

Neural Network Full-Body Human Predictive Models and their Use for Coordinated Robot Motion Planning

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik der
Universität Stuttgart zur Erlangung der Würde eines Doktors der
Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Philipp Johannes Kratzer

aus Heidenheim an der Brenz

Hauptberichter: Prof. Dr. Marc Toussaint

Mitberichter: Dr. Jean-Baptiste Mouret

Tag der mündlichen Prüfung: 23.09.2022

Institut für Parallele und Verteilte Systeme der Universität Stuttgart

2022

Abstract

Due to various improvements in robotics hardware and software in the past decades, today's robots are capable of performing a wide variety of tasks. However, while robots have been working alongside humans in factories since the 1960's, to this day, robots are still fenced in cages and relegated to repetitive tasks. Robots interacting with humans on a regular basis on streets, or in the household, are still scenarios from science fiction and far away from today's reality.

A reason for this is the lack of *Human-Awareness*. Human motion follows complex patterns that are hard to model. For a robot to interact among humans, it is required to be able to predict human motion and plan its motion by considering this prediction. An example of a successful human-robot space-sharing strategy is one that minimally intervenes in the human plan while maintaining the ability for the robot to achieve its own goal.

In this thesis, we focus on the problem of human motion prediction. We design and evaluate prediction techniques to forecast observed human full-body motion. To this aim, we use neural network techniques that are capable to learn from large data sets. We propose a novel framework for changing the prediction while retaining its predictive capabilities. Our technique allows to take the environment into account and experiments show that this improves the predictive capabilities of the model. On further examples, we combine the techniques with intent prediction to achieve longer-horizon predictions, and use hierarchical predictions for task sequences.

Furthermore, we investigate the problem of planning coordinated human-robot trajectories. To achieve this, we extend our framework to jointly co-optimize human predictions and robot motion allowing to account for coordination paradigms that depend on both agents simultaneously. We demonstrate with examples of handovers, joint collision avoidance, and a shared goal constraint that this framework is able to plan coordinated human-robot motion trajectories and outperforms a variety of

baselines.

Finally, we present *MoGaze*, a dataset with long, coherent full-body motions of pick and place tasks, and discuss the ideas and design considerations. The underlying data is a major element of data-driven models and our experiments demonstrate the important value of a sophisticated dataset.

Zusammenfassung

In den vergangenen Jahrzehnten gab es deutliche Verbesserungen in Hardware und Software für Roboter, wodurch diese unzählige Aufgaben erledigen können. So arbeiten Roboter und Menschen bereits seit den 1960er Jahren gemeinsam in Fabriken, jedoch nicht immer miteinander. Zumeist befinden sich Roboter in abgetrennten Bereichen, wo sie repetitiven Aufgaben nachgehen. Roboter, die mit Menschen interagieren, zum Beispiel im Haushalt oder auf der Straße, sind trotz den enormen technischen Fortschritten der heutigen Zeit kaum anzutreffen und nach wie vor nur ein Szenario aus dem Science-Fiction Genre.

Ein Grund dafür ist, dass die Mensch-Roboter-Interaktion Algorithmen benötigt, die den Menschen und dessen Bewegungen berücksichtigt. Allerdings folgen Bewegungen von Menschen komplexen Mustern und sind dadurch schwierig zu modellieren. Für einen Roboter, welcher in der Nähe von Menschen interagiert, ist es notwendig, die menschlichen Bewegungen vorherzusehen und seine Bewegungen entsprechend dieser Vorhersage anzupassen. Eine erfolgreiche Interaktionsstrategie für den Roboter hindert den Mensch nicht an der Erfüllung seiner Aufgabe, während die Fähigkeit die eigene Aufgabe zu erfüllen, bestehen bleibt.

Die vorliegende Arbeit widmet sich dem Problem, menschliche Bewegungen vorherzusagen. Dafür werden Vorhersagetechniken erstellt und evaluiert, die auf Basis von beobachtete Ganzkörperbewegungen des Menschen prognostizieren, wie diese Bewegungen fortgeführt werden. Zu diesem Zweck kommen Techniken des maschinellen Lernens, basierend auf künstlichen neuronalen Netzen, zum Einsatz. Diese Techniken ermöglichen das Lernen von Bewegungen anhand von großen Datenmengen an Beispielmovements. Im Rahmen dieser Arbeit werden Algorithmen vorgestellt, die die Vorhersagen eines neuronalen Netzes durch Berücksichtigung der Umgebung verändern können. Experimente bestätigen, dass auf diese Weise die Vorhersage verbessert werden kann. Mit weiteren Untersuchungen wird gezeigt, wie sich diese Technik mit

der Vorhersage von Absichten und mit hierarchischen Modellen kombinieren lässt, um Prognosen für längere Zeitspannen zu treffen.

Darüber hinaus wird in dieser Arbeit das Problem von koordinierter Mensch-Roboter Interaktion betrachtet. Dazu werden die Algorithmen erweitert, so dass zeitgleich eine Vorhersage für die Bewegung des Menschen als auch ein Plan für die Bewegung des Roboters erstellt werden können. Dies erlaubt es, Bedingungen zu berücksichtigen, die beide Agenten gleichermaßen betreffen. Anhand von Beispielen wie Objektübergaben, Verhindern von Kollisionen zwischen Mensch und Roboter und gemeinsamen Zielbedingungen wird demonstriert, dass unter Verwendung der vorgestellten Techniken das Planen koordinierter Bewegungsabläufe möglich ist.

Schließlich wird ein Datensatz, *MoGaze*, vorgestellt, der lange, zusammenhängende Ganzkörperbewegungen des Menschen enthält. Die zugrunde liegenden Daten sind ein wichtiger Bestandteil der datengestützten Techniken des maschinellen Lernens, und die in dieser Arbeit vorgestellten Experimente bestätigen die Vorzüge eines ausgeklügelten Datensatzes.

Acknowledgements

First, I would like to thank **Marc Toussaint**, for his support throughout the past years and the helpful advice. I would like to thank **Jim Mainprice**, it was a pleasure to work in his group. Our weekly discussions about many details and ideas, and the valuable feedback during the past years, substantially helped to form this thesis. I thank **Jean-Baptiste Mouret** for being on my Ph.D. committee.

I would also like to thank **Ruth Schulz** for her support already before my Ph.D. and for inspiring me to continue research in the interesting field of Human-Robot Interaction. I thank my Ph.D. colleagues **Yoojin Oh** and **Janik Hager**, who started about the same time as me and supported me the whole time.

I am grateful that I was within the International Max Planck Research School for Intelligent Systems (IMPRS-IS) and want to thank both, **Frank Allgöwer** and **Georg Martius**, for being on my thesis advisory committee, and for the helpful discussions and good advice.

During the time, I am glad that I was able to work alongside an amazing team of people. I really enjoyed my time at the Machine Learning and Robotics Lab in Stuttgart. A huge thanks goes to all of you, **Andreas Orthey**, **Camille Piquetal**, **Carola Stahl**, **Daniel Hennes**, **Danny Drieß**, **Heiko Zimmermann**, **Hung Ngo**, **Ingmar Schubert**, **Jung-Su Ha**, **Joaquim Ortiz de Haro**, **Matthew Bernstein**, **Ozgur Oguz**, **Peter Englert**, **Sabine Janzen**, **Sabrina Hoppe**, **Vien Ngo**, and **Valentin Hartmann**.

I want to thank the Master and Bachelor students that I supervised or co-supervised during their thesis or projects and that helped me with conducting experiments and collecting data, namely **An Thai Le**, **Jan Hoffmann**, **Johannes Seitz**, **Niteesh Midlagajni**, **Rohit Prakash**, **Simon Bihlmaier** and **Simon Hagenmayer**.

Finally, I want to thank my family, **Juliane Kratzer**, **Kai Kratzer**, **Tim**

Klaiber, Ulrike Kratzer and **Thomas Kratzer**, and my partner **Julia Menke**,
for their love and support during the whole time.

Thank you all.

Contents

List of Figures	xiii
List of Tables	xvi
List of Abbreviations and Acronyms	xix
List of Symbols	xxi
1 Introduction	1
1.1 Research Questions	3
1.2 Research Context	4
1.3 Thesis Outline	4
2 Background	7
2.1 Recurrent Neural Networks	7
2.1.1 Recurrent Neural Network with a Recurrent Hidden State	8
2.1.2 Computing Gradients	9
2.1.3 Gated Recurrent Neural Networks	9
2.1.4 Sequence-to-Sequence Prediction with a Recurrent Neural Network	11
2.2 Trajectory Optimization	12
2.2.1 System Equations	12
2.2.2 Problem Formulation	13
2.2.3 Direct Shooting and Direct Collocation Approaches	13
2.2.4 Constrained Optimization Methods	15
2.3 Summary	16
3 Human Motion Forecasting	17

3.1	The Motion Prediction Problem	18
3.2	Related Work on Human Motion Prediction	19
3.2.1	Data-Driven Motion Prediction	20
3.3	Modeling Human Motion for Neural Network Prediction	22
3.4	Recurrent Neural Network Approaches	23
3.4.1	Basic Recurrent Neural Network	23
3.4.2	Residual Connections	24
3.4.3	Velocity Connections	25
3.4.4	Rotation Representation	26
3.4.5	Training Procedure	27
3.5	Empirical Evaluation	28
3.5.1	Rotation Representation	28
3.5.2	Architectures and Baselines	29
3.5.3	Training Sets and Hyperparameter Search	30
3.5.4	Results of the Hyperparameter Search on the Validation Set	32
3.5.5	Results on the Test Set	33
3.6	Open Challenges in Human Motion Prediction	35
3.7	Summary	36
4	Controllable Motion Prediction	39
4.1	Related Work on Human Motion Generation	40
4.2	Formulating the Motion Prediction Problem as Nonlinear Program	42
4.2.1	Trajectory Optimization with <i>Shooting</i>	43
4.2.2	Controlled Human Body Dynamics	44
4.3	Solving the Nonlinear Program	46
4.3.1	Primal-Dual Interior Point Method for Nonlinear Programming	47
4.4	Motion Objectives and Constraints	48
4.4.1	Optimization Objective	48
4.4.2	User-Defined Constraints	49
4.5	Empirical Evaluation	51
4.5.1	Goal Constraint Experiments with One Actor	51
4.5.2	Collision Constraint Experiments with One Actor	53
4.5.3	Goal Constraint Experiments on MoGaze	54
4.5.4	Comparing Different Possibilities for Specifying Controls	58
4.6	Alternative Method with a Gaussian Process as Human Model	61

4.6.1	Method	61
4.6.2	Experiments	64
4.6.3	Comparison to the Recurrent Neural Network Approach . . .	64
4.7	Summary	65
5	Co-Optimizing Human-Robot Trajectories	67
5.1	Related Work	68
5.1.1	Human-Aware Motion Planning	68
5.1.2	Trajectory Optimization	70
5.2	Framework Overview	71
5.3	A Nonlinear Program for Human-Robot Collaboration Problems . . .	72
5.3.1	Nonlinear Program Formulation	72
5.3.2	Solving the Nonlinear Program	74
5.4	Objective and Constraints	75
5.4.1	Objective	75
5.4.2	Single Agent Constraints	76
5.4.3	Joint Constraints	76
5.5	Agent Dynamics Functions	77
5.6	Empirical Evaluation	78
5.6.1	Baselines	78
5.6.2	Joint Collision Avoidance Experiments	79
5.6.3	Computation Time	85
5.6.4	Handover Experiments	85
5.6.5	Pickup before Handover	88
5.7	Discussion and Limitations	89
5.7.1	Dataset	89
5.7.2	Trajectory Optimization	89
5.8	Summary	90
6	Intent Prediction and Planning for Longer Tasks	91
6.1	Related Work	92
6.1.1	Intent Prediction	92
6.1.2	Long Time-Horizons	94
6.2	Affordances for Intent Prediction	95
6.2.1	Overview	95

6.2.2	The Affordance Networks	96
6.2.3	Combination with the Controllable Model for Full-Body Prediction	98
6.2.4	Experiments	98
6.3	Gaze for Intent Prediction	101
6.4	Combining Hierarchical Motion Prediction with Task and Motion Planning	102
6.4.1	Hierarchical Motion Prediction	102
6.4.2	Task and Motion Planning	104
6.4.3	Experiments	105
6.5	Summary	106
7	Datasets	107
7.1	Related Work on Datasets of Human Motion	108
7.2	The <i>MoGaze</i> Full-Body Dataset	110
7.2.1	Setup for Recording	110
7.2.2	Calibration of Motion Capture and Eye-Tracker	113
7.2.3	The Human Model	113
7.2.4	Objects	114
7.2.5	Tasks	115
7.2.6	Recording Procedure	116
7.2.7	Data Labeling	117
7.2.8	Discussion and Limitations	118
7.3	Summary	118
8	Conclusions	121
8.1	Future Work	123
	Bibliography	125
A	Appendix	143
A.1	Additional Datasets	143
A.1.1	Additional Human Motion Data	143
A.1.2	Synthetic Datasets for Testing with Non-Human Data	146
A.2	Goal Constraint Experiment: All Baselines	147

List of Figures

2.1	Recurrent Neural Network	8
2.2	Unrolled Recurrent Neural Network	8
2.3	Gated Recurrent Unit	10
2.4	Encoder-Decoder RNN for Sequence-to-Sequence Prediction	11
2.5	Shooting Formulation for Trajectory Optimization	13
2.6	Collocation Formulation for Trajectory Optimization	14
3.1	The Motion Prediction Problem	18
3.2	Human Motion Prediction with a Recurrent Neural Network	23
3.3	Human Motion Prediction with a Recurrent Neural Network and Three Stacked GRU Layers	24
3.4	Human Motion Prediction with a Recurrent Neural Network and Residual Connections	25
3.5	Human Motion Prediction with a Recurrent Neural Network and Velocity Connections	26
3.6	Changing the Rotation Representation	27
3.7	Ambiguity Issues with Exponential Maps	28
3.8	Training Curves for Six-Dimensional and Exponential Map Rotation Representations	29
3.9	Hyperparameter Search	31
3.10	Training Curves for Different Architectures	32
3.11	Example Trajectories of Motion Forecasting	34
4.1	Controllable Trajectory Prediction Example	40
4.2	Adding Controls to the Recurrent Neural Network	45
4.3	Incorporating Constraints into the Recurrent Neural Network	49
4.4	Signed Distance Field on the MoGaze Dataset	50

LIST OF FIGURES

4.5	Example Prediction with Collision Avoidance	53
4.6	Example Trajectories with Goal Constraint on MoGaze	56
4.7	Example Trajectories with the Gaussian Process Approach	64
5.1	Circular Dependency of Planning and Prediction	68
5.2	Overview of the Co-Optimization Framework	72
5.3	Influence of the Robot Cost Scaling Parameter	82
5.4	Robot Avoiding the Human	83
5.5	Joint Collision Avoidance	84
5.6	Handover Examples	86
5.7	Joint Goal Example	88
6.1	Affordance Prediction System Overview	95
6.2	Affordance Network Architectures	96
6.3	Plane Features for the Placeability	97
6.4	Affordance Prediction for Placing a Jug	99
6.5	Combined Intent and Full-Body Prediction Example	100
6.6	Gaze for Intent Prediction	101
6.7	Robot and Human Jointly Setup a Table	103
6.8	Example Human Robot Trajectory with Task and Motion Planning . . .	105
7.1	Example Trajectory of the Mogaze Dataset	110
7.2	Dataset Capture Setup	111
7.3	Eye-tracker with Markers	111
7.4	Data Capture Overview	112
7.5	Movable Objects in the MoGaze Dataset	114
7.6	Chair Configurations for Data Recording	114
7.7	Number of Pick and Place Actions	117
7.8	Duration of a Placing Action	117
A.1	Additional Human Motion Datasets	143
A.2	Walking Patterns	144
A.3	Data with two Humans	145
A.4	Example Trajectory of the Four Joint Robot	146
A.5	Example Trajectories of the One-Dimensional Test Data	147
A.6	Example Trajectories for the Initial Baselines	148

A.7	Example Trajectories for the Sample Baselines	149
A.8	Example Trajectories of Our Method	150

List of Tables

1.1	Research Questions	3
3.1	Forecasting on the Test Set	33
4.1	Goal Constraint Results	52
4.2	Collision Constraint Results	54
4.3	Goal Constraint Results on MoGaze	57
4.4	Possibilities for Specifying Controls	58
4.5	Comparison of Different Possibilities for Specifying Controls	60
5.1	Collision Experiments with Baselines.	80
5.2	Handover Experiments with Baselines.	87
6.1	Full-Body Prediction with Intent	100
6.2	Example High-Level Trajectory	103
6.3	Dynamic LGP with Human Prediction	105
7.1	Joints of the Human Kinematic Model	113
7.2	Instructions for Participant Tasks	115

List of Algorithms

4.1	Human Motion Prediction with Trajectory Optimization	48
4.2	Sample Baseline for Goal Prediction	55
5.1	Human-Robot Joint Trajectory Optimization	75
5.2	Sample Baseline for Coordinated Planning and Prediction	79
6.1	Dynamic Task and Motion Planning	104

List of Abbreviations and Acronyms

- BFGS** Broyden-Fletcher-Goldfarb-Shanno. 47, 48, 75
- BPTT** Backpropagation Through Time. 9
- CG** Conjugate Gradients. 63
- CNN** Convolutional Neural Network. 93
- CPU** Central Processing Unit. 85
- CRF** Conditional Random Field. 21
- FD** Finite Differences. 58–60
- FK** Forward Kinematics. 49–51
- GP** Gaussian Process. 21, 61–65
- GPLVM** Gaussian Process Latent Variable Model. 21, 41
- GPU** Graphics Processing Unit. 85
- GRU** Gated Recurrent Unit. 10, 21, 22, 24–27, 30–35, 45, 49, 55, 148–150
- HMM** Hidden Markov Model. 20, 92
- HRC** Human Robot Collaboration. 2, 4, 5, 17, 19, 20, 36, 65, 67, 68, 70, 72, 73, 85, 90, 92, 95, 121–123
- IOC** Inverse Optimal Control. 20
- LGP** Logic-Geometric Programming. 94, 95, 104, 105

- LSTM** Long Short Term Memory. 10, 24, 93
- MaxEnt IRL** Maximum Entropy Inverse Reinforcement Learning. 103
- MDN** Mixture Density Network. 97, 99
- NLL** Neg-Log Likelihood. 97, 98
- NLML** Negative Log Marginal Likelihood. 63
- NLP** Nonlinear Program. 5, 13, 15, 39, 43, 47, 68, 70, 72–74, 104, 121
- PVRED** Position-Velocity Recurrent Encoder-Decoder Model. 22, 30–37, 44, 51, 52, 54–59, 61, 78, 95, 99, 103, 148–150
- RBF** Radial Basis Function. 62, 63
- RNN** Recurrent Neural Network. 2, 7–9, 11, 13, 16, 21–28, 30, 36, 37, 39, 42, 44–47, 49, 51, 52, 54, 58–62, 64, 65, 71, 77, 89, 90, 101, 102, 121, 123
- SDF** Signed Distance Field. 49–51, 54, 76, 79, 85, 86, 97
- SLSQP** Sequential Least Square Programming. 64
- SQP** Sequential-Quadratic Programming. 15
- SVM** Support Vector Machine. 93
- TAMP** Task and Motion Planning. 5, 94, 95, 102, 105, 106, 123
- vMF** von Mises-Fisher. 98

List of Symbols

Throughout the thesis the following notation is used: Scalars are lowercase (a, b, c), vectors are bold lowercase ($\mathbf{x}, \mathbf{y}, \mathbf{z}$), matrices are bold uppercase ($\mathbf{A}, \mathbf{B}, \mathbf{C}$), and sets are calligraphic style letters ($\mathcal{A}, \mathcal{B}, \mathcal{C}$). We write a sequence of vectors using the following notation: $\mathbf{x}_{i:j} = (\mathbf{x}_i, \mathbf{x}_{i+1}, \dots, \mathbf{x}_j)$.

Indices

t	Arbitrary time index
k	Present time frame/start of prediction
T	Prediction horizon

General Agent

\mathbf{x}^a	Agent joint state
\mathbf{u}^a	Agent controls
f^a	Agent dynamics function
\tilde{f}^a	Unrolled agent dynamics function
ϕ^a	Forward kinematics map of an agent

Human

\mathbf{x}^H	Human joint state
\mathbf{x}'^H	Human joint state prediction
\mathbf{v}^H	Human joint velocity

LIST OF SYMBOLS

\mathbf{v}^H	Human joint velocity prediction
\mathbf{u}^H	Human controls
f^H	Human dynamics function
\tilde{f}^H	Unrolled human dynamics function
c^H	Human trajectory cost function
ϕ^H	Human forward kinematics map

Robot

\mathbf{x}^R	Robot joint state
\mathbf{u}^R	Robot controls
\mathbf{v}^R	Robot joint velocity
f^R	Robot dynamics function
\tilde{f}^R	Unrolled robot dynamics function
c^R	Robot trajectory cost function
ϕ^R	Robot forward kinematics map

Neural Networks

ℓ	Loss function
$\boldsymbol{\theta}$	Weights
\mathbf{h}	Hidden state
l	Number of layers
\mathcal{D}	Dataset
\mathcal{B}	Batch of dataset

Optimization

h	Equality constraint
g	Inequality constraint
A	Metric
α^H	Weight of human costs
α^R	Weight of robot costs

Probabilities

p	Likelihood
\mathcal{N}	Gaussian distribution
μ	Mean
σ^2	Variance

Introduction

IN the past decades, countless important improvements in robotics occurred. Hardware, such as computers, sensors, and motors, are getting smaller and more powerful at the same time. Many advances in software allow robust robot control, planning of complex motions, and detecting objects accurately. Moreover, several problems in robotics benefit from the advances of deep learning, a machine learning technique that copes well with the large amounts of data that can be acquired by nowadays robotics systems. An example of such a domain is detecting possible grasps for unseen objects [1, 2].

This success in the field of robotics can also be seen in an increasing number of industrial robots, with three million industrial used robots in 2021 [3]. While robots have been working alongside humans in factories since the 1960's, to this day, robots are still fenced in cages and relegated to repetitive tasks. For robots, dynamic and stochastic environments are still challenging to cope with.

Human motion is influenced by many internal and external stimuli and consequently exhibits a stochastic nature. Thus, an environment where a robot is sharing space with a human is both: dynamic and stochastic. Robots interacting with humans on a regular basis on streets, or in the household, are still scenarios from science fiction and far from today's reality.

Today's robots' capabilities in *Human-Awareness* are limited. However, there are many domains, in which a human-aware robot interacting in close-proximity with humans would be beneficial, for example, robots could assist in repetitive tasks in nursing care facilities or hospitals, fields where increased skilled labor shortage is

predicted by many sources [4, 5]. Further possible domains that would benefit from human-aware robots are, inter alia, service robotics, manufacturing, construction, or disaster relief.

Attention to the potential of Human Robot Collaboration (HRC) also raises in industry. Collaborative robots – so called cobots – are designed to be able to work in close proximity to humans [6]. HRC benefits from the flexibility of humans and the precision, strength and inexhaustibility of robots.

One of the key abilities to achieve human-awareness is the ability to anticipate human actions. A successful coordinated motion plan for the robot needs to take into account how humans in close proximity are likely to move. Respecting a prediction for the human during motion planning for the robot allows to plan trajectories that are safe and efficient. For example, the robot motion can be planned to avoid collisions with the human prediction and to ensure human comfort.

Additionally, the ability to forward simulate human motion dependent on the robot actions is useful for planning coordinated motions. A challenge is that human behavior changes depending on what the robot is doing and the robot’s plan changes based on what the human is doing. This is especially important when planning coordinated motion in close distance, for example, for handovers, or for assembling an engine part together.

As a consequence, to achieve human-awareness, a predictive human model is required. However, human motion is the result of complex biomechanical processes that are challenging to model. Thus, state-of-the-art work on full-body motion prediction resort to data-driven models, such as Recurrent Neural Networks (RNNs) [7, 8]. A drawback of these architectures is that they purely forecast human motion and are not controllable, which makes it impossible to adapt the prediction based on different external criteria, such as, future goals, environmental conditions, or even a robot partner.

In contrast, in computer graphics exist controllable human models [9]. However, those models are designed for generating motion and lack the ability to forecast an observed trajectory. This ability to forecast observed trajectories is required in the field of robotics, as the robot usually needs to adapt to the observed trajectories of a human in close proximity.

In this thesis, we seek to improve human-awareness in motion planning for robotics. We build on state-of-the-art, deep learning based motion prediction frameworks

to create a predictive human behavior model and propose a novel framework for coordinated motion planning, which is able to plan and predict concurrently.

1.1 Research Questions

We identified six research questions that are important in the field of human motion prediction. The research questions can be seen in Table 1.1.

The most fundamental problem in human motion prediction is forecasting an observed full-body motion trajectory. RQ1 addresses this problem of short-term motion prediction. As state-of-the-art for motion prediction focuses on data-driven methods, this thesis focuses on data-driven machine learning techniques for motion prediction.

A follow up to RQ1 is introduced by RQ2: Given a model that is able to forecast an observed motion trajectory, how can the predictive model be adapted to fulfill external criteria? External criteria can arise from environmental context or also to simulate multiple futures.

Considering that we want to plan robot motion with respect to the predictive model, leads us to RQ3. Planning coordinated motion is challenging, since the prediction of the human and the robot plan influence each other. For example, in a

Table 1.1: Summary of the research questions in this thesis

Topic	No.	Research Question	Chapter
Forecasting	RQ1	Given a dataset of example motion trajectories, how can human full-body motion be predicted?	3
Adapting	RQ2	How can we adapt a data-driven human predictive model to fulfill external criteria?	4
Planning	RQ3	How can we plan robot motion to coordinate with the predictive human model?	5
Intent	RQ4	How can the intention of a human be estimated?	6
Long-term	RQ5	Which challenges arise from longer time horizons and how can they be addressed for planning and prediction?	6
Datasets	RQ6	What are design consideration and best practices for capturing human motion datasets?	7

handover scenario, the human would reach towards the hand of the robot and vice versa. Thus, RQ3 is a follow up to RQ2, since a controllable human predictive model is needed for coordinated planning.

While RQ1 focuses on short-term full-body motion, RQ4 raises the question for estimating the intention of the human. This is important for longer time horizons. Therefore, RQ5 is related, as RQ5 considers how motion can be predicted and planned for longer time horizons.

Finally, an important underlying part of human motion prediction is the dataset, that the predictive models are trained on. RQ6 is devoted to address how to capture human motion data and asks for the necessary design considerations for capturing such data.

