zhang
IAFSE-PS: Polytechnic Engineering Programs (EGR)
-
Wenlong.Zhang@asu.edu

Dear Wenlong Zhang:

On 2/20/2020 the ASU IRB reviewed the following protocol:

Type of Review:	Initial Study
Title:	Modeling of interaction behavior in humans during collaborative tasks.
Investigator:	Wenlong Zhang
IRB ID:	STUDY00011502
Funding:	None
Grant Title:	None
Grant ID:	None
Documents Reviewed:	• Consent form.pdf, Category: Consent Form; • Debriefing.pdf, Category: Participant materials (specific directions for them); • Feedback sheet.pdf, Category: Measures (Survey questions/Interview questions /interview guides/focus group questions); • flyer.pdf, Category: Recruitment Materials; • Instruction sheet-1.pdf, Category: Participant materials (specific directions for them); • Instruction sheet-2.pdf, Category: Participant materials (specific directions for them); • IRB Documents, Category: IRB Protocol; • Recruitment Email.pdf, Category: Recruitment Materials;

The IRB determined that the protocol is considered exempt pursuant to Federal Regulations 45CFR46 on 2/20/2020.

Figure B.1: IRB page-1

In conducting this protocol you are required to follow the requirements listed in the INVESTIGATOR MANUAL (HRP-103).

Sincerely,

IRB Administrator

cc: Pallavi Shintre
Pallavi Shintre
Wenlong Zhang
Yiwei Wang
Venkatesh Vaidyanathan
Sunny Amatya

Figure B.2: IRB page- 2