1.2 Research Context

The research presented in this thesis was conducted at the University of Stuttgart between autumn 2017 and spring 2022, within the “Machine Learning and Robotics Lab” and the “Humans to Robots Motion Research Group”, which is funded by the regional *System Mensch* alliance that investigates the *human system*.

Work in this thesis is mainly based on the following four published papers [10, 11, 12, 13] and the paper [14], which is currently under submission. Particularly successful was [11], as it was nominated for the best paper award at the International Conference on Robotics and Automation (ICRA).

Chapter 6 partly contains work from collaborations with undergraduate students whom we supervised in their Bachelors or Masters theses [15, 16, 17] and the following paper collaboration [18].

I found motivation for working in the field of human motion prediction during my master thesis and the collaborations with Ruth Schulz [19, 20, 21]. We conducted HRC user studies with a real robot and compared different interaction styles. The experiments showed the importance of proactive robot behavior, which requires to know what the human will do in the future.

1.3 Thesis Outline

This thesis comprises eight chapters and is structured as follows:

Chapter 1: Introduction motivates this thesis, gives an overview of the research questions, and contains this overview.

Chapter 2: Background introduces relevant background to the techniques used in this thesis, such as, sequence-to-sequence prediction and trajectory optimization.

Chapter 3: Human Motion Forecasting reviews state-of-the-art in full-body human motion prediction, compares approaches on our own dataset and improves on state-of-the-art prediction by changing rotation representations. Moreover, open challenges in motion prediction are identified, which we address in later chapters. This chapter addresses RQ1.

Chapter 4: Controllable Motion Prediction discusses options to constrain the prediction, outputted by a human motion prediction model, to external criteria. This is very useful, for example, to explore multiple possible futures, account for obstacles in environments, or account for possible end position of the hand in order to pick or place an object. In order to do this, we propose a novel framework based on trajectory optimization and additional control inputs for the predictive model. This chapter addresses RQ2.

Chapter 5: Co-optimizing Human-Robot Trajectories explains the challenges arising for concurrent prediction of human motion and planning of robot motion. We think that one of the main challenges is the circular dependency between planning robot motion and predicting human motion. We formulate the problem as Nonlinear Program (NLP) and expand our framework from Chapter 4 to HRC leading to a novel way of formulating and addressing human-robot coordination problems. This chapter addresses RQ3.

Chapter 6: Intent Prediction and Planning for Longer Tasks discusses how human intention can be predicted and presents two approaches, the use of eye-gaze and the use of object affordances. Intention prediction can be useful for long-term prediction, as it helps to infer the human goal. We also discuss techniques for predicting for longer time horizons, such as the use of hierarchical models and Task and Motion Planning (TAMP). Thus, this chapter addresses the research questions RQ4 and RQ5.

Chapter 7: Datasets presents the datasets used in this thesis and explains the ideas of our MoGaze dataset. For data-driven predictive models the data used to train a model is a crucial part and influences the whole prediction capabilities. This chapter addresses RQ6.

Chapter 8: Conclusions summarizes the findings presented in this thesis, reflects on the research questions introduced in section 1.1, and contains an outlook for potential future research directions.

Background

In this chapter we will introduce relevant techniques that are fundamental for the proposed algorithms in this thesis.

2.1 Recurrent Neural Networks

In many domains data is structured as sequences. Examples for such domains are videos, speech, temperature, or human motion. One problem arising in many of those domains is to predict a next sequence, given an input sequence. A problem, which is called sequence-to-sequence prediction and often used in language processing, for example, for generating human-like text as it is done in [22]. A sub-field of sequence-to-sequence prediction is time series forecasting. In time-series forecasting, data is structured in the time domain. Time-series forecasting aims to predict a future sequence given a past observation.

This thesis relates to predictive human models. As human motion is typically given as a trajectory, which is discretized in time, the problem of predicting human motion is a time-series forecasting problem. We are interested in human full-body motion, which is high-dimensional and nonlinear.

A RNN is a neural network that runs iteratively [23] and, thus, is specialized in sequential data, such as time series. Most RNNs are able to work with sequences of variable length. An overview over RNNs is, for example, available in [24].

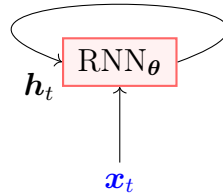


Figure 2.1: Recurrent Neural Network. The input \mathbf{x}_t and the hidden state \mathbf{h}_t are put into the RNN cell with weights $\boldsymbol{\theta}$. Iteratively the next hidden states are computed.

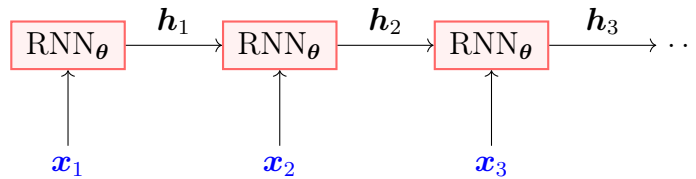


Figure 2.2: Recurrent Neural Network depicted *unrolled*. The recursive loop of the hidden states is removed by unrolling in the time dimension. Here the same network as in Figure 2.1. can be seen.

2.1.1 Recurrent Neural Network with a Recurrent Hidden State

Many RNNs use a recurrent hidden state \mathbf{h} . The idea is that the RNN cell function is iteratively applied to compute hidden states from a sequence of inputs $\mathbf{x}_{0:T}$ of length T . The hidden states can be further connected to other neural network layers, such as fully connected layers, or they can be directly used in a loss function for regression or classification. Note that some RNN architectures have an additional cell output that is different from the hidden state, which will be neglected here for simplification.

The cell function of the RNN computes the next hidden state \mathbf{h}_{t+1} from the previous hidden state \mathbf{h}_t , the input state variable \mathbf{x}_t at the specific timestep t , and the neural network weights $\boldsymbol{\theta}$. Thus, it is given by the following dynamic system equation:

$$\mathbf{h}_{t+1} = f(\mathbf{h}_t, \mathbf{x}_t; \boldsymbol{\theta}) \quad (2.1)$$

The graph for this RNN can be seen in Figure 2.1. The RNN can be applied any number of times, which makes it possible to cope for varying length trajectories.

The RNN can be unrolled, leading to the chained equation for \mathbf{h}_{t+2} :

$$\mathbf{h}_{t+2} = f(f(\mathbf{h}_t, \mathbf{x}_t; \boldsymbol{\theta}), \mathbf{x}_{t+1}, \boldsymbol{\theta}) \quad (2.2)$$

Unrolling for later timesteps works accordingly. The graph for the unrolled RNN can be seen in Figure 2.2.

RNNs use the same set of weights at every timestep. As a consequence, it uses the idea of parameter sharing. This has the advantage that significantly less parameters need to be used than with a fully connected network over the whole time series.

2.1.2 Computing Gradients

To compute gradients through a RNN, the Backpropagation Through Time (BPTT) algorithm is used. For example, suppose we have a loss ℓ applied to the last hidden state:

$$\ell(\mathbf{h}_T, \mathbf{y}; \boldsymbol{\theta}) \quad (2.3)$$

with \mathbf{y} being a ground truth label we compare to.

The gradient of ℓ can be computed by:

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}} = \frac{\partial \ell}{\partial \mathbf{h}_T} \cdot \sum_{t=0}^T \left(\frac{\partial \mathbf{h}_T}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \boldsymbol{\theta}} \right) \quad (2.4)$$

The gradient $\frac{\partial \mathbf{h}_T}{\partial \mathbf{h}_t}$ can be computed by applying chain rule and corresponds to the product of single step gradients as follows:

$$\frac{\partial \mathbf{h}_T}{\partial \mathbf{h}_t} = \prod_{i=t}^T \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} \quad (2.5)$$

A well-known problem in computing gradients for RNNs is the problem of vanishing or exploding gradients [25, 26, 27]. This problem arises from the fact that repeating compositions of the same function are applied at every timestep. Consequently, the magnitude of the gradient converges towards $\|\frac{\partial \mathbf{h}_T}{\partial \mathbf{h}_t}\| \rightarrow 0$ for vanishing gradients, or $\|\frac{\partial \mathbf{h}_T}{\partial \mathbf{h}_t}\| \rightarrow \infty$ for exploding gradients.

2.1.3 Gated Recurrent Neural Networks

To tackle the problem of vanishing or exploding gradients, specialized architectures have been proposed. Gated RNNs introduce so called forget-gates. Those gates enable the cell to learn the ability to delete information from the state. The additional non-linear transformations that are applied to the state at each timestep, improve the properties of the gradients.

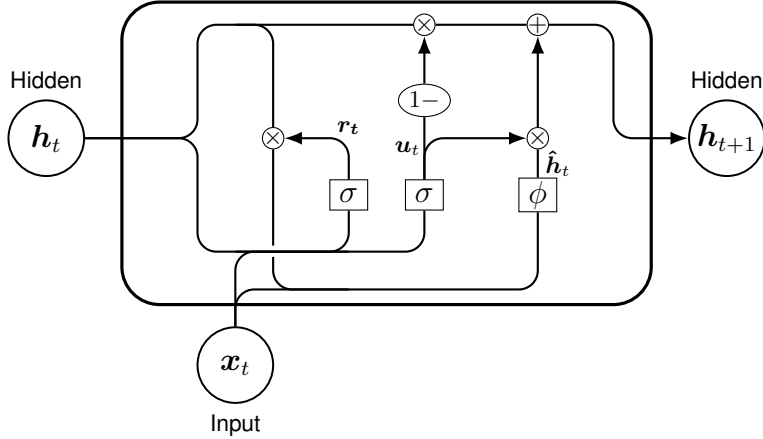


Figure 2.3: Overview of a GRU cell. The hidden state and input at timestep t are connected to several operators computing the output state \mathbf{h}_{t+1} .

An example is the Long Short Term Memory (LSTM) introduced by Hochreiter and Schmidhuber [28]. LSTMs use an input gate, an output gate and a forget gate. It has been shown that using a LSTM information is able to flow to much later timesteps.

Cho et al. introduced a similar architecture called Gated Recurrent Unit (GRU) [29]. It only uses two gates, a reset gate and an update gate. In a study by Chung et al. it was found that GRUs perform similarly to LSTMs [30]. The update rules of a GRU are given by:

$$\mathbf{u}_t = \sigma(\mathbf{b}_u + \mathbf{U}_u \mathbf{x}_t + \mathbf{W}_u \mathbf{h}_t) \quad (2.6)$$

$$\mathbf{r}_t = \sigma(\mathbf{b}_r + \mathbf{U}_r \mathbf{x}_t + \mathbf{W}_r \mathbf{h}_t) \quad (2.7)$$

$$\hat{\mathbf{h}}_t = \phi(\mathbf{W}_h \mathbf{x}_t + \mathbf{U}_h (\mathbf{r}_t \odot \mathbf{h}_t)) \quad (2.8)$$

$$\mathbf{h}_{t+1} = \mathbf{u}_t \odot \hat{\mathbf{h}}_t + (1 - \mathbf{u}_t) \odot \mathbf{h}_t \quad (2.9)$$

with trainable weights $\boldsymbol{\theta} = (\mathbf{b}, \mathbf{b}^u, \mathbf{b}^r, \mathbf{U}, \mathbf{U}^u, \mathbf{U}^r, \mathbf{W}, \mathbf{W}^u, \mathbf{W}^r)$. The update gate is \mathbf{u} and the reset gate is \mathbf{r} . The used activation functions are the sigmoid function σ and the hyperbolic tangent ϕ . A schematic drawing of the GRU cell is shown in Figure 2.3. The reset gates control, which parts of the state are used for calculating the next state candidate vector $\hat{\mathbf{h}}$ and the update gates control, which parts of the states are copied ($u \rightarrow 0$) or replaced with the candidate state vector ($u \rightarrow 1$).

In the GRU architecture the cell output and the hidden state output is the same. It is possible to stack multiple cells together by using the hidden output of the lower

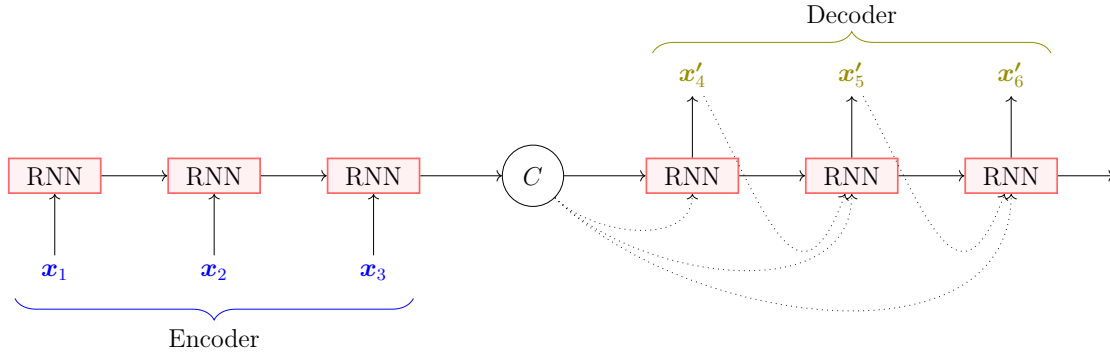


Figure 2.4: Encoder-Decoder RNN for Sequence-to-Sequence Prediction. The encoder outputs a context C , which is used as input to the decoder. Possible but alternative connections are depicted as dotted arrows.

layer as input to the higher layer. Furthermore, it is possible to stack non-RNN layers, such as a fully connected layer, on top of the cell output.

2.1.4 Sequence-to-Sequence Prediction with a Recurrent Neural Network

In 2014, a RNN architecture for supervised learning of a mapping from an input sequence of variable length to an output sequence of variable length was proposed by Cho et al. [31] and independently by Sutskever et al. [32]. This architecture, for example, can be used to forecast a time series with arbitrary length.

The main difference to previous architectures is that, from an input trajectory $\mathbf{x}_{0:k}$, not only one output token \mathbf{h}_k is predicted but instead a whole sequence $\mathbf{x}'_{k:T}$. This sequence can rely on information from all previous timesteps.

An example encoder-decoder structure can be seen in Figure 2.4. An input RNN is used to compute a context C . The context C can be used as input to the next RNN cell. Additionally, it is possible to connect it as inputs to all future cells. The decoder computes outputs \mathbf{x}' , which are compared to ground truth labels \mathbf{y} . It is possible to connect the RNN outputs to the following cell as an input, which allows for time series forecasting in a one-step-ahead prediction fashion.

The sequence-to-sequence RNNs usually are trained to maximize the mean value of $\log p(\mathbf{x}'_{k:T} | \mathbf{x}_{0:k-1})$ over all pairs of the training data set [24].

2.2 Trajectory Optimization

The aim of trajectory optimization is to find a desired behavior of a dynamical system. Typically, this is done by minimizing a performance measure while satisfying constraints.

Trajectory optimization is a local method, leading to an open-loop solution, which means that only one path is computed. In contrast, for example, dynamic programming, is a global method, which aims to find a closed-loop solution. As finding a local solution is sufficient for many problems, trajectory optimization is a useful approach, which is less difficult to compute and scales better to higher dimensions.

Trajectory optimization was initially mostly applied to calculate trajectories in the aerospace domain, such as trajectories for airplanes, rockets and satellites. Nowadays, it is also widely used in the field of robotics, where it is applied to a wide range of problems, including the computation of walking motion for humanoid robots [33] or for manipulation planning of a robot arm [34]. An overview over trajectory optimization is available in surveys by von Stryk and Bulirsch [35], or Betts [36]. More recent tutorials are, for example, by Kelly [37], focusing on direct collocation, or by Toussaint [38], focusing on Newton methods.

2.2.1 System Equations

The dynamics of a system are typically defined as an ordinary equation:

$$\dot{x}(t) = f(x(t), u(t)) \quad (2.10)$$

with states x and controls u being functions of time.

In discrete time the dynamics equation is of the form:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) \quad (2.11)$$

with states and controls being vectors. For each discrete timestep t a specific control vector is applied, leading to a new state \mathbf{x}_{t+1} as specified by f .

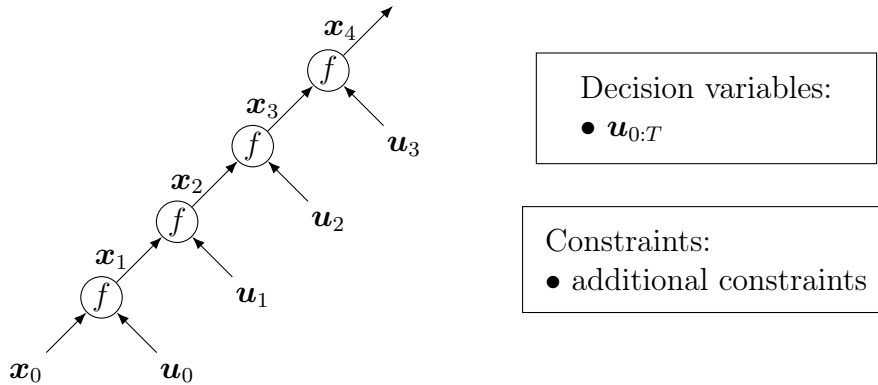


Figure 2.5: Shooting formulation for trajectory optimization. The output of the dynamics equation is directly used as state input at the next time step.

2.2.2 Problem Formulation

A basic formulation of the trajectory optimization problem in continuous time is given by:

$$\min_{u(t)} \int_{t_0}^T c(x(t), u(t)) dt \quad (2.12)$$

subject to: $\dot{x} = f(x, u)$
 $x(0) = x_0^*$
 additional constraints

with $c(x, u)$ being a performance measure, such as minimizing accelerations or torques of the trajectory, and x_0^* being the start state. Additional constraints, for example, can be used to specify a goal state.

A common method to solve the trajectory optimization problem is by converting it into a parameter optimization problem and solve it numerically. This step is called transcription or discretization. In this thesis we use discrete dynamics arising from a RNN and, thus, do not need an explicit discretization step.

2.2.3 Direct Shooting and Direct Collocation Approaches

Direct *collocation* and direct *shooting* are methods that solve a trajectory optimization problem by formulating it as a NLP.

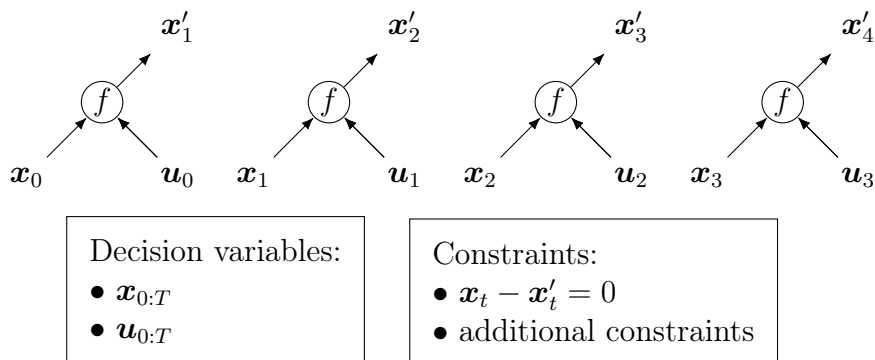


Figure 2.6: Collocation formulation for trajectory optimization. The output of the dynamics equation is not directly used as state input at the next time step, instead additional equality constraints are specified.

The formulation of the shooting method is given by:

$$\min_{\mathbf{u}_{0:T}} \sum_{t_0}^T c(\mathbf{x}_t, \mathbf{u}_t) \quad (2.13)$$

subject to: $x_0 = x_0^*$
additional constraints

The decision variables are only the control variables. The states are computed by using a simulation. Figure 2.5 shows an overview of the shooting formulation. The output of the previous step of the dynamics equation f is used as input for computation of the next state. As a consequence, there is no need for dynamics constraints, as the dynamics function is directly fulfilled through the forward simulation. A property of the shooting approach is that the controls have an influence on all later states, for example, the state x_4 is also influenced by the control u_0 .

In contrast, the formulation of the collocation method is given by:

$$\min_{\mathbf{u}_{0:T}, \mathbf{x}_{0:T}} \sum_{t_0}^T c(\mathbf{x}_t, \mathbf{u}_t) \quad (2.14)$$

subject to: $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$
 $x_0 = x_0^*$
additional constraints

with both: control variables and state variables as decision variables. An overview of the approach can be seen in Figure 2.6. The dynamics function only computes

one step. The states and controls are specified by the optimizer. For collocation, no simulation is required, instead constraints are needed in order to fulfill the dynamics.

While the advantage of the shooting approach is that less constraints and less decision variables are required, problems can arise due to high nonlinearity of the constraints since gradients need to be computed through the forward simulation. The repeated application of the dynamics function can make the gradients less well-conditioned.

A combination of both approaches leads to the multiple shooting method. For multiple shooting, the trajectory is divided into multiple segments, with each of the segments being represented by a simulation.

2.2.4 Constrained Optimization Methods

In the previous subsection we formulated the trajectory optimization problem as NLP. For solving such programs, many methods have been developed in the field of optimization. An overview is, for example, available in [39].

The general form of the optimization problem is given by:

$$\begin{aligned} \min_{\mathbf{x}} c(\mathbf{x}) & \tag{2.15} \\ \text{subject to: } g_i(\mathbf{x}) & \leq 0 \\ h_i(\mathbf{x}) & = 0 \end{aligned}$$

with c being the objective function and g and h being inequality and equality constraints.

There are many numerical methods to solve the problem, for example, interior point methods, augmented Lagrangian methods, and Sequential-Quadratic Programming (SQP).

In our work we use the numerical optimization software ipopt [40], which uses an interior point method. Interior point methods use barrier functions to convert the constrained optimization problem into unconstrained barrier problems:

$$B(x, \mu) = c(x) - \mu \sum_{i=1}^n \ln(g_i(x))$$

with μ being the barrier parameter. With μ converging towards 0, the minimum of the barrier problem B should converge to a solution of the optimization problem

(Equation 2.15). Usually the optimization is started with a parameter $\mu > 0$. The solution is improved by using gradient-based optimization, e.g. by using Newton's method, and the barrier parameter μ is decreased iteratively, till a sufficiently accurate solution is found or the maximum number of iterations is reached.

2.3 Summary

In this chapter we introduced the main techniques underlying the algorithms we propose in this thesis. Namely, RNNs, which will be used as a predictive model for motion prediction in chapters 3-6 and trajectory optimization, which will be used to optimize human motion in Chapter 4 as well as human and robot motion in Chapter 5.

3

Human Motion Forecasting

When human and robot act in close distance, foreseeing how the human will move in the close future, is important for planning safe and efficient robot motion. For example, avoiding regions the human is going to occupy, is a key ability in modern HRC.

In this chapter, we focus on the problem of forecasting pure human motion trajectories, known as the problem of *short-term* motion prediction (RQ1). As we are interested in full-body tasks, such as grasping and placing objects, we mostly focus on predicting full-body motion. In this chapter, we first formally describe the problem and review relevant related work. After that, we review relevant approaches more thoroughly and explain our modifications. Finally, we compare different approaches for motion prediction on real human motion data.

Parts of this chapter are based on the following publications:

- Kratzer, P., Toussaint, M., and Mainprice, J. "Prediction of human full-body movements with motion optimization and recurrent neural networks". International Conference on Robotics and Automation (ICRA) (pp. 1792-1798). © 2020 IEEE
- Kratzer, P., Toussaint, M., and Mainprice, J. "Planning coordinated human-robot motions with neural network full-body prediction models". 2022 (under submission)

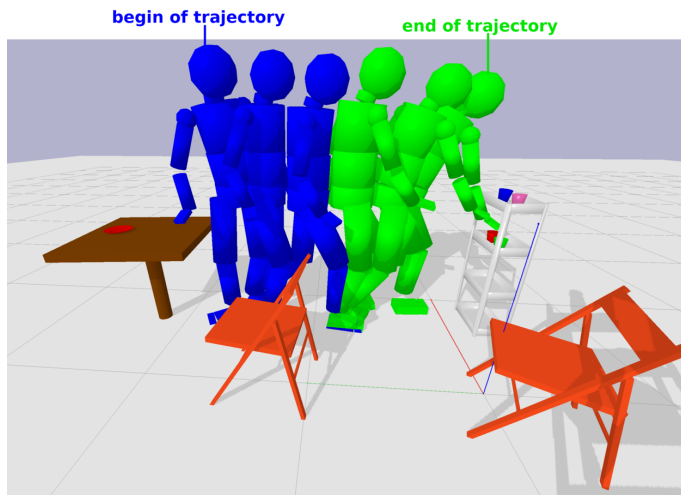


Figure 3.1: The Motion Prediction Problem: Given a human observed trajectory (depicted blue), the goal is to predict future human states (depicted green).

3.1 The Motion Prediction Problem

We discretize time with a small constant timestep Δt between consecutive timesteps.

At a timestep t a human has a state $\mathbf{x}_t^H \in \mathbb{R}^{d_H}$ with dimensionality d_H .

In this chapter, we aim to find a function f^H , for predicting the next states of the human:

$$\mathbf{x}'_{k:T}{}^H = f^H(\mathbf{x}_{0:k-1}^H) \quad (3.1)$$

We are given a trajectory of states $\mathbf{x}_{0:k-1}^H$ of a human H to a timestep $k - 1$. This trajectory has been observed beforehand, is considered to be constant and, thus, can not be influenced. We want to compute a trajectory of future states $\mathbf{x}'_{k:T}{}^H$ starting at the next timestep k and ending at a specified time horizon T . The predicted future states should match the ground truth motion $\mathbf{x}_{k:T}^H$, of what the human will perform in the future as closely as possible. This problem of motion prediction is a sequence-to-sequence prediction problem (see section 2.1).

A related problem is the motion generation problem often considered in computer graphics, in which aesthetic movements are generated. However, motion generation is not necessarily aiming for generating motion that actually occurs. Furthermore, motion generation often aims for generating motion that also look aesthetic in the long-term, while motion prediction techniques usually focus on short-term prediction, which we consider to be in between 0 and 5 seconds. This is usually enough to predict a motion of interest, such as picking or placing an object. For example, placing an

object takes less than three seconds on average [13].

Figure 3.1 shows an example of motion forecasting. The human is in a scenario with multiple objects. Different time frames of the human are visualized as kinematic model (blue and green). The blue part is the observed trajectory of human motion $\mathbf{x}_{0:k-1}^H$, the green part shows the continuation $\mathbf{x}_{k:T}^H$, which we would like to predict. The human is predicted to pick an object from the small shelf.

3.2 Related Work on Human Motion Prediction

Making accurate predictions of human motion is challenging because of its non-linear dynamics, the underlying biomechanical complexity of human motion, the high dimensionality, and the stochastic nature due the influence of a variety of external and internal stimuli.

Human motion comes in many forms ranging from two-dimensional ground-level trajectories to highly detailed motion, such as facial expressions. In close proximity HRC, typical motions include grasping or placing objects, assembly or manipulation of objects, or handover of objects. As a consequence, this thesis mostly focuses on full-body motion, as data of both: arms and legs, are important for these kind of tasks.

For tasks like pedestrian prediction in autonomous driving or navigation, the detailed leg or arm movement information incorporated in full-body motion are not necessarily needed. Therefore, a large fraction of work on motion prediction in these areas focuses on the prediction of ground-level trajectories. A survey on two-dimensional motion prediction is available in [41].

Musculoskeletal Modeling

Human motion is following complex biomechanical mechanisms and, thus, is challenging to model. Recent work on modeling humans in simulations makes use of detailed neuro-musculoskeletal models [42, 43]. Since biomechanical movement dynamics are highly nonlinear and complex, Dri   et al. propose neural networks to obtain a controller for such models [44]. Work on musculoskeletal modeling helps us to deepen our understanding of motion patterns, such as gait, and helps to design personalized treatments in patients with neurological disorders [45]. However, the use as predictive model for human-robot interaction is challenging. Musculoskeletal models usually

need to be adapted to a specific human making it hard to adapt to an unseen human on the fly. Also, external stimuli are challenging to integrate into the simulation.

3.2.1 Data-Driven Motion Prediction

Usually it is possible to collect a dataset of human motion \mathcal{D} that contains motions that are close to the ones that we are going to observe in HRC, for example, datasets with pick and place motions. As a consequence this experimental data can be used to train a predictive model of the human, trying to recreate or maximize the likelihood $p(\mathbf{x}_{k:T}^H | \mathcal{D}, \mathbf{x}_{0:k-1}^H)$.

Inverse Optimal Control

One possibility for predicting human motion using experimental data is Inverse Optimal Control (IOC). IOC aims to find the underlying cost function a human model is following. Berret et al. investigated cost functions for arm movement planning and report that such movements are closely linked to the combination of two costs related to mechanical energy expenditure and joint-level smoothness [46]. Mainprice et al. investigated prediction of human reaching motions in shared workspaces [47]. Using goal-set IOC and iterative replanning, the proposed method accounts for the presence of a moving collaborator and obstacles in the environment using a stochastic trajectory optimizer. While IOC is able to find optimality criteria for many human motions, such as reaching motions, full-body human motion usually consists of many different aspects including cyclic walking motions and acyclic stopping and reaching towards an object. This makes finding a generalizable cost function for the purpose of predicting observed motion challenging.

Graphical Models

Other prior work focuses on training a data-driven model from motion data with early work focusing on graphical models to predict human motion. For instance, Bennewitz et al. modeled human intention using Hidden Markov Models (HMMs) in order to improve navigation behavior of a mobile robot [48]. Kulić et al. used HMMs to model full-body motion primitives and applied it to motion imitation [49]. Elfring et al. used growing HMMs in order to learn human’s goal position from data and use a social forces-based motion model to predict human motion [50]. Lehrmann et

al. propose a non-parametric, nonlinear Markov model [51]. Koppula and Saxena propose a model for predicting trajectories of the human hand using Conditional Random Fields (CRFs) [52]. While these approaches are sound, they generally do not scale to large databases of motion capture or are limited to predict two-dimensional motion of humans and do not deal with the full-body case.

Gaussian Processes

Furthermore, Gaussian Processes (GPs) have been used to model human motion in prior works. Shon et al. learned robotic imitation of human motion using GPs, which they achieved by transforming motion-capture data of the human to a low-dimensional latent space and afterwards transforming it to a high-dimensional robotic state space [53]. Ek et al. use Gaussian Process Latent Variable Models (GPLVMs) for recovering human body pose from silhouettes [54]. Wang et al. introduced GP dynamical models for human motion where they used a low-dimensional latent space and associated dynamics to represent high-dimensional human motion [55]. While GPs are a promising technique to model human motion and their capability of modeling the variance is useful in many applications, they do not scale well to large, high-dimensional datasets.

Neural Networks

As more and larger datasets of human motion are available and advances in neural network techniques have been achieved, deep learning techniques became state-of-the-art for short term motion prediction. The property of neural networks that the query time of a single data point at test time is independent from the size of the training set, is advantageous over many other machine learning methods, such as, GPs. Since RNNs work well on temporal data, such as time series, they are suited well for the task of motion prediction.

In 2015, Fragkiadaki et al. proposed a RNN based model that incorporates nonlinear encoder and decoder networks before and after recurrent layers [56]. Their model is able to handle training across multiple subjects and activity domains. Jain et al. introduced a method to incorporate structural elements into a RNN architecture [57]. Autoencoders can be used within a RNN for denoising the prediction [58].

In 2017, Martinez et al. introduced a GRU based approach [7]. Their methods improved the standard RNN architecture by adding a residual connection to the loop

function of the RNN and showed that this outperforms prior RNN based methods. Gui et al. proposed to use an adversarial approach to predict motion [59]. Pavllo et al. further improved the RNN-based prediction by changing the joint angle representation to quaternions [8, 60]. However, this comes at the cost of additional normalization layers and normalization penalty. Another improvement by Wang and Feng is the Position-Velocity Recurrent Encoder-Decoder Model (PVRED) [61]. The authors propose to augment the state space of the RNN by adding an additional velocity connection as an input to the GRU cell in the recurrent structure.

Recently, motion prediction using graph neural networks [62] or transformers [63] has been shown to slightly improve the prediction performances.

Summary

Due to the fact that neural network based approaches scale well with large datasets and became state-of-the-art for full-body motion prediction, this thesis focuses mostly on neural network based approaches.

3.3 Modeling Human Motion for Neural Network Prediction

Human motion results from complex interactions of the human’s muscular and skeletal systems. While detailed modeling of a human’s musculoskeletal system is possible, the exact arrangement of muscles differs between humans and needs detailed length and weight measurement to fit a model to a human. As a result, musculoskeletal modeling does not easily generalize between different humans.

For the application of neural networks, typically simple models that generalize well to different humans are desired. Often the human body is modeled by using a kinematic model. A human state is given by rotations of the main joints of a human, such as knees, torso or shoulders. Generalization to different humans is done by specifying the distance between the joints.

An even simpler method to model human motion is positional modeling, with using key positions on the human body as state. While positional modeling of a human generalizes well to other humans, a drawback is that non-valid states could be generated easily, since the distance between joints is not fixed. A prediction

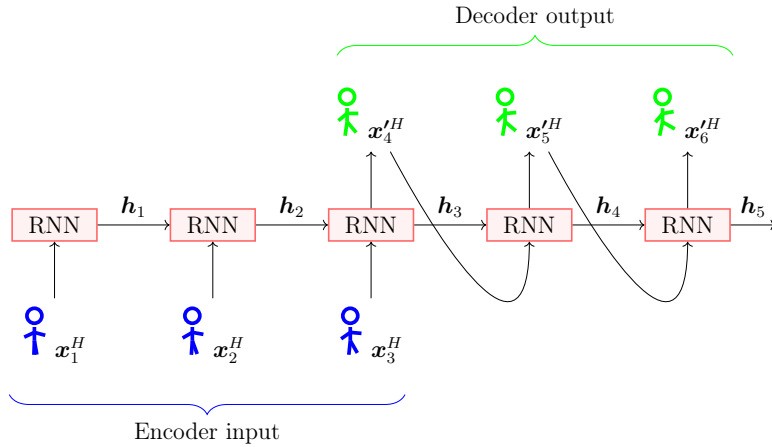


Figure 3.2: Human Motion Prediction with a RNN. The ground truth trajectory (blue) is fed into RNN cells (red), in order to predict a future trajectory (green).

system would need to automatically check that the scalings, e.g. the length of the forearm, are realistic and correct wrong predictions. In contrast, a kinematic state just specifies the angles and the scalings are constant during prediction.

As a result, we will use kinematic states of the human for modeling human motion. We model the kinematic state of a human as a vector consisting of base position, base rotation and joint angles: $\mathbf{x}^H = (p_{\text{base}}, r_{\text{base}}, r_{\text{joints}})$. Thus, the state of the human is positional and does not contain velocities.

3.4 Recurrent Neural Network Approaches

Due to their ability to cope well with high-dimensional time series data, RNNs are a natural choice to predict human motion. In the following subsections we discuss relevant RNN architectures for motion prediction in more detail. Background knowledge about RNNs is described in section 2.1.

3.4.1 Basic Recurrent Neural Network

A typical encoder-decoder RNN structure can be seen in Figure 3.2. The discrete human states that already have been observed, here states $\mathbf{x}_{1:3}^H$, serve as input to the RNN cells of the encoder part of the network. The RNN cell outputs a hidden state \mathbf{h}_t and the next prediction $\mathbf{x}_t'^H$. The hidden states and the observed states are inputs in the encoder part of the network. In the decoder part of the network, the

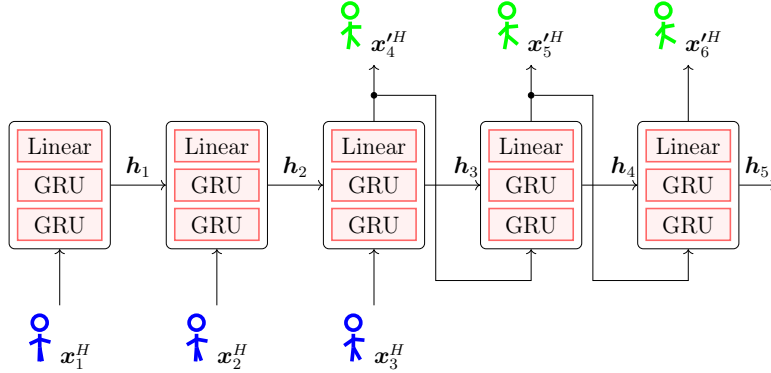


Figure 3.3: Basic RNN for motion prediction with two stacked GRU layers and one linear layer.

hidden state and the prediction output of the previous cell are used as input because no already observed data is available. Consequently, RNNs are able to predict for a varying numbers of time steps. In Figure 3.2, the human states $\mathbf{x}'^H_{4:6}$ are the states of the predicted trajectory of the network.

In order to avoid problems with vanishing gradients, LSTMs or GRUs are used as RNN cell. To achieve better performance, it is possible to stack multiple layers. Figure 3.3 shows an example that stacks two GRU cells. A linear layer is used on top to map to an output vector with size of the human state.

The RNN decoder can be interpreted as a human dynamics function

$$(\mathbf{x}'^H_{t+1}, \mathbf{h}_{t+1}) = f^H((\mathbf{x}'^H_t, \mathbf{h}_t)) = \text{RNNCell}_\theta(\mathbf{x}'^H_t, \mathbf{h}_t) \quad (3.2)$$

where the system state is the human states \mathbf{x}'^H augmented with the hidden states of the neural network \mathbf{h} , and with θ being the weights of the neural network.

The first predicted hidden state is calculated by unrolling the encoder part of the neural network using ground truth:

$$\mathbf{h}_{t+1} = \text{RNNCell}_\theta(\mathbf{x}^H_t, \mathbf{h}_t) \quad (3.3)$$

The hidden state for $t = 0$ is initialized by $\mathbf{h}_0 = \mathbf{0}$.

3.4.2 Residual Connections

A method to improve the generalization performance proposed by Martinez et al. is the use of a residual connection in the loop function [7]. An example architecture

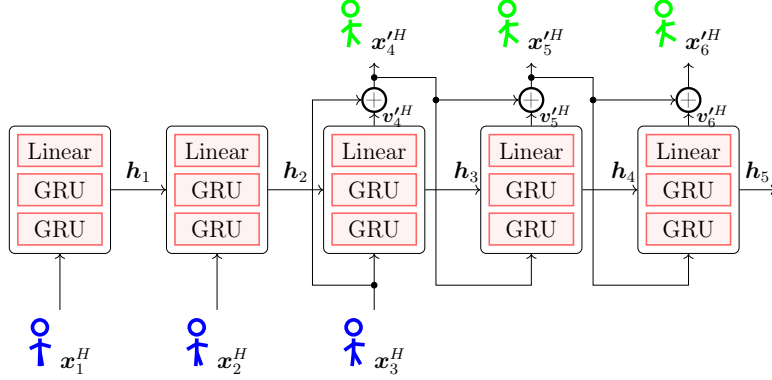


Figure 3.4: Basic RNN for motion prediction with residual connections.

can be seen in Figure 3.4. In the decoder part, the input state is added to the output of the RNN cell, leading to

$$\mathbf{x}'_t{}^H = \mathbf{x}'_{t-1}{}^H + \mathbf{v}'_t{}^H \quad (3.4)$$

with $\mathbf{v}'_t{}^H$ being the cell output.

As a consequence, the output of the RNN cell is trained to be the change from the previous state, instead of the full state of the human. As the human state is positional, this corresponds to predicting velocities. The velocities are independent from the full configuration and as velocities can be applied to several configurations, predicting velocities tends to generalize better. The intuition behind this is that it is harder to model all possible configurations, than to model the change in configuration, which is similar and typically close to zero for all input configurations.

3.4.3 Velocity Connections

A further possibility to improve the RNN with residual connections, is to augment the cell inputs with velocities, which was proposed by Wang and Feng [61].

In order to achieve this, for the ground truth, velocity inputs are computed using finite differences $\mathbf{v}_t^H = \mathbf{x}_{t+1}^H - \mathbf{x}_t^H$. The concatenated state, which consists of pose and velocities, serves as input to a stack of l GRU layers and one linear layer. In our case, we do not feed the position of the base p_{base} into the recurrent unit. This avoids conditioning the model on world positions, which we found leads to better generalization. The stacked layers output the predicted velocities \mathbf{v}'^H , which is achieved by adding the previous configuration to the velocities by using a residual connection $\mathbf{x}'_t{}^H = \mathbf{x}'_{t-1}{}^H + \mathbf{v}'_t{}^H$.

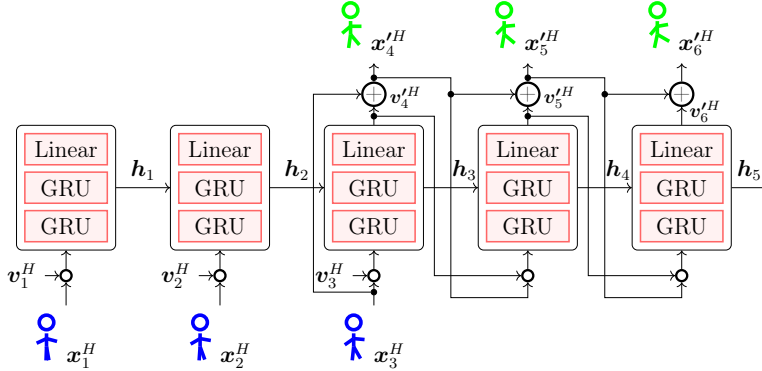


Figure 3.5: Basic RNN for motion prediction with velocity connections.

The full architecture can be seen in Figure 3.5. In the encoder part the velocities for the ground truth are concatenated with the poses and put into the cells. In the decoder loop function, the predicted velocities are directly concatenated with the predicted pose and serve as input for the next cell. The predicted state is also added to the output of the next cell as residual connection.

The function f^H obtained from the RNN with velocity connection is the dynamics function given by:

$$(\mathbf{x}_{t+1}^H, \mathbf{v}_{t+1}^H, \mathbf{h}_{t+1}) = f^H((\mathbf{x}_t^H, \mathbf{v}_t^H, \mathbf{h}_t)) = \text{RNNCell}_\theta(\mathbf{x}_t^H, \mathbf{v}_t^H, \mathbf{h}_t) \quad (3.5)$$

with the state being augmented with velocities \mathbf{v}^H and the hidden state \mathbf{h} of the RNN.

3.4.4 Rotation Representation

For representing three-dimensional rotations multiple options are available. The group of rotations in three-dimensional space is denoted as $\text{SO}(3)$. $\text{SO}(3)$ is a manifold with smoothly differentiable operators and, thus, is a Lie group. To express distances between rotations, different metrics have been proposed in the literature, with many of them being bi-invariant metrics on $\text{SO}(3)$ [64].

Common options for representing rotations in computer graphics and robotics are Euler angles, exponential maps, quaternions, and rotation matrices. Most widely used rotation representations have ambiguities or are discontinuous, which is disadvantageous for neural networks. Furthermore, additional steps like normalization or wrap arounds in the neural network structure lead to additional overhead.

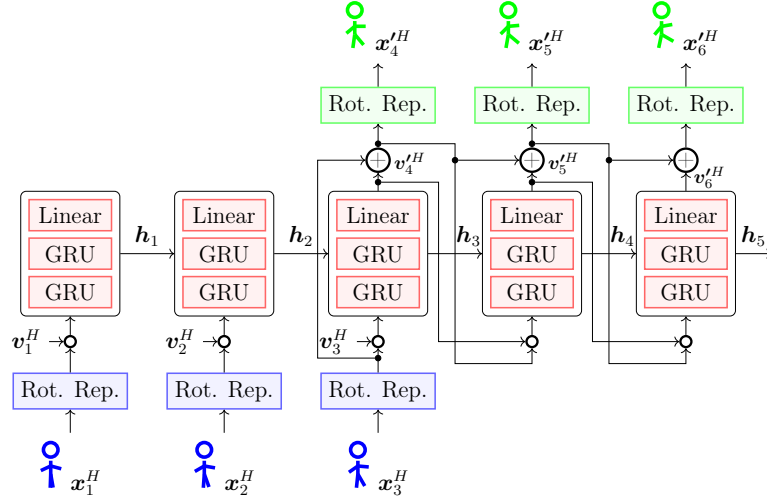


Figure 3.6: Changing the rotation representation. Additional layers before and after the cells allow end-to-end prediction.

A lot of work in human motion prediction, such as [7, 56], makes use of the exponential map rotation representation [65]. Later work showed that using quaternions improves prediction [60].

However, it was found that all representations of four or fewer dimensions are discontinuous in the real Euclidean spaces and a six-dimensional continuous representation was proposed [66]. Thus, we will investigate the use of this six-dimensional representation for the purpose of human motion prediction.

Conversion from and into common rotation representations is achieved by applying mathematical functions, which are differentiable. As a consequence, it is possible to integrate it directly into the network architecture as depicted in Figure 3.6. Thus, it is possible to have the input to the RNN cells in a specific rotation representation but the outputs in another representation that is more suitable for future use, for example, in a loss function or for forward kinematics computations.

3.4.5 Training Procedure

We train the RNN on a human motion dataset \mathcal{D} in order to find suitable weights θ for the RNN. A batch $\mathcal{B} \subset \mathcal{D}$ of fixed-length motion trajectories is sampled from the dataset \mathcal{D} at random. We use data augmentation by using a sliding window of size one to obtain all possible trajectories from the data.

Additionally, we randomize the base rotation r_{base} of the human. We do not feed

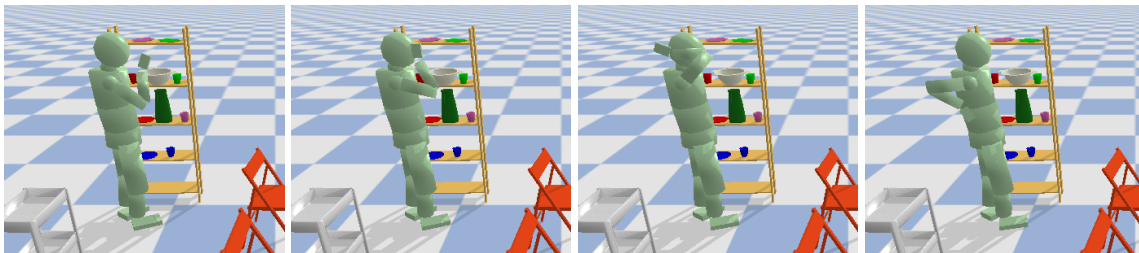


Figure 3.7: Ambiguity issues with exponential maps. Prediction with an exponential map rotation representation after 4, 4.2, 4.4 and 4.6 seconds can be seen. The arm of the human rotates in an impossible way.

the position of the base (i.e., pelvis) into the recurrent unit. This avoids conditioning the model on world positions and leads to better generalization. We found that the same approach does not hold for the base rotation which encodes the direction of motion. Hence we instead offset the rotation randomly during training time to avoid overfitting.

The trajectories are split into two parts, the first part is fed into the RNN encoder, the second part is used as ground truth in the loss computation. The training loss is a sum of translational and rotational components:

$$\ell(\boldsymbol{\theta}) = \|p'_{\text{base}} - p_{\text{base}}\|^2 + \|r' - r\|_1 \quad (3.6)$$

Where we use a mean squared error loss between true and predicted base positions and mean absolute error for the six-dimensional rotation representations r_{joints} and r_{base} .

The weights $\boldsymbol{\theta}$ of the layers are initialized randomly. We use the Adam optimizer [67] with a learning rate of 0.0001 to train the network.

3.5 Empirical Evaluation

In this section the aforementioned methods for motion prediction are empirically evaluated on real motion data.

3.5.1 Rotation Representation

In a first study we compared the exponential map and the six-dimensional rotation representation by training a fixed architecture with both representations. We use our MoGaze dataset for this experiment (see section 7.2). The data is split into

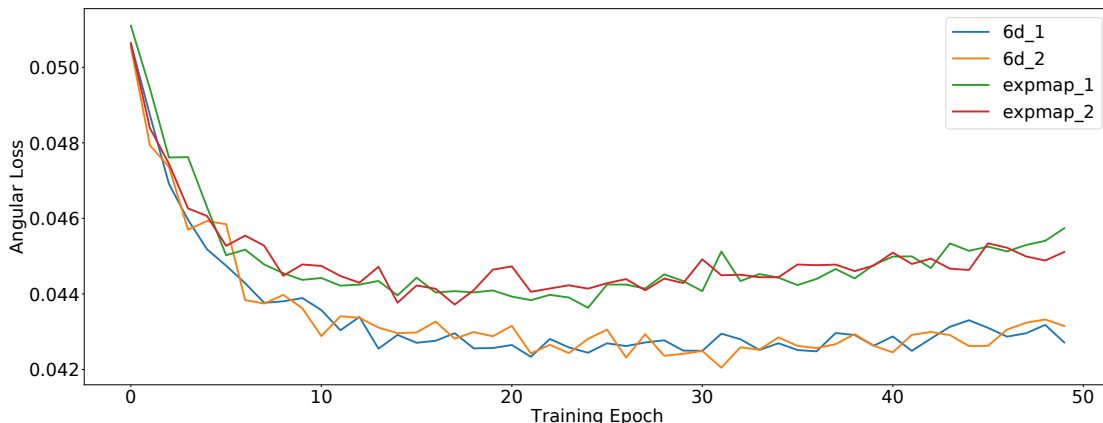


Figure 3.8: Angular test loss for Six-dimensional and Exponential Map Rotation Representations

training and test sets using 80% of each participant for training and 20% for testing. We trained ten models in total, with five randomized weight initializations for each rotation representation.

Figure 3.7 shows an example prediction of an model with exponential map rotation representation. After forecasting for four seconds the arm starts a rotation around the shoulder, which is not possible for the human body. The model probably accumulated to many errors after this relatively long prediction horizon so that it is out of distribution. Due to the 2π singularity of the exponential map it predicts a rotation around the shoulder. We did not find similar issues with the six-dimensional rotation representation.

The training curves for the two best performing models of each representation can be seen in Figure 3.8. We found that all trained models with six-dimensional representation outperform all the models with exponential map representation in terms of angular error. As a consequence, we use the six-dimensional rotation representation in all remaining experiments.

3.5.2 Architectures and Baselines

For evaluating the prediction performance of our human model we compare with the following architectures:

- **zerovel**: The zero-velocity prediction baseline simply predicts no movement. It has been shown to outperform several prediction methods [7].

- **basic-GRU**: A basic RNN with stacked GRU layers as described in subsection 3.4.1. It is similar to the network proposed in [56].
- **residual**: A RNN with stacked GRU layers and a residual connection, so that individual cells output velocities, as described in subsection 3.4.2. It is similar to the network proposed in [7].
- **PVRED**: The method as described in subsection 3.4.3. It is similar to the network proposed in [61].

For all architectures we made changes to the original proposed works to improve the prediction performance such as changing the rotations to the six-dimensional representation and using the according loss. To further ensure fair comparison, we performed the hyperparameter search described in subsection 3.5.3.

3.5.3 Training Sets and Hyperparameter Search

We use our MoGaze dataset for this experiment (see section 7.2). We use the whole data of participant three for testing. From the remaining participants, we do a hyperparameter search by using 80% of each participant for training and 20% for validation. By excluding a whole participant from training and validation, we ensure that the final results reflect behavior for a beforehand unseen human.

During hyperparameter search we change the following parameters:

- Batch size: $|\mathcal{B}| \in (8, 32, 128)$
- Number of GRU layers: $l \in (1, 2, 3)$
- Number of hidden units per layer: $d_h \in (100, 200, 300)$

As weights are initialized randomly and the training behavior depends on that initialization we train each configuration with three different weight initializations.

This leads to 81 trained models per architecture and 243 trained models total. To avoid overfitting, we add dropout and recurrent dropout of 0.2 to the GRU units during training. Each model is trained for 30 epochs, weights are stored after each epoch and the model is evaluated on the validation set. For each trajectory we use one second (20 timeframes) as input to the networks and one second (20 timeframes) as output for loss computation.

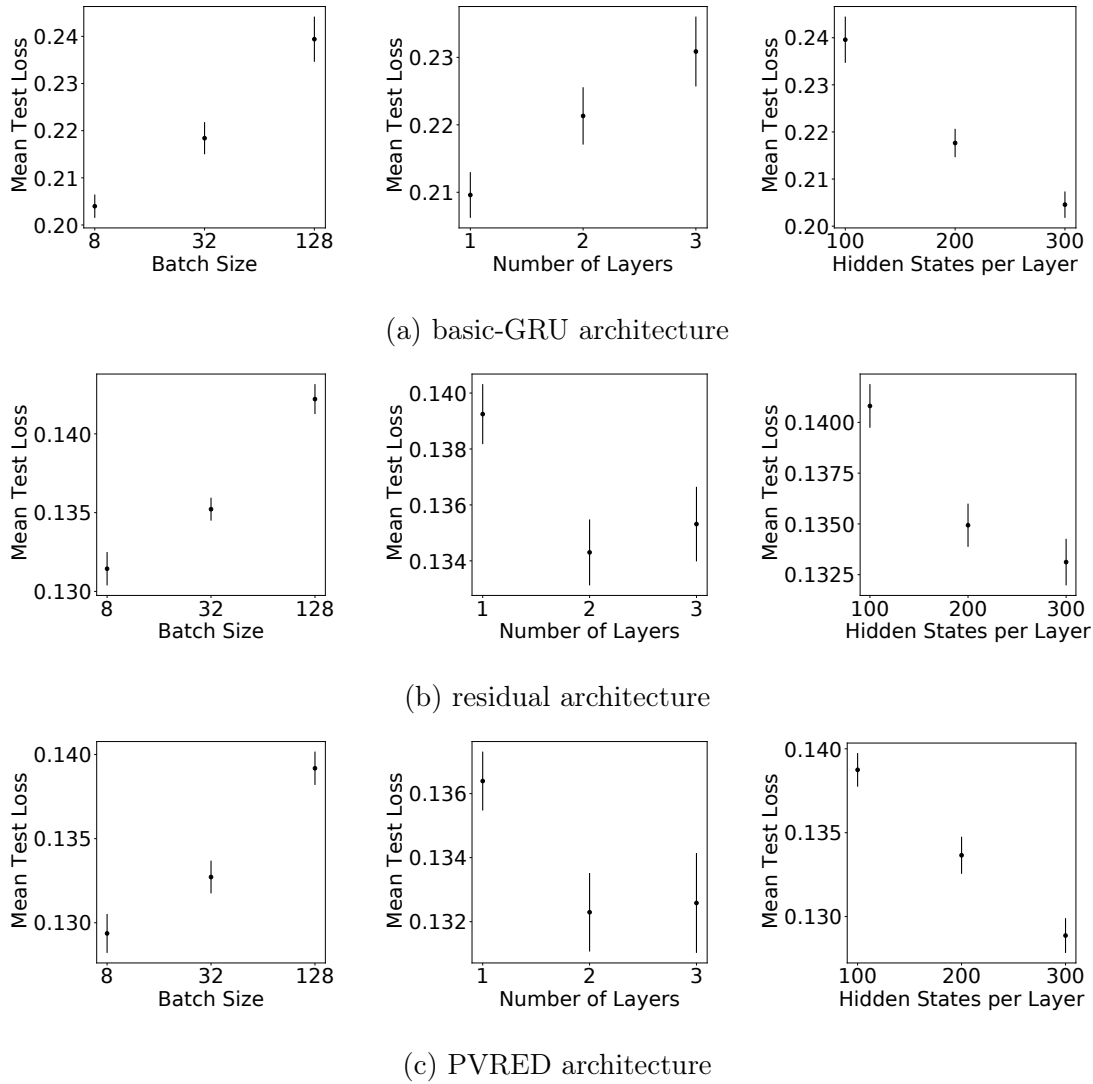


Figure 3.9: Mean of the best test losses for different options during hyperparameter search for the architectures. Error bars show the standard error of the mean.

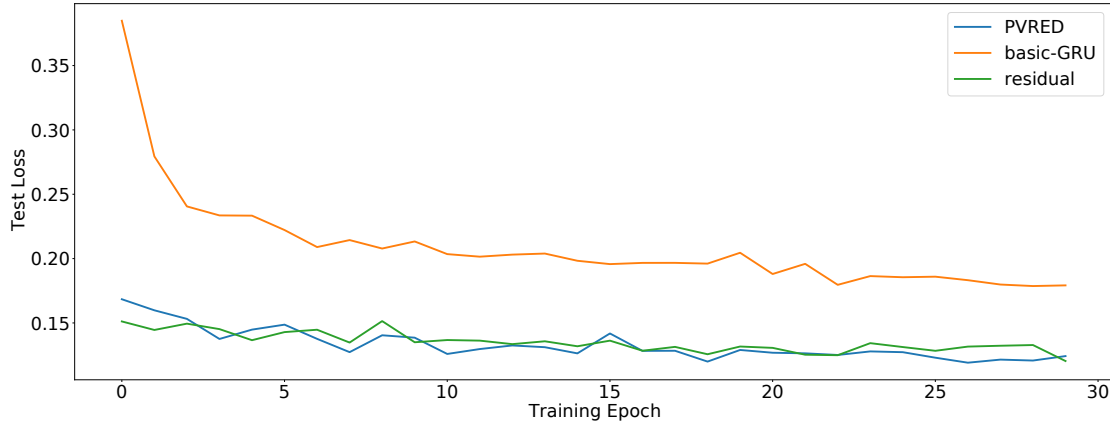


Figure 3.10: Test loss for the best performing model of each architecture during hyperparameter search.

3.5.4 Results of the Hyperparameter Search on the Validation Set

Figure 3.9 shows an overview of the different options for each architecture. We extract the minimal loss for each model on the validation set and take the mean for each option. It can be seen that for all architectures a batch size of $|\mathcal{B}| = 8$ has the lowest test loss. For the *basic-GRU* adding more layers leads to worse performance of the model, while for *residual* and *PVRED* $l = 2$ or $l = 3$ layers seem to work better than only one layer. For $l = 3$ layers the mean test loss is a bit worse but also the trained models show a higher variance in performance on the validation set, especially with the *PVRED* architecture.

The best performing architectures for *residual* and *PVRED* both have a batch size of eight, two layers, and 300 hidden states per layer. The best performing *basic-GRU* also has a batch size of eight and 300 hidden states per layer but only uses one layer.

The test loss for the three best performing architectures can be seen in Figure 3.10. The *residual* and *PVRED* both outperform the *basic-GRU*. The *residual* and *PVRED* have similar performance on the validation set, with the *PVRED* having the best test loss of 0.119 at epoch 26 and the *residual* has a loss of 0.120 at epoch 29. In our following experiments for each architecture the model, which performs best on the validation set, is used.

3.5.5 Results on the Test Set

In order to ensure a fair comparison of the performance of the different architectures, we use the best performing model of an architecture during hyperparameter search and investigate its performance on the test set. Since the test set consists of a beforehand unseen human, this also can give insights about generalization behavior of the methods.

Statistical Experiments

From the test set we extract 3508 trajectories using a sliding window of size 10. We use one second (20 frames) as observed trajectory frames and predict for five seconds (100 frames).

Table 3.1 depicts the mean *base position* metric and the mean *angle* metric at the specified points in time after the prediction started. The *base position* metric of a trajectory is specified by the euclidean distance of the pelvis of the human to the pelvis of the human ground truth and is given in meters. The *angle* metric of a trajectory is specified by the mean relative angle $\theta = 2\arccos(\mathbf{q}_1, \mathbf{q}_2)$ over all joints, with \mathbf{q}_1 being a rotation of a joint of the predicted trajectory in quaternion representation and \mathbf{q}_2 being the corresponding rotation of a joint of the ground truth trajectory in quaternion representation. The value for the *angle* metric is given in radians and ranges from 0 to π .

It can be seen that the *zerovel* baseline outperforms the *basic-GRU* in terms of angular error, which confirms earlier results from [7]. The *residual* method performs better than the *basic-GRU* and the *zerovel* in terms of *angular* error, but has problems on the *base position* metric on longer-term predictions after three and more seconds. The best performing architecture is the PVRED, as it outperforms all the other

Table 3.1: Mean base position metric and angle metric at several timesteps and for the different architectures on the test set

	base pos metric						angle metric					
	0.5	1.0	1.5	2.0	3.0	5.0	0.5	1.0	1.5	2.0	3.0	5.0
zerovel	0.22	0.43	0.61	0.76	0.96	1.0	0.25	0.33	0.37	0.4	0.42	0.44
basic-GRU	0.22	0.28	0.36	0.49	0.74	1.03	0.31	0.36	0.39	0.41	0.44	0.47
residual	0.11	0.22	0.35	0.5	0.79	1.22	0.2	0.27	0.32	0.36	0.41	0.43
PVRED	0.1	0.2	0.31	0.43	0.68	0.98	0.2	0.27	0.31	0.33	0.37	0.41

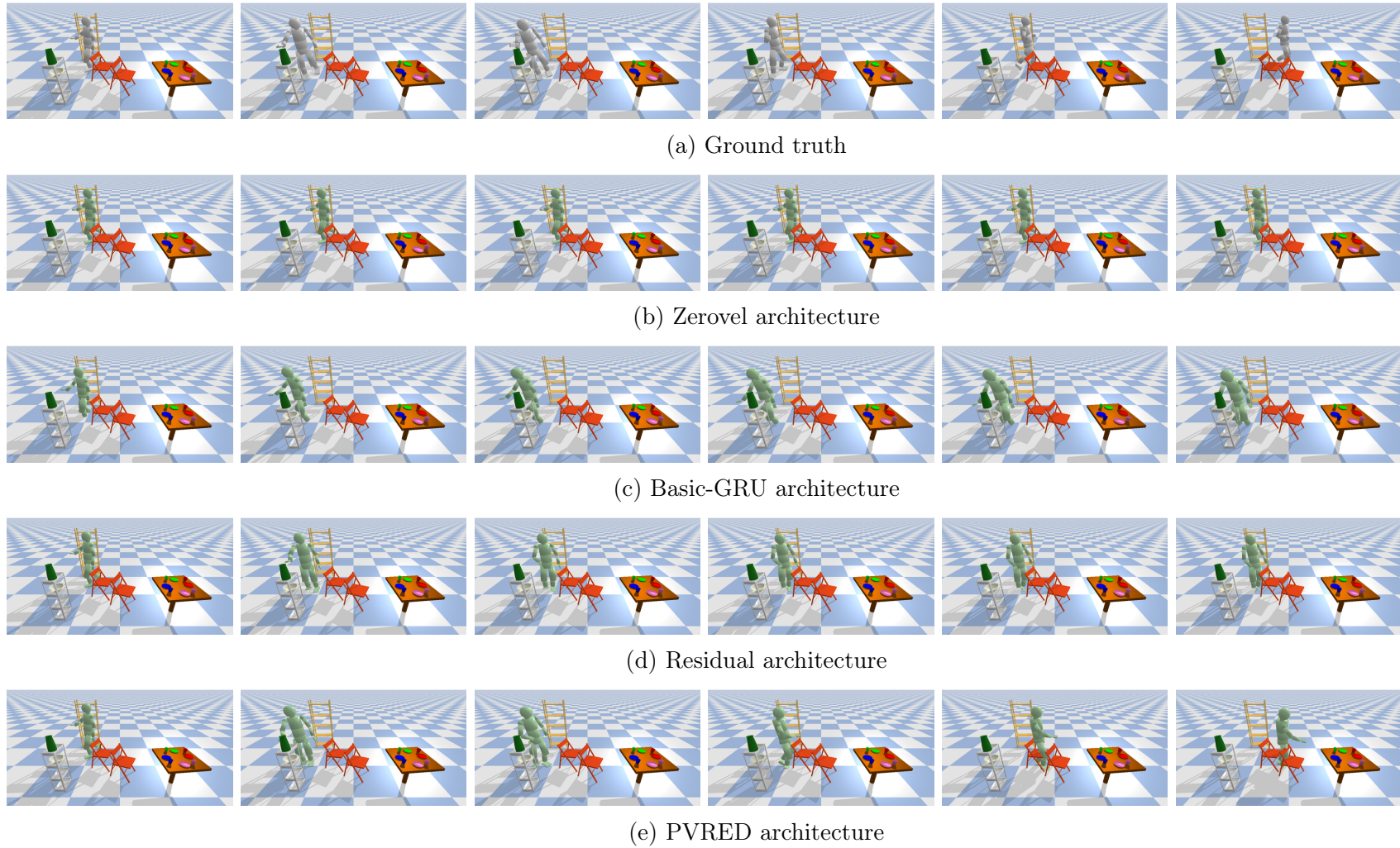


Figure 3.11: Example trajectories of motion forecasting for different methods. From left to right: Trajectory after 0s, 0.75s, 1.5s, 2.25s, 3s and 3.75s.

methods at all timesteps.

Trajectory Example

Figure 3.11 shows example prediction trajectories for the different methods. In the ground truth the human walks towards the small shelf for picking up an object, then rotates and walks towards the table. The *zerovel* baseline predicts that the human does not move, consequently the human is on the same position at all timesteps. In the prediction with the *basic-GRU* architecture, the human is predicted to move towards the small shelf. However, the prediction for leg movement is not accurate and the human is floating. Also, the human is predicted to stay close to the shelf after the shelf is reached. With the *residual* architecture the prediction performs better. The human walks towards the small shelf and then backs up and rotates a bit but almost keeps standing. With the *PVRED* architecture the human moves towards the small shelf, rotates and even walks towards the table. The prediction with the *PVRED* is closest to the ground truth and even works well for longer time horizons.

3.6 Open Challenges in Human Motion Prediction

Motion forecasting on very short time horizons can be done very accurately with the presented methods. However, there remain open challenges, which tend to become more relevant with increasing time horizons. The main challenges we identified are the following:

- Dataset
- Cascading errors of one-step-ahead prediction
- Multi-modality of motion prediction
- Incorporating environment context

One challenge is the availability of suited motion data for training data-driven full-body interaction models. Neural network training usually profits from large datasets. Additionally, a dataset needs to cover typical motions as would occur in the scenario we want to predict motion in.

The issue of cascading errors emerges from the fact that the predicted state is fed back into the neural network prediction. As the predicted state is not totally accurate,

the prediction error is fed back into the RNN cell and thus, the next prediction is based on an inaccurate input state. This error accumulates and increases for long prediction horizons. This issue is not new and, for example, specialized training procedures have been proposed to take it into account during training and improve the RNN prediction performance [68, 69].

Multi-modality arises from the fact that we do not know the exact plan of the human. For example, if the human moves towards a table with many objects, there are multiple possibilities of which object the human wants to grasp. For even longer time predictions, there might be multiple options on where the human walks to after he or she picked up an object.

Incorporating environmental context into the prediction system is desirable. Human motion strongly depends on its environment, as humans avoid obstacles and manipulate objects. For example, Figure 3.11 shows that the prediction of the PVRED method collides with the chair at second three, which is undesired and should be avoided. However, incorporating context from the environment usually is challenging, as the environment can consist of many, including unseen, objects. Those objects can be placed in many ways, leading to an enormous number of possible configurations. This makes using the environment as an input to a neural network motion prediction method hard and would require lots of training data.

An additional challenge is introduced in the context of HRC:

- Adapting to a robot partner

In HRC a human and a robot interact in close proximity. Thus, the human motion is influenced by what the robot is doing, for example, by taking a safety distance in order to avoid collision. Including the robot motion into the human motion forecasting is challenging.

We will address the here mentioned challenges in later chapters: Cascading errors and environment context in Chapter 4, adapting the motion prediction to the robot in Chapter 5, multi-modality in Chapter 6 and the dataset in Chapter 7.

3.7 Summary

In this chapter we focused on pure forecasting of human full-body motion. The problem is to predict a trajectory of human motion given an observed motion

trajectory (RQ1).

We investigated data-driven RNN architectures for the motion prediction problem. On experiments with own motion data we found that a continuous rotation representation outperforms the usually used representations. We compared several architectures. Ensuring a fair comparison by doing a detailed hyperparameter search, we found that the best performing architecture was the PVRED architecture, which was more accurate at all timesteps and still got reasonable human motion after five seconds.

Finally, we identified open challenges of motion forecasting which we tackle in the next chapters.

Controllable Motion Prediction

In this chapter we investigate how we can adapt the human motion prediction *online* in order to fulfill external criteria (RQ2). The ability to change the predictions is desirable, for example, for adapting the motion to the context of the environment, to explore multiple possible continuations, or to constrain it to task knowledge.

Figure 4.1 shows a schematic example of a two-dimensional trajectory. In blue the observed trajectory is shown. Future states are predicted by a motion prediction model and depicted in green. The uncertainty of the predicted state usually increases the further we predict into the future. If we want to find a prediction trajectory that ends up at a specific target state, such as the trajectory depicted in gray, we need a possibility to adapt the prediction. Adapting the prediction leads us to the notion of a controllable behavior model.

In this chapter, we formulate the motion prediction problem as NLP with additional differentiable constraints that can arise from the environments or task knowledge, such as goal constraints or constraints for collision avoidance. We add controls to the RNN architecture for human motion prediction which was introduced in the previous chapter (Chapter 3). As a consequence, it is possible to integrate the new controllable model into the NLP. We demonstrate the approach on real motion data. The experiments confirm the efficacy of the approach and the ability to improve the predictions. For example, predicted trajectories in collision with obstacles are improved to not collide with obstacles after optimizing the trajectory with our method.

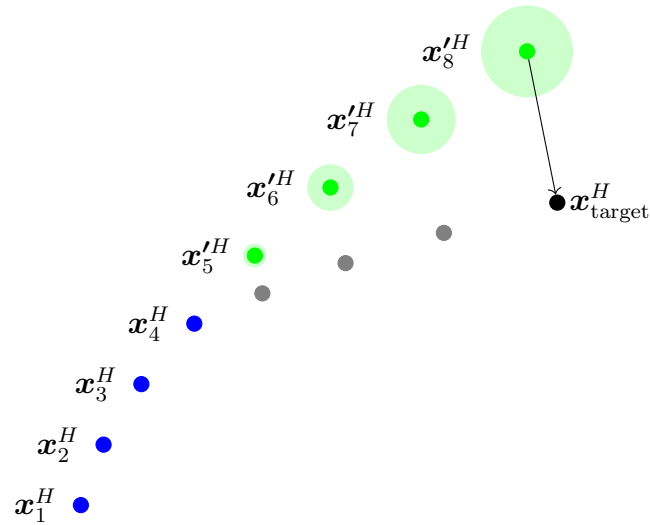


Figure 4.1: Controllable trajectory prediction example. The observed trajectory (blue) is used to predict a future trajectory (green). However, we want that the prediction ends up at a target. This would lead to a new prediction (gray).

Parts of this chapter are based on the following publications:

- Kratzer, P., Toussaint, M., and Mainprice, J. "Towards combining motion optimization and data driven dynamical models for human motion prediction." International Conference on Humanoid Robots (Humanoids) (pp. 202-208). © 2018 IEEE
- Kratzer, P., Toussaint, M., and Mainprice, J. "Prediction of human full-body movements with motion optimization and recurrent neural networks". International Conference on Robotics and Automation (ICRA) (pp. 1792-1798). © 2020 IEEE
- Kratzer, P., Toussaint, M., and Mainprice, J. "Planning coordinated human-robot motions with neural network full-body prediction models". 2022 (under submission)

4.1 Related Work on Human Motion Generation

Controlling a human motion model in order to generate specific motion, for example, for movies, video games, or simulations, is a common task in animating characters in computer graphics.

Graph Structures and Search

Early work on data-driven motion synthesis organizes motion data in graph structures and uses search-based methods to generate motion. Kovar et al. propose to construct a directed graph from a motion database and generate motion by building walks on the graph [70]. Lee et al. identify plausible transitions between motion segments and use efficient search on the resulting graph structure for motion generation [71]. Lau and Kuffner use a planning algorithm on a finite-state machine which defines movement capabilities and compute a sequence of motions to reach a user-designated goal position using global search [72]. Safonova and Hodgins apply optimal search on a motion graph to synthesize motion behaviors [73]. Disadvantages of graph structured approaches are that they can only satisfy constraints that are contained in the motion database and that searching the graph at generation time is necessary.

Sampling-based Approaches

To generate motion, sampling-based approaches have been proposed as well. Coros et al. use a physically-simulated model with a sampling-based planner for generating walking skills [74]. For contact rich motion, Liu et al. propose a sampling based strategy that is able to synthesize plausible motion [75]. A problem with sampling strategies is that noise can be introduced in the synthesized motion due to the random drawing of samples.

Reinforcement Learning

Reinforcement learning approaches can be used for synthesizing motion. Typically the aim is to find a control policy that indicates how the avatar should act in a given situation. For example, Lee and Lee apply dynamic programming to synthesize motion trajectory [76]. Levine et al. use a GPLVM to learn a low-dimensional embedding and use nonparametric approximate dynamic programming to learn a policy on the low-dimensional embedding [77]. More recent work builds on the success of deep reinforcement learning. Peng et al. introduce an actor-critic approach which is able to learn terrain adaptive locomotion skills [78]. In later work, Peng et al. propose DeepMimic, a framework to learn control policies in order to imitate motion-captured motions in a physical simulation [79].

Also, deep learning is an option so that user inputs can serve as direct input to a neural network system. To achieve real-time character control, Holden et al. propose a new neural network mechanism which uses a phase function for cycling the neural network weights according to the walking phase of the human [9].

Trajectory Optimization

Another approach to motion generation is the application of gradient-based optimization techniques. Chai and Hodgins make use of trajectory optimization for generating motion animations with user-defined constraints [80]. The authors compute a statistical dynamic motion model from a motion capture database and apply it to generate an animation that achieves the goal specified by the user. Mordatch et al. use contact-invariant optimization to improve movement trajectories of human behaviors [81] and animating human lower limbs [82].

Summary

Related work on controlling a human model mainly focuses on the *generation* of human motion. However, in our case, we want to control a *prediction* model. In contrast to pure motion synthesis, which aims to generate aesthetic motion, motion prediction focuses on forecasting an observed motion trajectory. Our aim is to find motions that minimally deviate from what a predictive model would output. We think that controlling a model while retaining its predictive capabilities is desirable for motion prediction in order to account for user-defined constraints. Therefore, we use a RNN motion forecasting model as described in Chapter 3 and use gradient-based optimization to change its prediction.

4.2 Formulating the Motion Prediction Problem as Nonlinear Program

The aim in motion prediction is to find a likely trajectory of human motion given the immediate past observation $\mathbf{x}_{0:k-1}^H$. In order to define likely trajectories, we take a data-driven approach and assume that we have a database \mathcal{D} containing typical human motion trajectories of scenarios that we are interested in. Consequently, we define a likely trajectory as a trajectory that has high probability to occur, given the

immediate past observations $\mathbf{x}_{0:k-1}^H$ and the motion data \mathcal{D} , leading us to the human motion trajectory likelihood:

$$\mathbf{x}'_{k:T} \sim p(\mathbf{x}_{k:T}^H | \mathbf{x}_{0:k-1}^H, \mathcal{D}) \quad (4.1)$$

We consider the case, where we are given a controllable model of the human and can formulate a human dynamics function:

$$\mathbf{x}_{t+1}^H = f^H(\mathbf{x}_t^H, \mathbf{u}_t^H) \quad (4.2)$$

with \mathbf{u}_t^H being the controls of a human at timestep t . The dynamics function computes the next state of the human given the previous state and the controls.

When predicting human motion, the aim is to find a trajectory of human states $\mathbf{x}'_{k:T}$ that maximizes the human motion trajectory likelihood and fulfills the human dynamics function at all timesteps. Additionally, we often want to fulfill additional constraints, such as user-defined constraints arising from the environment, from the task, or from multiple conditions designed to explore different modalities.

We can write the motion prediction problem with constraints as the following NLP:

$$\min_{\mathbf{x}_{k:T}^H, \mathbf{u}_{k:T}^H} \alpha^H c^H(\mathbf{x}_{k:T}^H, \mathbf{u}_{k:T}^H) - \log p(\mathbf{x}_{k:T}^H | \mathbf{x}_{0:k-1}^H, \mathcal{D}) \quad (4.3)$$

subject to:

$$\mathbf{x}_{t+1}^H = f^H(\mathbf{x}_t^H, \mathbf{u}_t^H) \quad (4.4)$$

$$h(\mathbf{x}_t^H) = 0 \quad (4.5)$$

$$g(\mathbf{x}_t^H) \leq 0 \quad (4.6)$$

with $c^H(\mathbf{x}_{k:T}^H, \mathbf{u}_{k:T}^H)$ being the cost function associated with the human trajectory and α^H being a hyperparameter to weight its influence. Equation 4.4 shows constraints ensuring the dynamics functions for the human. Additional equality and inequality constraints for the human (Equation 4.5 and Equation 4.6) can be used to ensure environment-dependent and task constraints.

4.2.1 Trajectory Optimization with *Shooting*

We use trajectory optimization to solve the NLP. Background details about trajectory optimization and the *shooting* and *collocation* approaches are available in section 2.2.

One possibility to solve the trajectory optimization problem is a *collocation* approach, which explicitly defines dynamics constraints and has both, the controls and

the states, as part of the decision variables. In our case, the system equations of the human will be specified by a RNN prediction model. This model introduces additional hidden states $\mathbf{h}_{0:T}$ per timestep. As the state would need to be augmented with the hidden states, those would also need to be specified as equality constraints. Consequently, the *collocation* approach would lead to an enormous number of constraints and high memory requirements.

In contrast, the *shooting* approach does not need to specify dynamics as explicit constraints, since it uses a forward simulation. With the future controls $\mathbf{u}_{k:T}^H$ available, the future states $\mathbf{x}_{k:T}^H$ can be computed by forward simulation using the unrolled dynamics \tilde{f}^H , which applies f^H recursively:

$$\begin{aligned}\mathbf{x}_{t+1:T}^H &= \tilde{f}^H(\mathbf{x}_t^H, \mathbf{u}_{t:T}^H) \\ &= (f^H(\mathbf{x}_t^H, \mathbf{u}_t^H), f^H(f^H(\mathbf{x}_t^H, \mathbf{u}_t^H), \mathbf{u}_{t+1}^H), \dots)\end{aligned}\quad (4.7)$$

A consequence of the *shooting* approach is that dynamic constraints are directly fulfilled. The states $\mathbf{x}_{k:T}^H$ are not part of the decision variables, since they are computed during the forward simulation. Only the controls $\mathbf{u}_{k:T}^H$ need to be part of the decision variables.

A disadvantage of this approach is that the controls at a specific timestep \mathbf{u}_t^H have an influence on all states at later timesteps $\mathbf{x}_{k:T}^H$. Consequently, it is required to propagate the gradient through the unrolled dynamics, which will be given by a RNN, for optimization.

4.2.2 Controlled Human Body Dynamics

In the previous Chapter 3, we described how RNNs can be used to predict human motion. While in earlier work we used the architecture with residual connection [83], we apply here the newer PVRED architecture that introduces velocity connections.

The neural network trained on the dataset \mathcal{D} approximates likely trajectories according to the human motion likelihood $p(\mathbf{x}_{k:T}^H | \mathbf{x}_{0:k-1}^H, \mathcal{D})$. With Equation 3.5 we are able to produce a likely prediction for future human states. However, this prediction is uncontrolled and we can not change it when predicting.

In order to change the prediction, we add controls \mathbf{u}^H to the inputs of the decoder RNN cells (see Figure 4.2, depicted orange). We modify both, the input state \mathbf{x}_t^H by adding the controls \mathbf{u}_t^H and the input velocities \mathbf{v}_t^H by adding finite differences of

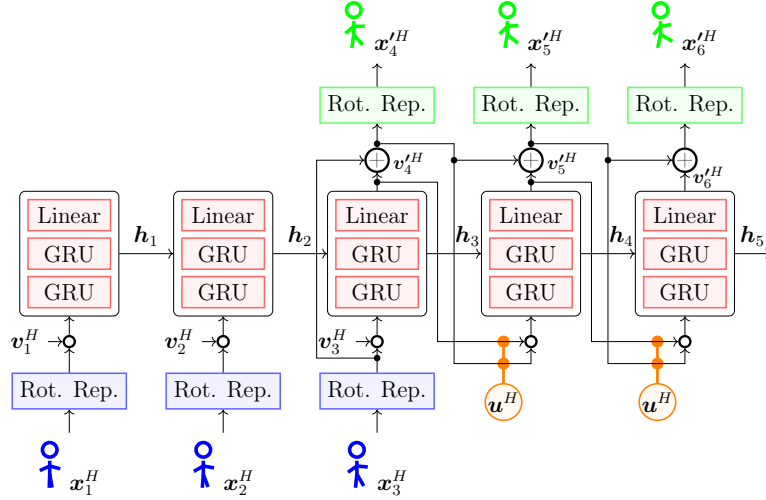


Figure 4.2: Adding Controls to the Recurrent Neural Network

the controls $\mathbf{u}_{t+1}^H - \mathbf{u}_t^H$ at each prediction timestep. This changes the uncontrolled dynamics (Equation 3.5) to the controlled version:

$$\begin{aligned} (\mathbf{x}_{t+1}^H, \mathbf{v}_{t+1}^H, \mathbf{h}_{t+1}) &= f^H((\mathbf{x}_t^H, \mathbf{v}_t^H, \mathbf{h}_t), \mathbf{u}_t^H) \\ &= \text{RNNCell}_\theta(\mathbf{x}_t^H + \mathbf{u}_t^H, \mathbf{v}_t^H + \mathbf{u}_{t+1}^H - \mathbf{u}_t^H, \mathbf{h}_t) \end{aligned} \quad (4.8)$$

As the controls can be viewed as a change of the state, and its finite differences derivatives are used as change of velocities, the control parameters can be viewed as velocity inputs. Interestingly, the control parameters do not directly change the output states. Only the input of the RNN is changed and after that the cell performs a prediction step grounded on the changed state. The fact that the output states are only predicted by the RNN and not changed directly, reduces the probability that the output is shifted outside of the data distribution.

Using the controls, we can now change the state and velocity input at every predicted timestep by using the velocity control parameters \mathbf{u}^H . Changing the predicted input states to the cells in the RNN decoder makes sense, since the predicted states only predict the mean and consequently will have a prediction error. Using \mathbf{u}^H to shift the RNN inputs slightly, allows us to improve the prediction error of the neural network by taking user-specified constraints into account.

We introduce the assumption that the human likelihood is integrated into the human dynamics function. We assume that when we solve for optimal controls with a cost that minimizes the magnitude of changes applied to the state, we also find likely trajectories of human motion. As the RNN is trained to predict likely trajectories, a

key to make the assumption true is to ensure that the predictions at every timestep are only changed slightly. Because the network just predicts a likely state and is inaccurate, small changes are within the prediction inaccuracy.

Our assumption implies that the likelihood of the trajectory increases with decreasing the changes made by the controls to the input states of the RNN decoder. Thus, decreasing the change approximately maximizes the human trajectory likelihood. We decrease the change by minimizing the controls $\|\mathbf{u}^H\|$ which we integrate in our optimization objective (see subsection 4.4.1). We will empirically test the assumption by changing real motion data and by comparing it to the ground truth.

We found that small changes along the trajectory can lead to a very different trajectory, since the prediction at later states changes when changing the predictions at earlier timesteps, and changes can accumulate over the trajectory. For example, when a walking trajectory prediction is slightly rotated to one side in the beginning, the network predictions for later timesteps will adapt and predict a trajectory where the human will walk towards that side. While the total changes made to input states of the trajectory can be very small, the human can end up at a totally different location after predicting for a defined period of time. This allows to find likely trajectories for varying future scenarios.

4.3 Solving the Nonlinear Program

Using the shooting method and the assumption that the human likelihood is integrated into the human dynamics function, the optimization problem simplifies to:

$$\min_{\mathbf{u}_{k:T}^H} c^H(\mathbf{x}_{k:T}^H, \mathbf{u}_{k:T}^H) \quad (4.9)$$

subject to:

$$h(\mathbf{x}^H) = 0 \quad (4.10)$$

$$g(\mathbf{x}^H) \leq 0 \quad (4.11)$$

where c are the costs for the human trajectory and h and g are user defined equality and inequality constraints. Note that the constraints can act on any state \mathbf{x}_t^H at any prediction timestep t . Thus, the unrolled dynamics \tilde{f}^H of the neural network and its derivatives are required.

In contrast to Equation 4.3, Equation 4.9 only optimizes over the controls \mathbf{u}^H and not the states \mathbf{x}^H , since the states are directly computed by the unrolled dynamics \tilde{f}^H .

As the dynamics include the human trajectory likelihood, the optimization objective only consists of the the human trajectory cost function c^H . Another difference is that the dynamics do not appear as individual constraints, as due to the state computation by the unrolled dynamics function \tilde{f}^H , they are fulfilled implicitly.

4.3.1 Primal-Dual Interior Point Method for Nonlinear Programming

Gradient-based methods can be used to optimize the problem with, for example, specifying constraints using barrier functions or Lagrange multipliers. We use an interior-point gradient-based solver (ipopt [40]) with linear solver MA57 [84] for optimizing the NLP.

Ipoint computes solutions to the barrier problems (see subsection 2.2.4), and then decreases the barrier parameters μ to drive them towards 0. As we do not provide the analytic Hessian, ipopt is approximating the Hessian using a Broyden-Fletcher-Goldfarb-Shanno (BFGS) update. While it would be possible to calculate analytic Hessians and analytic Hessians could lead to an improvement in the optimization behavior, computing the Hessian through the unrolled neural network is costly and, at the time of writing, most programming libraries for neural networks do not provide efficient methods for computing the Hessian.

We use the tensorflow library [85] to formulate the constraints end-to-end, directly in the network architecture (see Figure 4.3) which allows us to use the automatic differentiation functionality to obtain the gradients.

Algorithm 4.1 shows a high-level overview over the optimization procedure. The input to the algorithm is the observed human trajectory $\mathbf{x}_{0:k-1}^H$. The unrolled dynamics function \tilde{f}^H , which is based on a RNN for predicting the human motion, computes the prediction states using the controls \mathbf{u}^H and the start state \mathbf{x}_{k-1}^H . The forward functions for the optimization objective and constraints are computed by these states and the controls. Gradients are computed using the tensorflow library. The ipopt library uses the gradients and forward computations to compute logbarrier parameters, approximate the Hessian and compute the search direction. These steps are repeated till convergence or a maximum number of iterations is reached. Finally, the algorithm outputs a prediction for the states $\mathbf{x}_{k:T}^H$ and the controls $\mathbf{u}_{k:T}^H$ for the human.

Algorithm 4.1 Human Motion Prediction with Trajectory Optimization

Input: Observed trajectory $\mathbf{x}_{0:k-1}^H$
Init: $\mathbf{u}_{k:T}^H \leftarrow \mathbf{0}$
for $n < \text{maxiter}$ **do**
 $\tilde{\mathbf{x}}_{k:T}^H \leftarrow \tilde{f}^H(\mathbf{x}_{k-1}^H, \mathbf{u}_{k:T}^H)$
 Compute objectives c^H and constraints h_i, g_i
 Compute gradients $\frac{\partial c^H}{\partial \mathbf{u}^H}, \frac{\partial h_i}{\partial \mathbf{u}^H}, \frac{\partial g_i}{\partial \mathbf{u}^H}$ (tensorflow)
 Compute logbarrier parameters (ipopt)
 Approximate Hessian with BFGS (ipopt)
 Compute search direction and step size (ipopt)
 Update controls $\mathbf{u}_{k:T}^H$
end for
return $\mathbf{x}_{k:T}^H, \mathbf{u}_{k:T}^H$

4.4 Motion Objectives and Constraints

In this section, we describe the objective function and show examples of motion constraints, which we found to work well in our experiments.

4.4.1 Optimization Objective

As optimization objective we use the following trajectory costs of the predicted human trajectory, which penalizes the magnitude of the control terms:

$$c^H = \|\mathbf{u}^H\|_{\mathbf{A}}^2 \quad (4.12)$$

where \mathbf{A} is a norm which approximates $c^H \approx \sum_t^T \|\dot{\mathbf{u}}_t^H\|^2$ by finite difference:

$$A = \begin{bmatrix} 1 & -1 & 0 & \dots & 0 & 0 & 0 \\ -1 & 2 & -1 & \dots & 0 & 0 & 0 \\ 0 & -1 & 2 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 2 & -1 & 0 \\ 0 & 0 & 0 & \dots & -1 & 2 & -1 \\ 0 & 0 & 0 & \dots & 0 & -1 & 1 \end{bmatrix}. \quad (4.13)$$

Since the neural network is trained to find a likely prediction of human motion and thus approximates $p(\mathbf{x}_{k:T}^H | \mathbf{x}_{0:k-1}^H, \mathcal{D})$, minimizing $c^H = \|\mathbf{u}^H\|_{\mathbf{A}}^2$ shifts the neural network cell inputs closer to the inputs of the initial prediction and increases the likelihood. As the neural network is trained to maximize the likelihood of future states $p(\mathbf{x}'_{k:T} | \mathbf{x}_{0:k-1}^H)$ on the dataset, we assume that minimizing c^H leads to a similar effect as maximizing the likelihood of future states. Note that setting $\mathbf{u}^H = \mathbf{0}$ leads to the initial prediction of the neural network, which we use to warm start the optimizer.

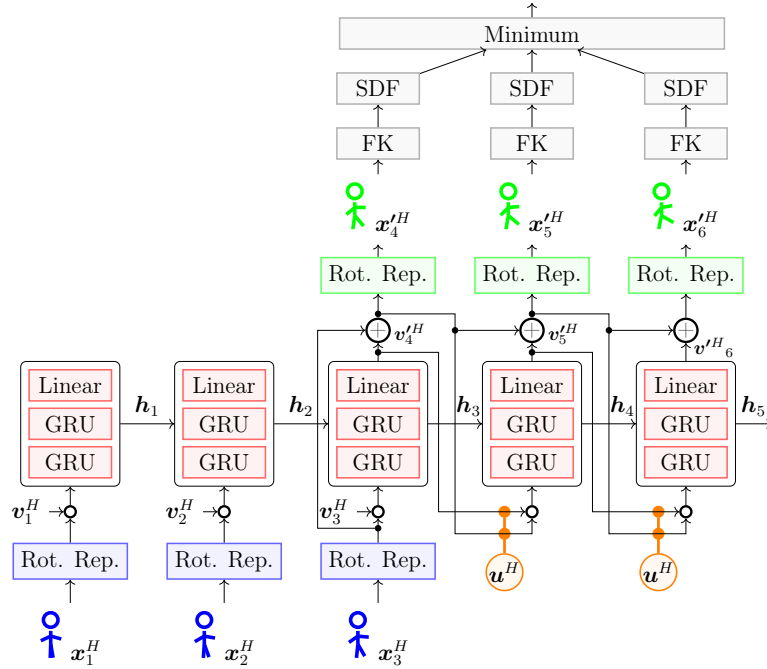


Figure 4.3: Incorporating Constraints into the RNN

4.4.2 User-Defined Constraints

For constraints, we focus on goal and collision constraints which we think are most important in the context of pick and place tasks and, therefore, essential when interacting with a robot. Additional constraints that could improve the human prediction, for example, are constraints on foot contact, joint limits, and constraints on kinematic integrity, such as self collisions.

In the following, constraints h are equality constraints, which means that we achieve $h = 0$ after optimization, and constraints g are inequality constraints, which means that we want to have $g \leq 0$ after optimization. The differentiable kinematics function ϕ^H is used to calculate positions or orientations attached to a specific link of the human.

Goal Constraints

Often we want to predict trajectories where the human moves to a specific position or needs to pick up a specific object. Thus, we formulate a goal constraint as equality constraint with a specific link, such as hand or base, ending up at a specific goal

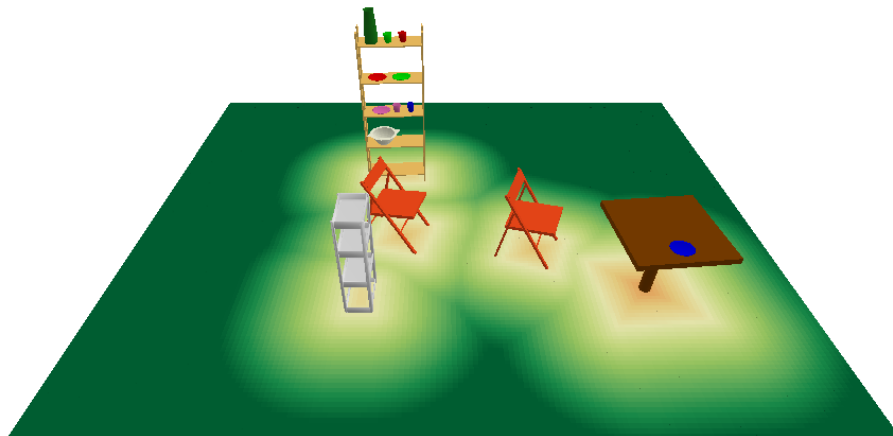


Figure 4.4: Visualization of a two-dimensional SDF on the MoGaze Dataset.

position p^* at timestep t :

$$h_t^{\text{Goal}} = \|(\phi_{\text{linkpos}}^H \circ \tilde{f}_t^H)(\mathbf{x}_k^H, \mathbf{u}_{k:t}^H) - p^*\|^2 \quad (4.14)$$

where $\phi_{\text{linkpos}}^H(\mathbf{x}^H)$ is the Forward Kinematics (FK) of a given link, such as the human hand.

Collision Constraints

In order to avoid collision with obstacles in the environment at timestep t , we use a Signed Distance Field (SDF). A SDF is a function that for a queried point gives its distance to the closest point on an object boundary or, if the point is inside an object, the negative distance to the closest point on the object boundary. An example of a two-dimensional distance field for the MoGaze dataset can be seen in Figure 4.4. Obstacles, such as chairs, shelves and the table, are projected to the two-dimensional ground plane.

Using the SDF, we formulate the collision constraint as follows:

$$g_t^{\text{Collision}} = (\text{SDF} \circ \phi_{\text{linkpos}}^H \circ \tilde{f}_t^H)(\mathbf{x}_k^H, \mathbf{u}_{k:t}^H) \quad (4.15)$$

where $\phi_{\text{linkpos}}^H(\mathbf{x}^H)$ is the FK of a given link of the human. For the two-dimensional SDF that avoids collision with the environment, for example, using the human pelvis makes sense.

For avoiding collision at all timesteps, one constraint for every timestep can be defined. However, to reduce computation time, we use a single constraint using a

minimum function as follows:

$$g^{\text{Collision}} = \min(g_k^{\text{Collision}}, g_{k+1}^{\text{Collision}}, \dots, g_T^{\text{Collision}}) \quad (4.16)$$

We found that both approaches work reasonably well. In our implementation we specify the collision constraint as a single constraint because it allows us to compute all needed g_t while unrolling \tilde{f}^H and using a single automatic differentiation call to compute the gradient. Note that to improve the optimization behavior, it may make sense to use a smooth approximation of the minimum function, such as a RealSoftMax (a.k.a. LogSumExp).

Figure 4.3 shows the integration of the collision constraint into the RNN architecture depicted in gray. A predicted state of the human (depicted green) serves as input to a FK layer, which maps the kinematic state to the specified link position. This link position is connected to a SDF layer which outputs a signed distance. The distances of all timesteps are concatenated and the minimum is taken. The final output is the negative distance of the link that is most far inside an object, or the distance of the link that is closest to an object. As a consequence, the constraint always acts on the most important position and during optimization, the controls are adapted until all positions are out of collision.

4.5 Empirical Evaluation

In this section we evaluate our method for controlling a predictive human model using trajectory optimization on motion data.

4.5.1 Goal Constraint Experiments with One Actor

In our first experiment, we evaluate whether the prediction of the PVRED architecture can be improved by our prediction method when we already know the target position of the reaching motion for the human hand.

We train our model on data with one actor: the *walking* data, the *reaching1* data and the *reaching2* data (see section A.1.1 and section A.1.1). A fraction of 10% from every dataset is held out for testing purposes.

From the *reaching1* test set, we extract 25 reaching trajectories with a length of one second. We compare our method with three baselines:

- **zerovel** predicts the same state for all future steps.
- **interp** interpolates the hand position of the human. While it is informed with the goal, only the hand position is predicted and not the full state.
- **initial** just queries the PVRED network without optimizing the predicted trajectory.

We compute the distance of key joints of the human (wrists, elbows, knees, ankles and pelvis) and compute the mean distance of the predictions to the ground truth.

In our method, we place a goal position constraint and optimize the prediction to end up at the goal position. Thus, our method is informed with details about the goal that the *zerovel* and *initial* baselines do not need, and we expect that our method takes advantage of these details to improve the prediction.

Table 4.1 (b) shows the sum of the mean distances of the nine key joints. It can be seen that our method, by using trajectory optimization with a goal constraint, significantly improves the prediction. The prediction is not only improved at the last time step, where the goal constraint is placed, but along the whole trajectory. The optimization outperforms the baselines at all timesteps.

In Table 4.1 (w) we only compute the distance of the wrist to the ground truth. We also compute the *interp* baseline between the start position of the wrist and the target position. The *interp* baseline and our method are informed with the goal state and thus able to get zero error in the last time step for the baseline and almost zero error for our method. Our method, where the RNN implicitly reconstructs the

Table 4.1: Error of state prediction on different time steps in the future for the whole body (b) and the right wrist only (w). Reported values are in meters. For the whole body the sum distance of nine key joints is shown.

	ms	125	250	375	500	625	750	875	1000
Zerovel (b)		0.72	1.37	1.80	2.31	2.72	3.01	3.15	3.25
initial (b)		0.20	0.36	0.45	0.57	0.68	0.78	0.86	0.94
ours (b)		0.20	0.35	0.44	0.53	0.56	0.59	0.62	0.64
Zerovel (w)		0.14	0.28	0.37	0.48	0.56	0.61	0.62	0.62
initial (w)		0.03	0.07	0.09	0.11	0.13	0.14	0.15	0.15
ours (w)		0.03	0.07	0.08	0.09	0.08	0.06	0.04	0.01
Interp (w)		0.05	0.11	0.15	0.17	0.17	0.13	0.08	0.00

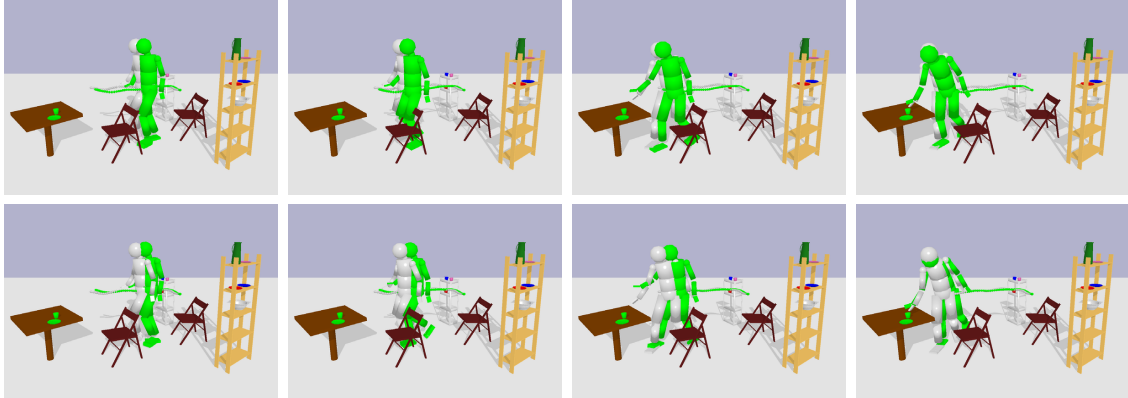


Figure 4.5: Human trajectory of reaching towards an object on the table for two seconds. The prediction by our method is shown in green, baseline of human motion in gray. From left to right we show the prediction after 1s, 1.3s, 1.6s and 2s. The top row shows the prediction without obstacle constraint, the bottom row with obstacle constraint. In the prediction without obstacle constraint the human collides with the chair.

underlying human dynamics, outperforms the interpolation baseline on all other time steps.

These results show that our method is able to reconstruct the full-body trajectory of the human given a target goal location of a joint, which is useful for scenarios where possible target locations of the human are already known, either obtained by a higher level prediction mechanism or through scene understanding. Mechanisms to predict the intention of the human will be discussed in Chapter 6.

A limitation of the approach with goal set constraint is that we have to predefine the duration of the trajectory which is a known problem in the trajectory optimization literature [34]. Simple approaches to solve this issues include optimizing trajectories of different time horizon or reoptimizing the trajectory with a longer horizon if the target position is not reached. A more general approach would dynamically add and remove samples during optimization, or add the time t at which the goal constraint should be fulfilled to the decision variables of the optimizer.

4.5.2 Collision Constraint Experiments with One Actor

In this experiment we evaluate whether the obstacle potential helps to further improve the prediction. Therefore, we extract 22 reaching trajectories from the *reaching2* test set with a length of two seconds. Some of the trajectories contain motion trajectories

that are close to an obstacle (chair). We compute the SDF of the scene and add an obstacle objective in addition to the goal objective.

Figure 4.5 shows an example reaching trajectory with the ground truth (gray) and our prediction (green). The top row shows the prediction without obstacle objective. There the human prediction collides with the chair because the RNN has no information about obstacles in the scene. Activating the obstacle objective in the trajectory optimization step penalizes collisions and forces the prediction around the object. In the second row, the improved prediction with obstacle avoidance is shown. The prediction no longer collides with the chair.

We performed a quantitative comparison averaged over the 22 extracted reaching motions (see Table 4.2). The table shows the mean full body error of the key joints compared to the ground truth. Note that only a few of the trajectories contain motion trajectories that are close to the obstacle. Nevertheless, the method with additional obstacle objective outperforms the other prediction methods at all timesteps larger than 500 milliseconds and achieves similar performance at 250 milliseconds.

Our results demonstrate that a collision constraint can improve the prediction of a neural network forecasting model without retraining the network. As especially collision avoidance is challenging to directly integrate into the neural network architecture, our method is promising to enhance prediction performances.

4.5.3 Goal Constraint Experiments on MoGaze

While the previous experiments considered only one actor, we now compare our method with several baselines on the MoGaze dataset. We use the best performing models from our hyperparameter search (see subsection 3.5.3) and use the test set with an unseen participant for the experiments. Thus, the results reflect the

Table 4.2: Error of state prediction on different time steps in the future for the whole body. Our method with goal objective (g) and goal and obstacle objectives (g+o) is shown.

ms	250	500	750	1000	1250	1500	1750	2000
Zerovel	1.21	2.45	3.90	5.50	7.53	9.49	11.15	11.74
PVRED	0.70	1.25	1.79	2.48	3.48	4.56	5.70	6.39
ours g	0.68	1.25	1.79	2.27	2.57	2.52	2.24	2.18
ours g+o	0.69	1.25	1.74	2.18	2.45	2.41	2.15	2.10

behavior on a beforehand unseen human. We compare the following architectures: zerovel, basic-GRU, residual and PVRED. For each architecture, we additionally have following options for comparison:

- **initial (i)**: We use the initial (blackbox) prediction outputted by the corresponding architecture only.
- **sample (s)**: We sample 100 human predictions by adding Gaussian noise to the encoder hidden state \mathbf{h}_t before inputting it to the decoder. The predictions are ranked based on the distance to the goal. The trajectory with the closest distance to the goal is taken for comparison (see Algorithm 4.2).
- **optim (o)**: We optimize the prediction using our framework. For *basic-GRU* and *residual*, only the state input to the RNN is changed, as there are no velocity inputs. For *zerovel* we add controls directly to the joint states, leading to a velocity-optimized kinematic solution.

The method *PVRED-optim* is our full proposed method.

Our hypothesis is that using our framework, the initial prediction by a method can be significantly improved, when an oracle goal constraint is added to the hand of the human at the last timestep. Ideally, our method should improve the predictions not only at the last timestep, where the goal constraint is placed, but already on earlier timesteps.

To test our hypothesis, we extract 192 reaching motions from the test data set. We use trajectories of a duration of three seconds and use one second as input to

Algorithm 4.2 Sample Baseline for Goal Prediction

Input: Observed trajectory $\mathbf{x}_{0:k-1}^H$, variance σ^2
 $\mathbf{h}_k \leftarrow \text{ENCODER}(\mathbf{x}_{0:k-1}^H)$
for $i \in \{0, 1, 2, \dots, 100\}$ **do**
 $\mathbf{h}_k' \leftarrow \mathbf{h}_k + \mathcal{N}(\mathbf{0}, \sigma^2)$
 predictions[i] $\leftarrow \text{DECODER}(\mathbf{h}_k')$
end for
RANK(predictions) w.r.t. goal distance
 $\mathbf{x}_{k:T}^H \leftarrow \text{BEST}(\text{predictions})$
return $\mathbf{x}_{k:T}^H$

4. CONTROLLABLE MOTION PREDICTION

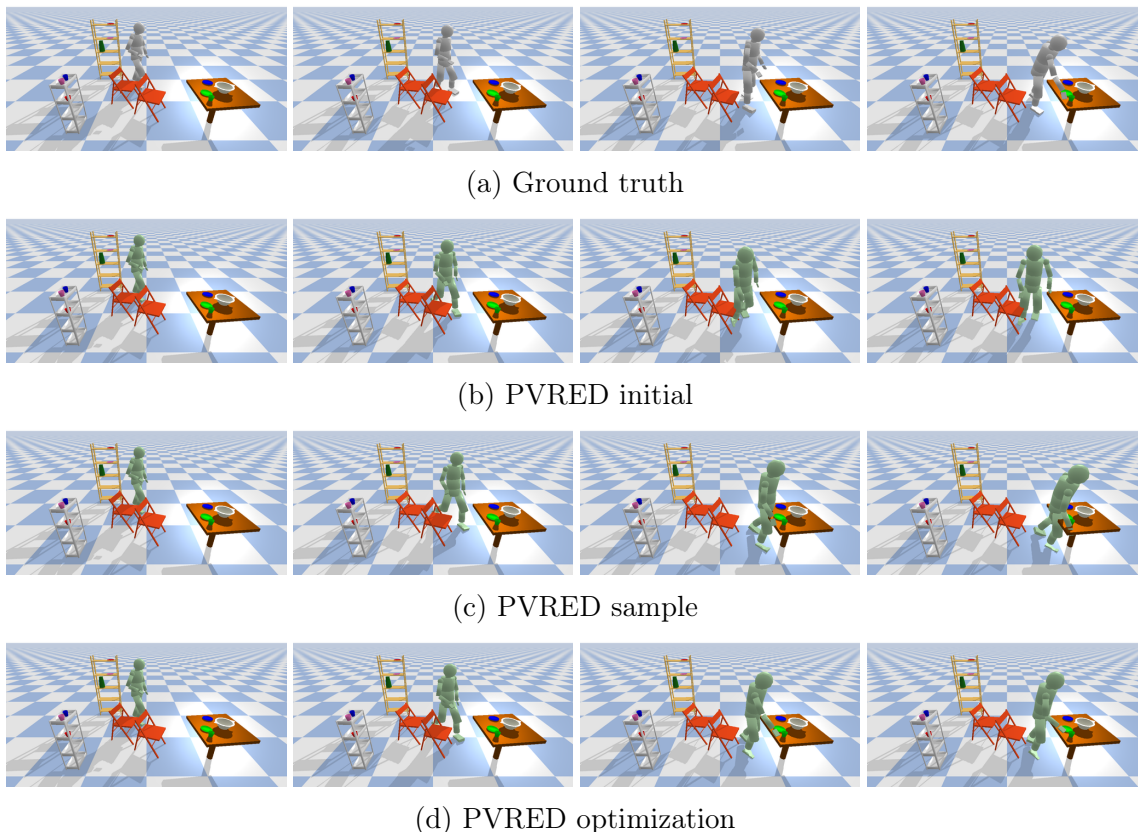


Figure 4.6: Example trajectories with goal constraint on MoGaze. The ground truth as well as the PVRED architecture with our optimization method and the *initial* and *sample* baselines can be seen. From left to right the prediction after 0s, 0.66s, 1.33s and 2s is shown. An extension of this figure with more timesteps and more baselines is available in section A.2.

the network and two seconds should be predicted. We compare the prediction of the network to the ground truth from the dataset.

Our method and the *sample* baseline are informed with a goal position of the human wrist extracted from the ground truth. For our method, we set a goal constraint to the wrist of the human on the last timestep. For the *sample* baselines, we simply take the sample which ends closest to the goal.

Reaching Trajectory Example

Figure 4.6 shows an example reaching trajectory and the predicted trajectory for the PVRED baselines. It can be seen that the initial prediction from the network does not end up with the hand close to the goal. The sample baseline improves this

and finds a trajectory with the hand ending closer to the goal. Our method finds a similar trajectory with the hand ending at the goal. In this example, the prediction by our method differs slightly to the sample baseline. Using our method, the human prediction is rotated slightly differently and does not end as close to the table, which is more similar to the ground truth trajectory. Also, we observed more artifacts, such as sliding motions with the sampling baseline. An extension of the figure with more timeframes, all different baselines, and architectures is available in section A.2.

Statistical Experiments

Table 4.3 shows the mean over trajectories for different metrics to the ground truth at several timesteps. The first part (i) shows the initial predictions, the second part (s) shows the predictions by the sample baseline, and the third part (o) shows the results for our optimized framework. The *base pos metric* is the distance to the base in meters. The angle metric is the mean relative angle, $L = 2 \arccos(\mathbf{q}^\top \mathbf{q})$ with \mathbf{q} being rotations in quaternion representation of all joints. The angle arm metric is the same but only for the right arm joints *rElbow* and *rShoulder*.

Evidently, *residual* and *PVRED* perform best for the *initial* baselines with *PVRED* being slightly superior in the base position and angle metric and the residual being slightly better for the angle arm metric. The results of the methods in terms of position at the goal can be improved by using the *sampling* strategy. However, this increases the angular losses. With our optimization based framework, the results

Table 4.3: Oracle Goal Constraint on the MoGaze dataset. Comparison of distance to ground truth for the initial baseline (i), for the sample baselines (s), and for our method (o).

seconds	base pos metric					angle metric					angle arm metric				
	0.4	0.8	1.2	1.6	2.0	0.4	0.8	1.2	1.6	2.0	0.4	0.8	1.2	1.6	2.0
zerovel i	0.27	0.53	0.77	0.96	1.06	0.23	0.31	0.31	0.4	0.41	0.44	0.7	0.91	1.09	1.11
basic i	0.2	0.25	0.3	0.41	0.5	0.29	0.33	0.36	0.38	0.38	0.51	0.74	0.8	0.73	0.7
residual i	0.1	0.2	0.29	0.38	0.46	0.18	0.26	0.29	0.33	0.34	0.41	0.66	0.82	0.74	0.79
PVRED i	0.09	0.18	0.25	0.33	0.41	0.18	0.25	0.29	0.32	0.33	0.43	0.68	0.83	0.78	0.81
basic s	0.51	0.49	0.41	0.37	0.39	0.4	0.41	0.42	0.44	0.44	0.82	0.91	0.94	0.85	0.81
residual s	0.16	0.27	0.32	0.32	0.35	0.25	0.33	0.35	0.37	0.38	0.62	0.82	0.94	0.85	0.88
PVRED s	0.15	0.25	0.29	0.28	0.31	0.26	0.32	0.34	0.35	0.36	0.57	0.82	0.86	0.83	0.82
zerovel o	0.17	0.29	0.37	0.41	0.43	0.24	0.32	0.32	0.41	0.42	0.45	0.7	0.95	1.13	1.16
basic o	0.22	0.28	0.28	0.33	0.4	0.29	0.33	0.34	0.37	0.37	0.51	0.72	0.79	0.74	0.7
residual o	0.1	0.19	0.24	0.24	0.27	0.19	0.26	0.28	0.3	0.3	0.41	0.65	0.8	0.72	0.7
PVRED o	0.09	0.18	0.23	0.23	0.25	0.18	0.26	0.28	0.29	0.3	0.41	0.68	0.81	0.7	0.68

of all architectures improve over the other baselines. While at the beginning of the trajectory, at 0.4 and 0.8 seconds, the pure prediction forecasts have a similar distance to ground truth than our methods, our methods significantly outperform at later timesteps (1.2, 1.6 and 2 seconds), which are closer to the goal. The best result is achieved by the *PVRED* architecture with our optimization framework.

Our results clearly confirm our experiment’s hypothesis and thus demonstrate that our framework, with introducing controls in the architecture, can be used to improve motion prediction by specifying constraints and using gradient-based optimization. Note that for prediction purposes, it is possible to obtain a goal location using intent prediction or task knowledge (see Chapter 6).

4.5.4 Comparing Different Possibilities for Specifying Controls

There are multiple ways on how controls can be specified in the architecture. In the *PVRED* architecture, adding controls to the input of the RNN cell of the states and its finite differences to the velocities intuitively makes sense, since they serve as a velocity control signal when specified this way. Nevertheless, there are multiple possibilities of how the state of the network can be changed in order to influence the prediction. In this experiment, we empirically compare possible ways of placing the control signal \mathbf{u}^H .

Table 4.4: Possibilities for specifying controls. For each option it is shown which exact term is added to the specific cell input or output.

	Cell Input			Cell Output	
	pose \mathbf{x}_t^H	velocity \mathbf{v}_t^H	hidden \mathbf{h}_t	pose \mathbf{x}_{t+1}^H	velocity \mathbf{v}_{t+1}^H
No controls	0	0	0	0	0
Velocity input	0	\mathbf{u}_t^H	0	0	0
Position input	\mathbf{u}_t^H	0	0	0	0
Concat input	$\mathbf{u}_{t,\text{pos}}^H$	$\mathbf{u}_{t,\text{vel}}^H$	0	0	0
FD input	\mathbf{u}_t^H	$\mathbf{u}_{t+1}^H - \mathbf{u}_t^H$	0	0	0
Velocity output	0	0	0	0	\mathbf{u}_t^H
Position output	0	0	0	\mathbf{u}_t^H	0
Concat output	0	0	0	$\mathbf{u}_{t,\text{pos}}^H$	$\mathbf{u}_{t,\text{vel}}^H$
Hidden	0	0	\mathbf{u}_t^H	0	0

Possibilities for Controlling the RNN

An overview over the possible ways we compare can be seen in Table 4.4. The following possibilities of adding controls to the cell inputs or outputs in the PVRED architecture are shown:

- *No optimization*: Baseline with not changing the predictions at all.
- *Velocity input*: Only changing the velocities by adding the control signal to the input of the cell.
- *Position input*: Only changing the positions by adding the control signal to the input of the cell.
- *Position and velocity concatenation input*: Changing both: position and the velocities, with an individual set of controls at the input of a cell. Thus, the dimension of the controls doubles in size since the control vector is a concatenation of controls for the state and controls for the position.
- *Position and Velocity Finite Differences (FD) input*: Changing the position by adding the control signal and changing the velocities by adding the finite differences derivative of the control signal at the input of the cell.
- *Velocity output*: Only changing the velocities at the output of a cell before they are added to the previous states by the residual connection. Thus, it changes both, position and velocity output at the next cell. Furthermore, it directly changes the predicted state.
- *Position output*: Only changing the positions at the output of a cell after the velocities are added to the previous state by the residual connection.
- *Position and Velocity concatenation output*: Changing both: position and the velocities, with an individual set of controls, at the output of the cell.
- *Hidden*: Changing the hidden state of the RNN by adding the control signal to it. The dimension of the control signal is the dimension of the hidden state.

Using the control signal to change the output of the RNN cell, has the effect that the predicted states are directly changed. In contrast, changing the input of a cell, the predicted states can not be directly influenced, as the states need to go through

the RNN cell functions first. This affects taking the derivatives and can lead to different optimization behavior. As the hidden state does not have a direct influence of the prediction, there is no difference for changing it at the input or output of the cell.

Empirical Evaluation

We perform this experiment on our synthetic data set with sinus curves (see section A.1.2). As this data set has lower dimensionality, training the models and evaluating on the test and validation data is feasible on a larger scale and it is possible to generate an arbitrary amount of data points.

We want to compare the influence of different architectures with different complexities, to achieve this, we change the number of layers between one and five layers and keep the hidden dimension per layer at a fixed size of 50. We generate 1000 trajectories for training and 100 for testing the model. We initialize the weights of the neural network at random and train the model for 50 episodes. For each of the above possibilities for specifying the control signal, we run our optimization framework with an oracle goal constraint on the test set. The mean distance to the ground truth after convergence of the optimization over the test set gives the error for the model per control signal possibility. The experiment is repeated 50 times per architecture, leading to a total of 250 trained models.

Table 4.5 shows how many of the 50 trained models per layer perform best with the given configuration. It can clearly be seen that not optimizing at all, never leads

Table 4.5: Comparison of Different Possibilities for Specifying Controls. The number of models which perform best by the specific possibility are shown. Per layer there are 50 trained models.

l	Cell Input					Cell Output			
	no	vel	pos	FD	concat	pos	vel	concat	hidden
1	0	14	14	16	4	0	0	0	2
2	0	7	4	19	1	5	1	3	10
3	0	10	3	34	0	1	0	0	2
4	0	12	4	28	2	0	0	1	3
5	0	12	8	25	2	2	0	0	1
sum	0	55	33	122	9	8	1	4	18

to the best performance as expected. Specifying the position and velocities at the input using finite differences significantly works best, in more than 50% of the cases at the higher layers.

In general, it is evident that specifying at the input of a cell leads to better performance than specifying at the output of the cell. A reason for this could be that specifying at the output of a cell can shift the prediction more far away from the data distribution, since the gradient directly pulls on it and no restrictions are made to the changes, whereas by specifying at the input cell, only states that are predicted by the RNN cell are possible and the gradient always goes through the cell.

Adding the control signal to the hidden states also is possible and improves prediction performances, which is interesting, since the controls specified in that manner have no intuitive meaning. However, specifying the controls as hidden state, usually leads to higher dimensionality of the control vector, and has less often the best optimization performance.

The intuitive method of influencing position and velocities of the PVRED input by using finite differences performs best. A reason for this could also be that the encoded dependence, position and its velocities, in the control signal, leads to more reasonable inputs that are more closely at the data distribution and less noisy. Our experiments on real human motion data also confirm that the control signal specified like that lead to good performance.

4.6 Alternative Method with a Gaussian Process as Human Model

While most of the work in this chapters focused on the application of RNNs for the human dynamics, it is also possible to use other techniques as underlying human model. In this section we describe an alternative approach using a GP instead of a RNN for modeling the human dynamics. More details about this method can be found in our publication [10].

4.6.1 Method

The approach is similar to the RNN-approach and works in three phases:

4. CONTROLLABLE MOTION PREDICTION

1. Offline, a predictive dynamics model of the human $\mathbf{x}_{t+1}^H = f^H(\mathbf{x}_t^H)$ is learned that comes from a GP model: $f^H(\mathbf{x}_{0:k-1}^H) \sim \text{GP}(\mu(\mathbf{x}_{0:k-1}^H), c(\mathbf{x}_{0:k-1}^H, \mathbf{y}_{0:k-1}^H))$. In contrast to the RNN based model, a GP is also predicting a variance.
2. Online, we use the learned model to unroll a trajectory of future states starting with the state at the current time step \mathbf{x}_{k-1}^H .
3. Finally, the trajectory of future states $\mathbf{x}_{k:T}^H$ is optimized further by simultaneously minimizing the distances to the mean predictions, the variances of the predictions and additional constraints, for instance goal set constraints.

To specify the GP distribution f^H , we make use of the following Radial Basis Function (RBF) kernel defined between trajectories:

$$c(\mathbf{x}_{0:k-1}^H, \mathbf{y}_{0:k-1}^H) = \alpha_1 e^{-\frac{1}{2}\alpha_2 \|\mathbf{x}_{0:k-1}^H - \mathbf{y}_{0:k-1}^H\|_{\Theta}^2} \quad (4.17)$$

with $\alpha_{1,2}$ being the base hyperparameters of the RBF and Θ being a $i \times d$ matrix of hyperparameters weighting the entries of the trajectory for time steps i and state dimension d . RBFs are commonly used with GPs. They have the advantage of being smooth and infinitely differentiable, which is also a desirable property for the trajectory optimization step.

Gaussian Process

We use the mean function of GP regression for predicting the next state \mathbf{x}_k^H when observing an unseen trajectory $\mathbf{x}_{0:k-1}^H$:

$$\mu(\mathbf{x}_k^H | \mathbf{x}_{0:k-1}^H, \mathcal{D}) = \mathbf{k}^\top \mathbf{C}^{-1} \mathbf{Y} \quad (4.18)$$

\mathbf{C} is the $N \times N$ covariance matrix with elements $\mathbf{C}_{nm} = c(\mathbf{y}_n^H, \mathbf{y}_m^H) + \beta^{-1} \delta_{nm}$, with the trajectories \mathbf{y}_n^H and \mathbf{y}_m^H from the training data \mathcal{D} and β being a white noise constant. \mathbf{k} is a vector with elements $k_n = c(\mathbf{y}_n, \mathbf{x}_{0:k-1}^H)$ and \mathbf{Y} is a matrix of next states from the training data with rows $\mathbf{y}_n = \mathbf{y}_k^H$.

The corresponding variance function of the GP is

$$\sigma^2(\mathbf{x}_k^H | \mathbf{x}_{0:k-1}^H, \mathcal{D}) = c(\mathbf{x}_{0:k-1}^H, \mathbf{x}_{0:k-1}^H) - \mathbf{k}^\top \mathbf{C}^{-1} \mathbf{k} \quad (4.19)$$

which gives the uncertainty about the prediction of the next state given training data \mathcal{D} and current trajectory $\mathbf{x}_{0:k-1}^H$.

For predicting multiple steps into the future, we simply use a sliding window of size k . This means we append the predicted state to the trajectory and remove the first state.

Hyperparameter Tuning

The covariance function c of the GP relies on the scalar hyperparameters α_1 and α_2 and the matrix hyperparameters Θ .

Because integrals over the parameters in a GP are analytically tractable, it is possible to compute the marginal likelihood. The log marginal likelihood for a column vector $\mathbf{y} \in \mathbf{Y}$ is given by:

$$\ln p(\mathbf{y}|\alpha_{1,2}, \Theta) = -\frac{1}{2}\ln|\mathbf{C}| - \frac{1}{2}\mathbf{y}^\top \mathbf{C}^{-1}\mathbf{y} - \frac{N}{2}\ln(2\pi) \quad (4.20)$$

Details about how the marginal log likelihood is obtained are available in Rasmussen and Williams [86]. The hyperparameters of the covariance function can be adapted to the data by minimizing the Negative Log Marginal Likelihood (NLML) of the GP with respect to the hyperparameters (see Equation 4.20). We can obtain the NLML for each column in \mathbf{Y} corresponding to each dimension of the state space and then minimize the sum of the NLML using a Conjugate Gradients (CG) algorithm.

Optimizing the Trajectory

For optimization, we minimize the distance to the mean of the next prediction. Additionally, we introduce a cost term penalizing high-variance. This cost term has the effect of moving the prediction close to the data distribution (low variance) and, therefore ensures that also on later timesteps the prediction remains useful.

The optimization problem is defined as:

$$\min_{\mathbf{x}_{k:T}^H} \sum_{d=k}^T \underbrace{\|\mathbf{x}_{d+1}^H - f_\mu^H(\mathbf{x}_{d-k:d}^H)\|^2}_{\text{mean cost}} + \underbrace{\gamma f_{\sigma^2}^H(\mathbf{x}_{d-k:d}^H)}_{\text{variance cost}} \quad (4.21)$$

$$\text{subject to } h(\mathbf{x}_{k:T}^H) = 0 \quad (4.22)$$

where γ is a weight describing the importance of the variance minimization, and h being an additional goal constraint. Because of the RBF kernel structure, the derivation of the gradient of the loss function is straightforward. We minimize the

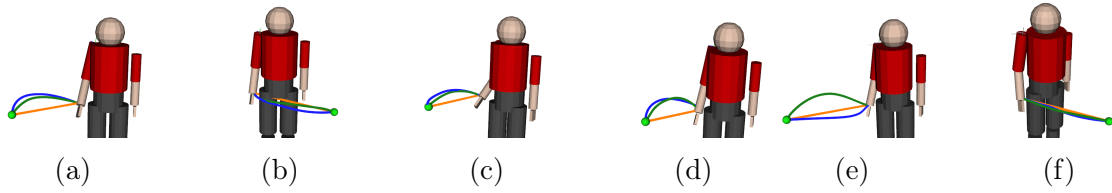


Figure 4.7: Example trajectories with the GP approach. Ground truth (green), prediction by our method (blue) and linear prediction baseline (orange). All trajectory durations are one second.

cost function using Sequential Least Square Programming (SLSQP), an optimization algorithm designed for constrained non-linear optimization problems [87].

4.6.2 Experiments

For the experiments we use the upper-body dataset (see section A.1.1).

We computed 15 test trajectories on the test set with reaching data. The mean distance between the predictions for the wrist position by our method and the ground truth is 0.078 meters ($SD = 0.061$) which is less than the linear prediction with a mean distance of 0.087 meters ($SD = 0.045$). While our prediction method outputs a trajectory for the whole human, the linear prediction is only a baseline for the wrist position.

Figure 4.7 shows examples of the predicted trajectories along with the ground truth and the linear prediction baseline, which simply interpolates the hand position.

While many of the predicted trajectories are very similar to the ground truth trajectory, some trajectories remain further away from the ground truth in the specific case, for example, Figure 4.7 (e) and (h).

4.6.3 Comparison to the Recurrent Neural Network Approach

An issue with GPs is that at test time, for predicting the next state, it is required to compute the vector \mathbf{k} , which requires to compute the kernel function over the whole dataset, leading to the fact that query time is increasing by the size of the dataset. In contrast, the RNN always has a fixed size. While increasing the dataset usually increases the training time, the query time of a single data point remains constant. This makes the RNN more suitable for larger datasets, such as our MoGaze dataset.

An advantage of GPs is the possibility to obtain a variance, which is helpful for sampling multiple trajectories or can be used in the optimization objective as described in our approach.

Improving GPs to better scope with larger datasets is an ongoing research field and also, combining RNNs and GPs by learning a kernel for a GP is possible [88]. However, as state-of-the-art research on motion prediction focuses on deep learning methods, and we also found them to work better than the GP approach, we focus on RNN approaches in this thesis.

4.7 Summary

In this chapter we addressed RQ2, “How can we adapt a data-driven human predictive model to fulfill external criteria?”. Adapting motion prediction is useful in order to constrain the prediction to environmental conditions or tasks, and for simulating human behavior with different conditions. Thus, we proposed a framework that allows to control a predictive, RNN-based human model. We added additional control parameters to the RNN architecture. The use of trajectory optimization allowed us to solve for control parameters that satisfy user-defined constraints, such as collision or goal constraints. By keeping the changes of the individual inputs small by penalizing them in the objective function, we ensured that the predictive capabilities of the neural network are maintained.

Our experiments validate our hypothesis that our framework can be used to improve prediction performances of human motion. The framework tackles multiple of the open challenges in motion prediction we identified in section 3.6, as it can be used to explore multiple options (multi-modality), incorporate environment constraints, and account for accumulating errors by incorporating constraints. Furthermore, we showed that the method can also be applied to non RNN approaches, such as GPs.

In this chapter we made use of oracle goal constraints, which demonstrate in experiments with real motion data that the prediction can be improved by our method. In Chapter 6 we discuss how to predict possible goal locations from the human behavior and perform longer time predictions.

An open challenge is the use of the human model for prediction in a HRC scenario. We will extend the framework to a human and a robot agent in the next chapter.

Co-Optimizing Human-Robot Trajectories

In HRC scenarios, where a human and a robot are acting and collaborating in close proximity, there are two problems arising simultaneously:

1. Planning a trajectory for the robot partner.
2. Predicting a trajectory for the human partner.

A prediction of the human trajectory is required for planning the robot motion in order to plan coordinated behavior, such as handovers or coordinated pick and place motions, and to ensure trajectories that are safe to execute, for example, by avoiding collisions between both partners.

On the other hand, the plan of the robot is required for predicting human motion, as the human adapts his or her motion depending on the robot's action, for example, reaching towards the robot hand for a handover or by making room for the robot to avoid collisions.

This leads to a **circular dependency** (see Figure 5.1). The concurrent nature of the two problems raises the need for simultaneous prediction and planning as well as for enabling the possibility of a bidirectional information flow between the two agents.

In the previous chapter we proposed a novel framework for controlling a human prediction to adapt to the environment. In this chapter, we aim to tackle the question on how we can plan robot motion to coordinate with the predictive human

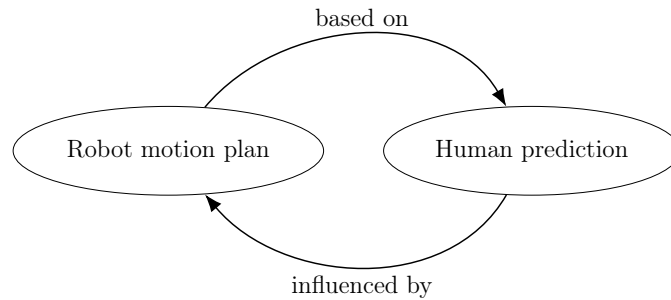


Figure 5.1: Circular dependency of planning and prediction. The robot motion plan is based on the prediction for the human. The human would change his or her behavior based on what the robot would do. Consequently, the prediction is influenced by the robot motion plan and vice versa.

model (RQ3). Thus, we build on top of the ideas from the previous chapter and extend the framework for planning robot motion and predicting human motion at the same time. We achieve this by formulating the HRC problem as NLP and using trajectory optimization for planning a robot motion and adapting the human motion concurrently.

Parts of this chapter are based on the following publication:

- Kratzer, P., Toussaint, M., and Mainprice, J. "Planning coordinated human-robot motions with neural network full-body prediction models". 2022 (under submission)

5.1 Related Work

In the following, we first describe related work on human aware motion planning. Afterwards, we look into related work on trajectory optimization.

5.1.1 Human-Aware Motion Planning

HRC is a rapidly growing research field. It focuses on robotic systems that are able to fulfill a common task with a human. The human and the robot are considered as members of a human-robot team. Usually, human and robot will interact with each other and perform actions in close proximity. A main challenge in close proximity

interaction is ensuring safety and comfort of the human partner, while maintaining time and energy efficient execution [89, 90].

Preferred Interaction Strategies

Studies found that humans tend to prefer interaction strategies with a robot that shows proactive behavior. For example, Baraglia et al. investigated whether the robot or the human should take initiative during collaborative tasks and found that people collaborate best with a proactive robot, yielding better team fluency [91]. Schulz et al. performed similar experiments [21]. The authors found that in tasks with a higher cognitive load people prefer the robot to lead the interaction. Also, in other tasks a proactive robot behavior is desired.

Human-Awareness

In order to achieve efficient and proactive execution, the human partner needs to be taken into account *explicitly* when planning the robot's motion, leading to human-aware motion planning systems. This is confirmed by a study by Lasota and Shah who found that human-aware motion planning improves human-robot team fluency and human worker satisfaction [92].

Multiple possibilities to achieve human-aware motion planning exists. For example, Kulić and Croft propose to introduce human-awareness by incorporating a cost function to evaluate the safety of a robot path [93]. Many works try to predict which part of the workspace will be occupied by the human and avoid this area [94, 95]. This strategy of avoiding the human leads to high safety, however, it could decrease efficiency and it avoids direct collaboration. In our opinion, a real collaborative robot should be able to plan interactions with the human instead of avoiding it.

Human Comfort

In order to ensure human comfort, Sisbot and Alami propose to explicitly reason on human's kinematics, field of view, posture, and preferences [96]. Kruse et al. found that *Proxemics*, considering public, personal and private spaces, are important to ensure human comfort [97].

Interaction-Aware Planning

One problem in multi-agent scenarios, such as in HRC, is that predicted trajectories of other agents are influenced by own actions. In autonomous driving and navigation, interaction-aware and intention-aware planning frameworks have been proposed. Evestedt et al. generate trajectory candidates and evaluate them by simulating the nearby traffic situation by taking the effect on other drivers into account [98]. Bandyopadhyay et al. incorporate human intention uncertainty into a planning framework [99]. This allows a robust pedestrian avoidance policy in autonomous navigation. To address uncertainty in human predictions, Fridovich-Keil et al. propose a confidence-aware motion planner that takes the uncertainty of the predictive model into account [100].

Summary

In contrast to prior work in human-aware motion planning, we co-optimize robot and human motion, using a predictive human behavior model. We incorporate interaction paradigms, such as *Proxemics*, as constraints in trajectory optimization. We use a kinematic model for the human, allowing to incorporate planning respecting a variety of coordination paradigms.

5.1.2 Trajectory Optimization

Gradient-based optimization algorithms are widely used in the field of robotics and optimal control for optimizing trajectories [34, 101, 102, 103]. These techniques have been shown to successfully generate motions with a variety of kinematic and dynamic objectives and constraints.

Trajectory Optimization in Robotics

An interesting approach for trajectory optimization is CHOMP, proposed by Ratliff et al. [34]. CHOMP is a trajectory optimization technique that is capable of avoiding collisions by using covariant gradient techniques. Schulman et al. proposed TrajOpt, which uses a NLP formulation using obstacle avoidance with inequality constraints [103]. Toussaint proposed KOMO, a newton method based approach [104]. A tutorial relating newton methods to other fields is given in [38]. Also, the utilization

of interior point methods for navigating narrow passages have been proposed by Mainprice et al. [105].

Trajectory Optimization for Human Behavior

While many work has been proposed for optimizing robot trajectories, there is also related work that uses trajectory optimization to synthesize human behavior [81, 82]. Those works are often interested in generating natural looking motions. Moreover, it is possible to use trajectory optimization in order to optimize a human full-body model respecting ergonomic criteria [106]. It would be possible to incorporate such ergonomic constraints for human movement into our framework, in order to constrain the robot motions so that it leads to ergonomically beneficial human trajectories.

Concurrent Human and Robot Trajectory Optimization

While our work is the first to our knowledge, which uses trajectory optimization on a human model and a robot model simultaneously, first proposed in [11], similar ideas have been proposed in the meantime. For example, Fishman et al. propose a method to simultaneously plan for robot arm movements while predicting human actions [107]. In their work, the human model is a simple floating sphere model. For pedestrian prediction, Schaefer et al. use gradient information of a neural network for planning two-dimensional trajectories within a crowd of pedestrians [108].

In contrast to those works, our approach makes use of a full-body motion prediction model for the human, which allows to plan motions using a single framework for tasks, such as handovers, or tasks where navigation as well as pick and place actions are combined.

5.2 Framework Overview

An overview of the framework can be seen in Figure 5.2. Our method for co-optimizing human and robot motion works in two phases. Offline, a RNN human model for predicting motion is trained on human motion data to learn the network weights θ . Online, trajectory optimization is used to optimize human and robot trajectories simultaneously to find human controls $\mathbf{u}_{k:T}^H$ and robot controls $\mathbf{u}_{k:T}^R$. The optimization respects user-defined constraints, such as environmental, task

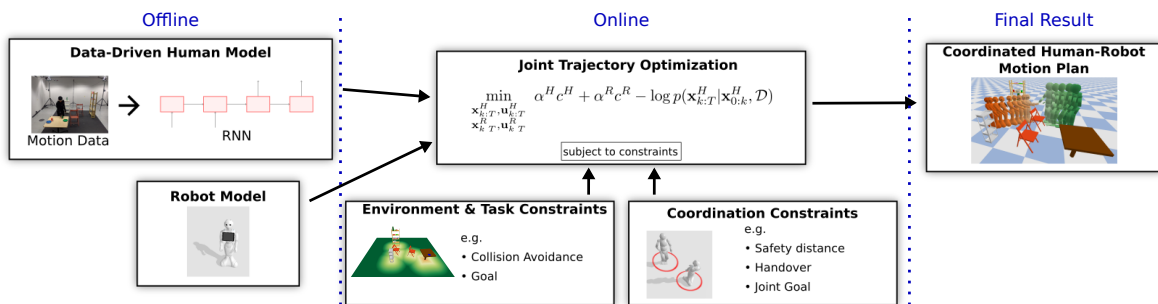


Figure 5.2: Overview of the framework. Offline, a human model is trained on motion data and a robot model is constructed. Online, the states and controls of human and robot are jointly optimized, accounting for task, environment and, coordination constraints. The final result is a coordinated motion plan.

and coordination constraints. The final result is a coordinated human-robot plan, consisting of the human predicted trajectory and the planned robot trajectory.

5.3 A Nonlinear Program for Human-Robot Collaboration Problems

In this section we describe the HRC problem as a NLP. We do this similar to the pure human motion prediction case from the previous chapter (see section 4.2). Here, we formulate the problem in a more general manner with arbitrary agents. The pure human motion forecasting problem from the previous chapter can be seen as a special case with only one agent. Interestingly, the formulation with multiple agents allows introducing constraints that depend on both agents. This enables the possibility of a bidirectional information flow between the two agents.

5.3.1 Nonlinear Program Formulation

We consider scenarios where multiple agents a are involved who have to solve specific tasks in the environment. For simplification, we consider and evaluate our framework with one human agent H and one robot agent R . Note that our framework can be easily extended to multiple agents.

We discretize time with a small constant time change of Δt between consecutive timesteps. At a timestep t , each agent $a \in \{H, R\}$ has a state $\mathbf{x}_t^a \in \mathbb{R}^{d_a}$ with dimensionality d_a . When applying controls $\mathbf{u}_t^a \in \mathbb{R}^{n_a}$ a discrete dynamics function

f^a can be used to compute the state at the next timestep:

$$\mathbf{x}_{t+1}^a = f^a(\mathbf{x}_t^a, \mathbf{u}_t^a) \quad (5.1)$$

In order to arrange coordinated motion, it is necessary to plan ahead for all involved agents. Thus, we are interested in planning the future trajectory of robot states $\mathbf{x}_{k:T}^R$, as well as planning a *likely* future trajectory of human states $\mathbf{x}_{k:T}^H$, where T is a suitable time horizon and k is the first predicted timestep. The states $\mathbf{x}_{0:k-1}^a$ are the past states of the agents which we assume are known or observed.

We formulate our HRC problem, which aims to plan a robot trajectory $\mathbf{x}_{k:T}^R$ while predicting a likely human trajectory $\mathbf{x}_{k:T}^H$, as the following NLP:

$$\min_{\substack{\mathbf{x}_{k:T}^H, \mathbf{u}_{k:T}^H \\ \mathbf{x}_{k:T}^R, \mathbf{u}_{k:T}^R}} \alpha^H c^H(\mathbf{x}_{k:T}^H, \mathbf{u}_{k:T}^H) + \alpha^R c^R(\mathbf{x}_{k:T}^R, \mathbf{u}_{k:T}^R) - \log p(\mathbf{x}_{k:T}^H | \mathbf{x}_{0:k-1}^H, \mathcal{D}) \quad (5.2)$$

subject to:

$$\mathbf{x}_{t+1}^H = f^H(\mathbf{x}_t^H, \mathbf{u}_t^H) \quad \forall t \in (k, T) \quad (5.3)$$

$$\mathbf{x}_{t+1}^R = f^R(\mathbf{x}_t^R, \mathbf{u}_t^R) \quad \forall t \in (k, T) \quad (5.4)$$

$$h^H(\mathbf{x}^H) = 0, \quad g^H(\mathbf{x}^H) \leq 0 \quad (5.5)$$

$$h^R(\mathbf{x}^R) = 0, \quad g^R(\mathbf{x}^R) \leq 0 \quad (5.6)$$

$$h(\mathbf{x}^H, \mathbf{x}^R) = 0, \quad g(\mathbf{x}^H, \mathbf{x}^R) \leq 0 \quad (5.7)$$

with $c^H(\mathbf{x}_{k:T}^H, \mathbf{u}_{k:T}^H)$ and $c^R(\mathbf{x}_{k:T}^R, \mathbf{u}_{k:T}^R)$ being cost functions associated with the trajectories of human and robot, α^H and α^R are hyperparameters used to weight the influence of the respective agents, and the likelihood of the human motion given dataset \mathcal{D} and observed motion $\mathbf{x}_{0:k}^H$ is given by $p(\mathbf{x}_{k:T}^H | \mathbf{x}_{0:k-1}^H, \mathcal{D})$.

Equation 5.3 shows constraints ensuring the dynamics functions for the human and Equation 5.4 for the robot. Additional equality and inequality constraints for human (Equation 5.5) or robot (Equation 5.6) act on a single agent only and can be used to ensure environment-dependent and task constraints. Joint constraints between human and robot (Equation 5.7) are useful to ensure collaborative interaction paradigms.

The NLP we formulated for human and robot coordination problems, is similar to the NLP for pure human motion prediction formulated in section 4.2. The NLP naturally extends to include a robot agent by adding an additional robot trajectory cost term in the optimization objective, additional robot dynamics constraints, and

additional user-defined constraints for the robot, such as environment, joint limits, or task-specific constraints. The most important difference is the possibility to have joint constraints between both agents (Equation 5.7).

Our formulation defines both, planning for the robot and prediction for the human, in one NLP. The possibility to have constraints acting on both agents simultaneously allows a bidirectional information flow between the two agents and is one of the core ideas of our approach.

5.3.2 Solving the Nonlinear Program

Similar to the pure motion prediction NLP we described in the previous chapter (see section 4.2), we use the shooting method and the assumption that the human likelihood is integrated into the human dynamics function.

Accordingly, the optimization problem simplifies to:

$$\min_{\mathbf{u}_{k:T}^H, \mathbf{u}_{k:T}^R} \alpha^H c^H(\mathbf{x}_{k:T}^H, \mathbf{u}_{k:T}^H) + \alpha^R c^R(\mathbf{x}_{k:T}^R, \mathbf{u}_{k:T}^R) \quad (5.8)$$

subject to:

$$h^a(\mathbf{x}^a) = 0, \quad g^a(\mathbf{x}^a) \leq 0 \quad (5.9)$$

$$h(\mathbf{x}^H, \mathbf{x}^R) = 0, \quad g(\mathbf{x}^H, \mathbf{x}^R) \leq 0 \quad (5.10)$$

where c are the costs and h and g are user defined equality and inequality constraints.

The future states of the agents $\mathbf{x}_{k:T}^a$ are computed by the unrolled agent dynamics function:

$$\mathbf{x}_{t+1:T}^a = \tilde{f}^a(\mathbf{x}_t^a, \mathbf{u}_{t:T}^a) = (f^a(\mathbf{x}_t^a, \mathbf{u}_t^a), f^a(f^a(\mathbf{x}_t^a, \mathbf{u}_t^a), \mathbf{u}_{t+1}^a), \dots) \quad (5.11)$$

Constraints can act at any state which requires to compute the gradients through \tilde{f}^a for gradient-based optimization.

Algorithm 5.1 shows a high-level overview over the optimization procedure for co-optimizing human and robot motion. The inputs to the algorithm are the observed human trajectory $\mathbf{x}_{0:k}^H$ and the optimization parameters α^H and α^R . Both, the human and the robot future states, are computed by the respective unrolled dynamics functions. Using the computed states and the controls, the forward functions for the optimization objective, the human constraints, the robot constraints and the joint constraints are computed. Gradients of the forward functions are computed using the tensorflow library. The ipopt library uses the gradients and forward computations to

Algorithm 5.1 Human-Robot Joint Trajectory Optimization

Input: Observed trajectory $\mathbf{x}_{0:k-1}^H$, Robot initial state \mathbf{x}_{k-1}^R , Optimization parameters α^R, α^H

Init: $\mathbf{u}_{k:T}^H \leftarrow \mathbf{0}$, $\mathbf{u}_{k:T}^R \leftarrow \mathbf{0}$

for $n < \text{maxiter}$ **do**

$\mathbf{x}_{k:T}^H \leftarrow \tilde{f}^H(\mathbf{x}_{0:k-1}^H, \mathbf{u}_{k:T}^H)$

$\mathbf{x}_{k:T}^R \leftarrow \tilde{f}^R(\mathbf{x}_{k-1}^R, \mathbf{u}_{k:T}^R)$

Compute objectives c^H , c^R and constraints h_i , g_i

Compute gradients $\frac{\partial c^H}{\partial \mathbf{u}^H}$, $\frac{\partial c^R}{\partial \mathbf{u}^R}$, $\frac{\partial h_i}{\partial \mathbf{u}^H}$, $\frac{\partial h_i}{\partial \mathbf{u}^R}$, $\frac{\partial g_i}{\partial \mathbf{u}^H}$, $\frac{\partial g_i}{\partial \mathbf{u}^R}$ (tensorflow)

Compute logbarrier parameters (ipopt)

Approximate Hessian with BFGS (ipopt)

Compute search direction and step size (ipopt)

Update controls $\mathbf{u}_{k:T}^H$, $\mathbf{u}_{k:T}^R$

end for

return $\mathbf{x}_{k:T}^H$, $\mathbf{u}_{k:T}^H$, $\mathbf{x}_{k:T}^R$, $\mathbf{u}_{k:T}^R$

compute logbarrier parameters, approximate the Hessian and compute the search direction. These steps are repeated till convergence or a maximum number of iterations is reached. Finally, the algorithm outputs a prediction for the human states $\mathbf{x}_{k:T}^H$ and the human controls $\mathbf{u}_{k:T}^H$, as well as the robot states $\mathbf{x}_{k:T}^R$ and robot controls $\mathbf{u}_{k:T}^R$.

5.4 Objective and Constraints

In the following we explain the constraints and objective we use in the experiments for our human-robot joint trajectory optimization framework. Many more constraints are possible.

5.4.1 Objective

For both, human trajectory and robot trajectory, we penalize the magnitude of the control terms $c^H = \|\mathbf{u}^H\|_{\mathbf{A}}^2$ and $c^R = \|\mathbf{u}^R\|_{\mathbf{A}}^2$ where \mathbf{A} is a norm which approximates $c^H \approx \sum_t^T \|\dot{\mathbf{u}}_t^a\|^2$ by finite difference. The optimization objective simply is the weighted sum $\alpha^H c^H + \alpha^R c^R$ of both terms.

Both cost functions are similar and, thus, optimization behavior for both agents is similar. The parameters α^H and α^R we use in the optimization objective can be used to weight the influence of the cost term for each agent.

As the objectives only depend on the controls and not on the states, the dynamics functions are not needed. The derivative of the objective is straightforward to calculate and the computation does not take much computation time.

5.4.2 Single Agent Constraints

Single agent constraints can, for example, be used to avoid collisions with the environment or specify a pick or place goal for a single agent.

For constraints depending on single agents, we use the same as in subsection 4.4.2, as they can be used for the robot accordingly.

Thus, the goal constraint for an agent a at time t is:

$$h_t^{\text{Goal}} = \|(\phi_{\text{linkpos}}^a \circ \tilde{f}_t^a)(\mathbf{x}_k^a, \mathbf{u}_{k:T}^a) - p^*\|^2 \quad (5.12)$$

with ϕ^a being the forward kinematics map of the respective agent for a specific link, such as, the hand.

And collision avoidance using a SDF for an agent a at time t is:

$$g_t^{\text{Collision}} = (\text{SDF} \circ \phi_{\text{linkpos}}^a \circ \tilde{f}_t^a)(\mathbf{x}_k^a, \mathbf{u}_{k:t}^a) \quad (5.13)$$

with the possibility to either specify a constraint per timestamp or to specify it using a single constraint with a minimum function.

5.4.3 Joint Constraints

To avoid collision between the agents, we want to maintain a minimal clearance of d between the base positions of the agents at all time. We formulate this as an inequality constraint with the following *joint collision constraint* at every timestep t :

$$g_t^{\text{Jointcoll}} = \|(\phi_{\text{basepos}}^H \circ \tilde{f}_t^H)(\mathbf{x}_t^H, \mathbf{u}_{k:t}^H) - (\phi_{\text{basepos}}^R \circ \tilde{f}_t^R)(\mathbf{x}_t^R, \mathbf{u}_{k:t}^R) - d\|^2 \quad (5.14)$$

Similar to $g^{\text{Collision}}$ it is possible to formulate multiple constraints along the trajectory as a single constraint using the maximum over $g_{k:T}^{\text{Jointcoll}}$.

In case we want to grasp an object, but leave to the optimizer which of the agents should pick up the object and when the grasp should happen, the following *joint goal constraint* per timestep t can be used:

$$h_t^{\text{Jointgoal}} = \min(\|(\phi_{\text{hand}}^H \circ \tilde{f}_t^H)(\mathbf{x}_t^H, \mathbf{u}_{k:t}^H) - p^*\|^2, \quad (5.15)$$

$$\|(\phi_{\text{hand}}^R \circ \tilde{f}_t^R)(\mathbf{x}_t^R, \mathbf{u}_{k:t}^R) - p^*\|^2) \quad (5.16)$$

with p^* being the grasp point for the object. To compute it over all the prediction timesteps we use the minimum over $h_{k:T}^{\text{Jointgoal}}$ letting the optimizer decide, which timestep is suitable for the grasp.

In order to plan for handover motion, we consider the following *handover constraint*:

$$h_t^{\text{Handover}} = \left\| (\phi_{\text{handpos}}^H \circ \tilde{f}_t^H)(\mathbf{x}_t^H, \mathbf{u}_{k:t}^H) - (\phi_{\text{handpos}}^R \circ \tilde{f}_t^R)(\mathbf{x}_t^R, \mathbf{u}_{k:t}^R) \right\|^2 + \quad (5.17)$$

$$\left\| (\phi_{\text{baserot}}^H \circ \tilde{f}_t^H)(\mathbf{x}_t^H, \mathbf{u}_{k:t}^H) - (\phi_{\text{baserot}}^R \circ \tilde{f}_t^R)(\mathbf{x}_t^R, \mathbf{u}_{k:t}^R) \right\|^2 \quad (5.18)$$

where the first part (Equation 5.17) of the handover loss is the distance between the hand positions of both agents. We define ϕ_{handpos}^H and ϕ_{handpos}^R to be an offset to the hand palms so that the hands end up in a distance suitable for handing over objects. The second part (Equation 5.18) is the distance of the base rotations so that it equals to 0 when the agents face each other.

5.5 Agent Dynamics Functions

For the human agent we apply the controllable RNN model we introduced in subsection 4.2.2.

A typical dynamics function for the robot agent f^R uses torques or velocities as controls \mathbf{u}^R and position and velocities as states \mathbf{x}^R . Hence the dynamics function can be derived from the connectivity of the kinematic chains and the mass parameters of the robot.

In our experiments we use a model of the Pepper¹ robot with active joints for its right arm. We model the dynamics as simple velocity control. The states are composed of the robot two-dimensional base position $\mathbf{x}_{t,0,2}^R$, the robot base orientation $\mathbf{x}_{t,2}^R$, and joint angles $\mathbf{x}_{t,3:d_R}^R$ that arise from the robot kinematics. The controls are base velocity $\mathbf{u}_{t,0}^R$, angular base velocity $\mathbf{u}_{t,1}^R$, and the velocity controlled joint angle

¹<https://www.softbankrobotics.com/emea/en/pepper>

inputs $\mathbf{u}_{t,2:d_R-1}^R$. This leads to the following dynamics function:

$$\mathbf{x}_{t+1}^R = f^R(\mathbf{x}_t^R, \mathbf{u}_t^R) = \begin{pmatrix} \mathbf{x}_{t,0}^R + \cos(\mathbf{x}_{t,2}^R) \cdot \mathbf{u}_{t,0}^R \\ \mathbf{x}_{t,1}^R + \sin(\mathbf{x}_{t,2}^R) \cdot \mathbf{u}_{t,0}^R \\ \mathbf{x}_{t,2}^R + \mathbf{u}_{t,1}^R \\ \dots \\ \mathbf{x}_{t,d_R}^R + \mathbf{u}_{t,d_R-1}^R \end{pmatrix}$$

5.6 Empirical Evaluation

In the following sections we evaluate our human-robot coordination framework on our MoGaze dataset (see section 7.2).

5.6.1 Baselines

As earlier, we use the best performing PVRED model from our hyperparameter search (see subsection 3.5.3) for the experiments.

We compare the following methods and baselines:

- **initial:** We only use the initial prediction of the PVRED. The robot trajectory is optimized with respect to the predicted trajectory. Only the robot trajectory can be changed by the optimizer, the human prediction is kept fixed.
- **sample:** We sample 100 human predictions by adding Gaussian noise to the encoder hidden state \mathbf{h}_t before inputting it to the decoder. The predictions are ranked based on the distance to the goal for the joint collision experiments and based on the initial handover loss for the handover experiments. The robot is optimized with respect to the best ranked prediction. In case of failure, we continue with the next best prediction till a successful trajectory is found or all samples have been tested. An overview of the sample baseline can be seen in Algorithm 5.2.
- **ours:** Our method optimizes human and robot concurrently allowing both agents to adapt to each other.

For the collision avoidance experiments we use the following additional baselines:

- **with_coll** optimizes human and robot without considering the other agent. Both agents can collide.

Algorithm 5.2 Sample Baseline for Coordinated Planning and Prediction.

```

Input: Observed trajectory  $\mathbf{x}_{0:k}^H$ , variance  $\sigma^2$ 
 $\mathbf{h}_k \leftarrow \text{ENCODER}(\mathbf{x}_{0:k}^H)$ 
for  $i \in \{0, 1, 2, \dots, 100\}$  do
     $\mathbf{h}'_k \leftarrow \mathbf{h}_k + \mathcal{N}(\mathbf{0}, \sigma^2)$ 
    predictions[ $i$ ]  $\leftarrow \text{DECODER}(\mathbf{h}'_k)$ 
end for
predictions_ranked  $\leftarrow \text{RANK}(\text{predictions})$ 
repeat
     $\mathbf{x}_{k:T}^H \leftarrow \text{POP}(\text{predictions\_ranked})$ 
     $\mathbf{x}_{k:T}^R, \mathbf{u}_{k:T}^R, \text{success} \leftarrow \text{Trajectory optimization w.r.t. } \mathbf{x}_{k:T}^H$ 
until success = TRUE or predictions_ranked =  $\emptyset$ 
return  $\mathbf{x}_{k:T}^H, \mathbf{x}_{k:T}^R, \mathbf{u}_{k:T}^R, \text{success}$ 

```

- **human_avoids** first optimizes a robot trajectory with respect to goal and scene obstacles. Then the human trajectory is optimized to avoid the robot.
- **robot_avoids** is similar to *human_avoids*, but first optimizes a trajectory for the human, then for the robot.

5.6.2 Joint Collision Avoidance Experiments

In this experiment we want to identify a shared motion plan containing a coordinated robot and human trajectory, which ensures a safety distance between human and robot. We achieve this by using the *joint collision constraint* (see Equation 5.14). We set the minimal distance between human base and robot base to be $d = 0.5\text{m}$. Further, for each of the agents, we define a goal constraint and a SDF collision constraint similar as in the previous chapter (see Figure 4.4).

From the test set, we extract 76 pick trajectories of human motion. The trajectories have a length of three seconds. We use one second of the extracted human motion as observation and predict for the remaining two seconds. We set the goal constraint for the human end position from the ground truth data. For the robot the start state and goal constraint are defined manually. We design the problem so that the trajectories of the human and robot need to get close to each other and, thus, are likely to interfere.

Quantitative Results

We plan coordinated motion trajectories for the human and robot on the set of problem described above using the baselines and our methods.

We consider three variants of our method with different scaling parameters for human and robot in the optimization objective:

- **human_prio** uses $\alpha^H = 100$ and $\alpha^R = 1$.
- **robot_prio** uses $\alpha^H = 1$ and $\alpha^R = 100$.
- **ours** uses $\alpha^H = \alpha^R = 10$.

We compute several metrics on the resulting trajectories: the travel distance of the human and robot base as well as the smoothness metrics for the robot trajectory. The smoothness metrics are given by the negative mean squared jerk, the log dimensional jerk, and the spherical arc length. The smoothness metrics are designed that values closer to zero mean smoother trajectories.

Table 5.1 shows the median for the metrics and the success rate. We define a trajectory to be successful when the distance of the hand of the human to the goal is smaller than 0.1, the distance of the robot base to the goal is smaller than 0.2, human and robot do not collide, and the optimization objective is smaller than 0.1.

The *with_coll* baseline gives a lower boundary for the travel distance and smoothness metrics for the optimization based approaches since it optimizes without considering collision between the two agents. This leads to very smooth trajectories with

Table 5.1: Collision Experiments with Baselines.

Method	travel dist		smoothness			success rate
	human	robot	ms-jerk	ld-jerk	sparc	
with_coll	1.28	2.28	-0.03	-1.62	-2.51	13
initial	1.22	2.94	-8.35	-5.45	-2.27	0
sample	1.25	3.55	-15.7	-5.86	-2.35	3
human_avoids	1.11	2.26	-0.01	-2.45	-2.4	36
robot_avoids	1.32	3.19	-7.86	-5.64	-2.37	58
human_prio	1.33	2.97	-13.86	-6.04	-2.32	58
ours	1.32	2.72	-1.26	-4.66	-2.29	78
robot_prio	1.45	2.34	-0.15	-3.32	-2.31	83

very small total travel distances for human and robot but is only successful in 13% of the cases since human and robot would collide in the other cases.

The *initial* baseline has a success rate of 0. The reason for this is that it blindly forecasts and can only reach the goal randomly. In none of the predicted trajectories the hand ended up close to the goal so that it never found a successful trajectory of human motion. This success rate is only slightly improved by the *sample* baseline. While it is possible that the sample baseline finds a trajectory close enough to the goal, another requirement is to find collision free trajectories. It is very unlikely to find collision free trajectories with the hand ending up close to the goal only by sampling. As both methods output trajectories with the human not reaching the goal, the median path length of the human base is lower than the one reported by *with_coll*.

The *robot_avoids* and *human_avoids* baselines show higher success rates, with the *robot_avoids* being better than the *human_avoids* baseline. A reason for the superior performance of the *robot_avoids* baseline is that the robot is more flexible, since it is completely freely movable and not, like the *human_avoids* baseline, tied to the neural network prediction. A problem of the *robot_avoids* and *human_avoids* baselines is that there is no information flow between the two agents. One of the agents is treated as a blackbox and the other agent aims to find a collision free path. As it could be that, for example, the first optimized agent might block the path through a narrow passage, finding a successful coordinated motion plan is often not possible in the timeframe. While the overall success rate is improved over the *initial* and *sample* baselines, the robot often finds very long, non-smooth paths to reach the goal, leading to a long travel distance in the *robot_avoids* baseline. In the *human_avoids* baseline there are more non-successful trajectories, in which the human is blocked by the robot and does not reach the goal, leading to a lower human base travel distance.

In contrast, our method is able to plan a robot trajectory while adapting the human prediction to the plan. This results in finding a coordinated motion trajectory for both agents. The bidirectional information flow allows that the agents make room for each other by moving further at the side, wait to let the other agent pass, or speed up in order to make way for the other agent. This behavior shows that our method is able to find coordinated motion.

With a higher parameter for the human costs, as in the *human_prio* baseline, the

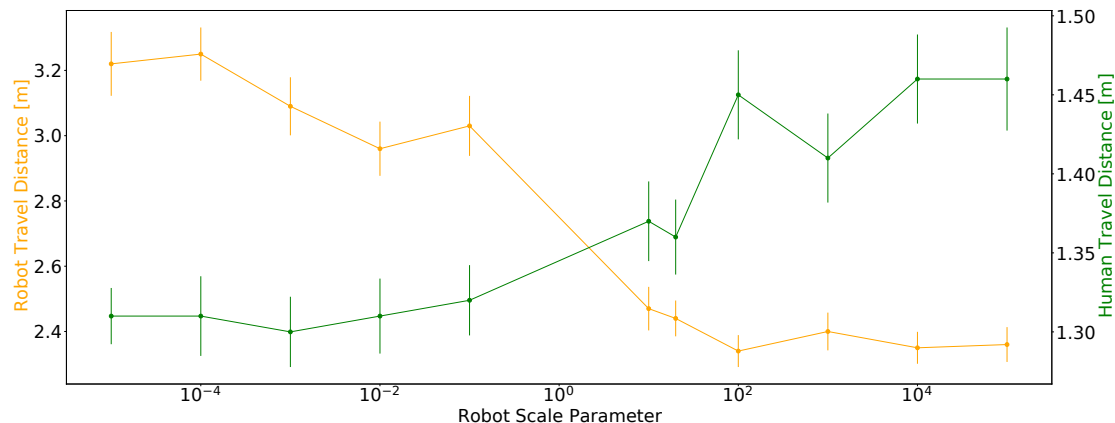


Figure 5.3: Influence of the robot cost scaling parameter. Higher α^R decreases the length of the path of the robot but increases the length of the human path.

changes made to the prediction of the human are too restrictive so that the prediction often does not fulfill the constraints leading to a lower success rate compared to the *robot_prio* and *ours* methods.

Ours and *robot_prio* are the best performing methods, with *robot_prio* increasing the path the human needs to take and decreasing the path length of the robot. Moreover, both methods are among the methods with the best smoothness losses for the robot.

Influence of the Robot Cost Scaling Parameter α^R

The optimization behavior can be influenced by the objective scaling parameters of human α^H or robot α^R . In this experiment, we evaluate the influence of these parameters by changing the robot cost scaling parameter α^R . Our hypothesis is that a high α^R leads to less change in the robot and decreases its base path length, and respectively, a low α^R increases the path length.

We run our method multiple times on the trajectories and compute the median path lengths of human and robot. The results are displayed in Figure 5.3, with error bars showing the median absolute error. As expected, increasing α^R decreases the robot travel distance but increases the human travel distance. As a consequence, it is possible to use this parameter to balance how much the human and robot should deviate from their predicted and optimal path respectively.

One can also see that the robot travel distance is changing more than the human's.

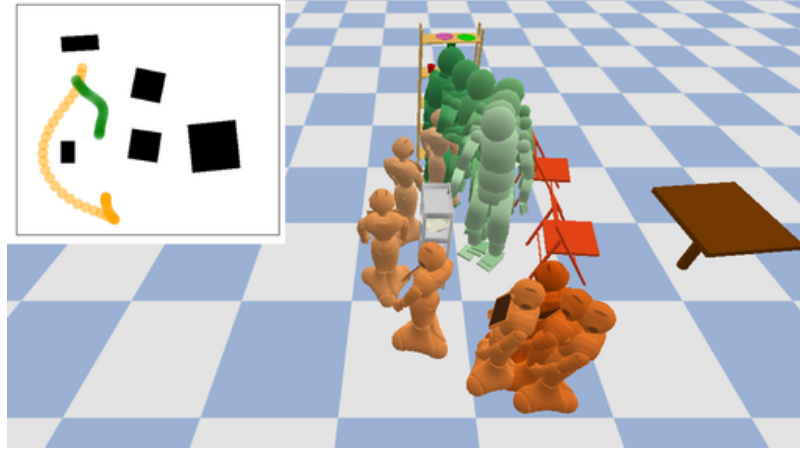


Figure 5.4: Robot avoiding the human by backing up and taking a longer route.

This is expected since the human is more closely tied to the prediction by the neural network while the robot can move completely free. For example, the robot can also move backwards to avoid the human (see Figure 5.4). Similar behavior for the human is unlikely since such motions are not included in the dataset. Limitations associated to the dataset will be discussed in section 5.7.

Qualitative Results

For a qualitative comparison, we consider an example where the human’s goal is to pick up an object from the shelf and the robot needs to move to a location on the opposite side. Figure 5.5 shows trajectories computed using our method and the two tuning variations of the objective function: *human_prio* and *robot_prio*. We consider two different scenarios: 1) the robot coming from the left side (left column) and 2) the robot coming from the right side (right column). The human observed motion is kept fixed in both scenarios. It can be seen that the human adapts to the side the robot is coming from. In the trajectories where the robot starts from the left side, the human slightly walks further to the right side and ends up further on the right side of the shelf to make room for the robot.

When α^H is increased (second row), the human is less adaptive. In both scenarios, the robot waits longer and speeds up more towards the end of the trajectory. With increased α^R (third row), the human needs to adapt much more and smoother trajectory for the robot are found.

To conclude, the joint collision avoidance experiments show that our method

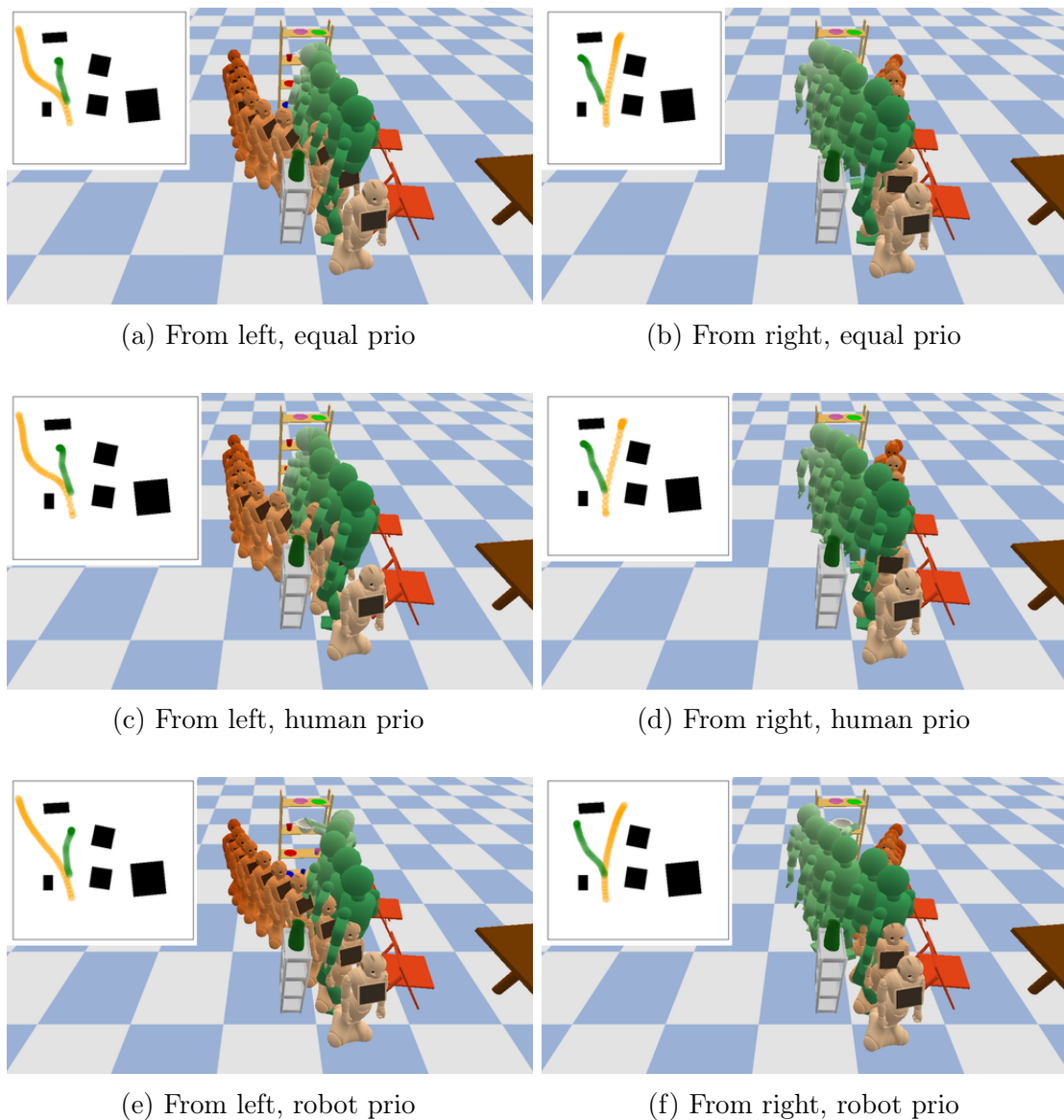


Figure 5.5: Joint collision avoidance example with robot coming from the left side and from the right side. Three different scaling parameters can be seen: equal prio $\alpha^H = \alpha^R$, human prio $\alpha^H = 100$, $\alpha^R = 1$, and robot prio $\alpha^H = 1$, $\alpha^R = 100$.

can be used to find a coordinated motion plan in a HRC scenario. The technique is able to adapt the human prediction to the robot plan as well as the robot plan to the human prediction simultaneously. Using optimization parameters, it can be influenced how much the individual agent deviate from their optimal path.

5.6.3 Computation Time

In terms of computation time, the bottleneck of our system is computing the gradients for the constraints which need to be computed through the neural network. We implemented the neural network with tensorflow. We use an Intel Core i7 laptop with 2.80GHz for the timing experiments. The computations are running on the Central Processing Unit (CPU).

For computing a two second trajectory with 20fps we measured the following mean computation times during the joint collision experiments: computing the goal constraint takes 4.97ms and its gradient takes 10.76ms, the SDF collision constraint takes 4.7ms and its gradient takes 10.5ms, the joint collision constraint takes 4.2ms and its gradient takes 9.12ms.

The optimizer we use usually needs to call the gradient computation once per iteration and the forward pass can be called multiple times. Usually, we optimize for 50 to 200 iterations, depending on the used constraints. As the computation of individual constraints is independent from each other, it would be possible to compute gradients for multiple constraints in parallel, making the cost of 100 iterations of ipopt approximately two seconds which can be further improved by simplifying the network, decreasing the framerate of the motion, using optimized real time neural network libraries, or using improved hardware, for example, using a Graphics Processing Unit (GPU) could decrease computation time

5.6.4 Handover Experiments

In this experiment we consider that human and robot want to perform a handover. We are interested in the planning phase of the handover where human and robot approach each other (i.e., within two seconds). The objective is that the human and robot perform a trajectory where both hands end up close to each other, so that either the robot or human can transfer an object over to the counterpart.

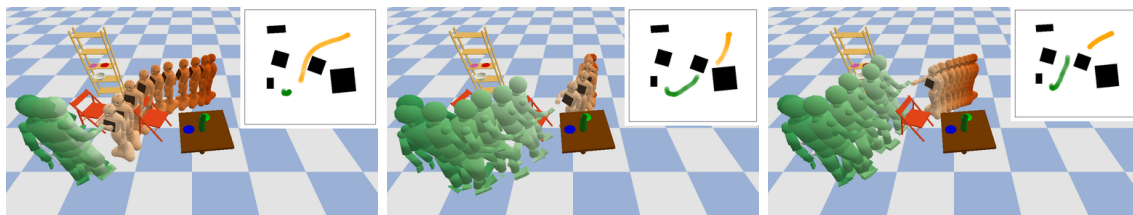


Figure 5.6: Handover Examples. From left to right: *initial* baseline, *sample* baseline, our method.

We perform the experiment on the MoGaze dataset. Trajectories are extracted and a robot agent is placed into the scene. We jointly optimize human and robot using our framework. We use the obstacle SDF constraint and the handover constraint. We compare to the *initial* baseline and the *sample* baseline. For the *sample* baseline, trajectories are ranked with respect to their initial handover constraint loss (see Equation 5.17 and Equation 5.18).

Qualitative Results

Figure 5.6 shows example trajectories. In this example, the initial baseline (left) predicts that the human is mostly standing and only turns slightly away from the shelf. Since this prediction is used as a blackbox, the robot has to move all the way towards the human, in order to plan a successful handover.

The *sample* baseline (middle) samples multiple predictions and chooses a prediction with a low initial handover loss. Thus, a prediction is chosen, in which the human walks in the direction of the robot. However, in this example, the human ends up on the right side of the chair. Thus, the handover with the robot is blocked by the chair. The robot needs to perform a slow rotation first and then speeds up towards the human because otherwise, it would collide with the chair.

In contrast, our method (right) is able to make small changes to the human trajectory and the robot trajectory simultaneously. This makes it possible to find a trajectory where the human and the robot directly move towards each other. The paths the human and the robot need to move, are better distributed than using the *initial* baseline and the resulting path is more direct and smooth compared to the *sample* baseline. A further advantage of our approach is that no handcrafted heuristics are required as the human trajectory is directly changed and adapted by the optimizer.

Quantitative Results

To further evaluate the framework, we perform a quantitative comparison of our method with the *initial* and *sample* baselines. From the test data, we extract 100 trajectories of human motion data. We place a robot into the scenario so that a handover with the human within the next two seconds is possible. We run our framework and the baselines on the extracted trajectories to find suitable handovers. Table 5.2 shows the median results of the method using the same metrics as in subsection 5.6.2: path length for human and robot base, smoothness losses, and success rates. We consider trajectories as successful when they not collide with obstacles and reach a handover loss threshold of < 0.1 and an objective threshold of < 0.1 .

The lowest success rate of 32% has the *initial* baseline. Since the *initial* baseline does neither take the relative position of human and robot nor any context from the environment into account, this makes sense, as the baseline consequently often collides with an obstacle, faces towards obstacles, or faces away from the human. Moreover, this results in a longer travel distance of the robot, as the robot needs to adapt to the human prediction and often needs to take a longer route.

With a success rate of 75%, the *sample* baseline performs much better. This is based on the fact that the method samples multiple futures for the human and, thus, can pre-select a superior sample and re-optimize with other sampled predictions in case of failure. However, it is still possible that all future trajectories collide with obstacles, for example, when the human would need to walk through a narrow passage, which can be given by two chairs standing close to each other. In such scenarios, it is hard to find a successful path just by sampling.

Our method is able to make small changes to the human prediction and, thus, can adapt the human and robot trajectories to each other continuously. As a consequence,

Table 5.2: Handover Experiments with Baselines.

Method	travel dist		smoothness			success rate
	human	robot	ms-jerk	ld-jerk	sparc	
initial	1.2	2.19	-2.57	-4.6	-2.11	32
sample	1.34	2.16	-30.34	-5.17	-2.15	75
ours	1.13	1.09	-0.09	-1.89	-2.19	90

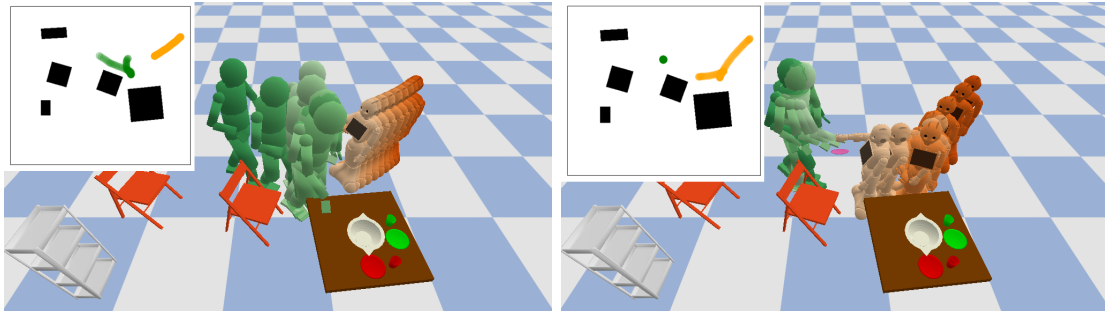


Figure 5.7: Example of a joint goal constraint followed by a handover constraint. Left: Human picks up the object and hands it to the robot. Right: Human movement is penalized, hence, the robot picks up the plate.

our method is able to find optimized trajectories for the human and robot, minimizing both, the human change to its initial trajectory and the robot cost function. With our method, the travel distances for both agents are smaller compared to the other baselines. With 95% our method significantly improves the success rate over the baselines.

5.6.5 Pickup before Handover

In this experiment the *joint goal constraint* (Equation 5.15) is evaluated, which lets the optimizer specify who of the agents is selected to fulfill the goal and at which timestep the goal is fulfilled. In this example, we set the goal to pickup a plate from the table. Additionally, we set a handover constraint at the end of the trajectory.

Figure 5.7 shows two example trajectories. In the initial configuration (left), the human picks up the object and rotates towards the robot for transfer. In a second configuration (right), we are interested to see the behavior of our framework when the human is not picking up the plate. To achieve this, we simply add a cost term to the objective that penalizes movement of the human base. As a consequence, the framework automatically adapts and the optimizer finds a trajectory where the robot picks up the object and hands it over to the human.

This result shows that the optimizer can be used to select a suitable agent for a task, depending on the specific configuration. The *joint goal constraint* can be used to select which agent is suitable for picking up an object.

5.7 Discussion and Limitations

Our experiments demonstrate that our framework is able to predict realistic human motions. The framework is based on the underlying neural network architectures and the prediction can be changed in order to account for differentiable constraints by using trajectory optimization. While we use a RNN as underlying neural network, the underlying data-driven model can be changed to other methods.

5.7.1 Dataset

The neural network model is trained on a specific dataset and predicted trajectories are within the data distribution of this dataset.

This makes the underlying dataset a crucial part of our framework, as motions that are outside the data distribution are unlikely to be predicted. For example, in the MoGaze dataset no humans are walking backwards and therefore no backwards walking humans are predicted in our experiments.

An approach to improve the framework is to enlarge the dataset in order to incorporate many possible motions. We think that data could particularly benefit from including interactive motion, such as human-human or human-robot interaction trajectories. We therefore started to capture human-human full-body interaction trajectories (see section A.1.1). Another approach to cover a large fraction of possible human motion trajectories is the AMASS dataset which aims to aggregate multiple datasets from different sources into a common parameterization [109].

5.7.2 Trajectory Optimization

Trajectory optimization is a powerful class of methods that can be applied for motion planning. Generically, it uses gradient-based optimization, e.g. Newton’s method, applied to a trajectory objective. Its flexibility allows its application to many problems with varying differentiable constraints. Our experiments demonstrate that trajectory optimization can successfully be used in order to optimize trajectories through a recurrent neural network model for human dynamics.

A drawback of trajectory optimization is that trajectory optimization algorithms are local methods and the trajectory objective is typically non-convex. As a consequence, this can lead to sub-optimal solutions by getting stuck in poor local minima. Possibilities to tackle this issue are: restarting of the optimization with different

initial values, combining trajectory optimization with sampling-based approaches, for example, to warm-start the optimization, or by convexifying the problem [105].

5.8 Summary

In this chapter we discussed the problem of planning coordinated robot motion and, thus, addressed RQ3: “How can we plan robot motion to coordinate with the predictive human model?”. We identified that a main problem in combining human motion prediction and planning is the circular dependency that arises between the two agents.

In order to tackle this problem, we introduced a novel framework for human-aware motion planning in HRC scenarios. The framework uses a RNN model for predicting human motions as well as gradient-based optimization to change the predictions while planning a robot trajectory. Thus, it is possible to take environmental, task and coordination constraints into consideration for both, the predicted human trajectory and the planned robot trajectory.

We demonstrated our method on experiments with joint collision avoidance and handovers. Our results show that a RNN based predictive human model can be used for shared human-robot planning and that the concurrent planning allows to adapt the robot to the prediction and the prediction to the robot plan at the same time.

Intent Prediction and Planning for Longer Tasks

While the previous chapters focused on short-term motion by forecasting and planning motion starting from the current timeframe, this chapter focuses on the aspects of motion prediction in the long-term.

One key ability for long-term motion prediction is the estimation of the intention of the human (RQ4). Intention can be defined as the goal of the human, such as the object a human is going to pick, or the end position of a place action. In this chapter, we will investigate intent prediction using eye-gaze and affordance-inspired ideas. Moreover, in the second half of the chapter, we tackle prediction of long pick and place sequences (RQ5) by using hierarchies and propose a method for planning robot motions with respect to it.

Parts of this chapter are based on the following publications:

- Kratzer, P., Midlagajni, N. B., Toussaint, M., and Mainprice, J. "Anticipating human intention for full-body motion prediction in object grasping and placing tasks". International Conference on Robot and Human Interactive Communication (RO-MAN) (pp. 1157-1163). © 2020 IEEE
- Kratzer, P., Bihlmaier, S., Midlagajni, N. B., Prakash, R., Toussaint, M., and Mainprice, J. "Mogaze: A dataset of full-body motions that includes workspace geometry and eye-gaze". Robotics and Automation Letters, 6(2), 367-373. © 2020 IEEE

- Le, A. T., Kratzer, P., Hagenmayer, S., Toussaint, M., and Mainprice, J. "Hierarchical Human-Motion Prediction and Logic-Geometric Programming for Minimal Interference Human-Robot Tasks". International Conference on Robot and Human Interactive Communication (RO-MAN) © 2021 IEEE

6.1 Related Work

In the following we will discuss relevant related work. The first section will be discussing intent prediction, the second will focus on prediction and planning for longer time horizons.

6.1.1 Intent Prediction

Human motion usually follows a higher-order goal, which is called *human intent*. For example, the human may intent to pick up a specific object. In HRC, the ability to estimate human intent, is very helpful for planning robot motion.

In related work, intent prediction often is achieved by predicting a discrete action or a goal position. As the full trajectory of a human often is required, for example, to predict which parts of the workspace will be occupied by the human partner, many works use intent prediction to estimate a goal position for the human and use the predicted intent for motion prediction in a second step. For example, Elfring et al. use a HMM to predict a goal position and predict a human path by using social forces [50]. Bennewitz et al. compute a probabilistic belief of future human positions and incorporate this belief into path planning [48].

Similar to these ideas, this chapter demonstrates the use of an intent prediction model and its combination with our motion prediction framework.

Object Affordances

One possibility to improve intent prediction is to take the notion of object affordances into account. The concept of affordances was introduced by Gibson and stems its roots in psychology [110, 111]. Object affordances aim to answer the question of how an object can be used by an agent and, thus, relates to the action patterns an object offers to the agent. A comprehensive survey of affordances by Jamone et al. is available in [112].

Visual Affordances

Numerous related work on affordances tries to estimate an affordance from an image or video. The field of visual affordances targets extracting information regarding affordances as a computer vision problem [113]. For example, Roy et al. use a Convolutional Neural Network (CNN) based architecture to extract affordance segmentations in RGB images [114]. Nguyen et al. model affordances using an autoencoder structure [115].

Affordances in Robotics

In Robotics, affordances can be used to model the actions a robot is able to perform [116, 117, 118]. For example, Montesano et al. use Bayesian networks to encode affordances and demonstrate how a humanoid robot can use it to interact with objects [116]. For human robot interaction, affordance models are used to model human action possibilities and to be able to infer human intent [52, 119]. Koppula and Saxena define object affordance as potential functions depending on how the object will be interacted with [52].

In this thesis, we design and implement a system to understand human object affordances in a real world table setup task performed in a motion capture environment. As we aim to apply the affordance model in order to predict human motion, we use a probabilistic model. Given the human state and the scene context, it predicts a density of interaction possibilities for the corresponding affordance. In particular, we concentrate on graspability and placeability affordances, and model them using a probabilistic neural network framework.

Eye-Gaze for Intent Prediction

Humans often start looking towards an object before they start to move [120]. This indicates that gaze features are a useful indicator for intent prediction and early detection of human activity. In related work, for instance, Huang and Mutlu performed an experiment where participants had to order items from a robot [121]. Using data from eye tracking, their system uses early prediction to enable the robot to proactively support the human. Similarly, Hoffmann et al. found that gaze features are helpful for activity detection and anticipation and showed that a LSTM based method outperforms a Support Vector Machine (SVM) [122]. In another experiment

on action anticipation, Duarte et al. investigated nonverbal cues and found that eye-gaze provides the key information that helps humans identify actions of other humans correctly [123].

In our work, we evaluate whether eye-gaze can help to identify human intent on our MoGaze dataset which combines full-body motions tracked with a motion capture framework and eye-gaze from an eye tracker.

6.1.2 Long Time-Horizons

Prediction of very long time horizons is challenging due to the many interaction possibilities that an environment offers. While full-body forecasting methods (see Chapter 3) are capable of predicting motion for several seconds, predicting a task that, for example, consists of multiple pick and place actions is challenging and is not addressed by a lot of publications.

An approach of planning whole tasks often found in related work, is to include high-level symbolic planning. For example, Alami et al. propose a human-aware task planning system, in which the robot plans for itself and for the human in order to find a human-aware robot plan [124]. Lemaignan et al. found that combining symbolic and geometric planning leads to natural human-robot interaction [125].

Task and Motion Planning in Robotics

High-level planning has been studied in classical AI and planning paradigms have been developed to solve symbolic planning [126]. In Robotics, these concepts have been integrated in order to solve motion planning problems. The field of TAMP aims to find sequential manipulation movements fulfilling a high-level task. TAMP involves reasoning on a symbolic level which provides discrete action sequences, and continuous motion planning which tries to find motion trajectories fulfilling the discrete action sequence.

Multiple approaches to solve TAMP are possible, with the main areas being sampling-based approaches [127, 128, 129], constrained-based approaches [130, 131], and numerical optimization based methods [132]. Toussaint proposed Logic-Geometric Programming (LGP) which combines logic tree search with trajectory optimization techniques [132]. We use a similar approach in order to perform TAMP with a human-robot team.

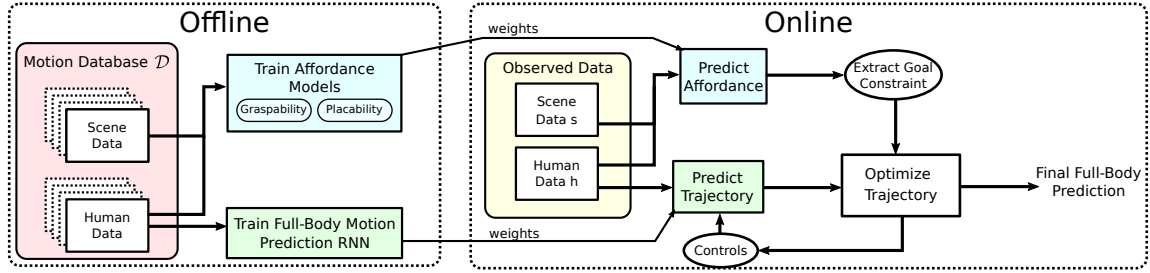


Figure 6.1: Affordance Prediction System Overview. Offline, affordance and full-body motion prediction models are trained on the dataset. Online, a goal constraint is extracted from the affordance and the full-body prediction is optimized to fulfill the constraint.

Task and Motion Planning in HRC

In HRC, related work focused on combining symbolic and geometric planning [125, 133, 134]. In [135], a human-robot collaboration task is implemented using LGP, where the human prediction is modeled with simple cost terms.

In our work, we combine TAMP with a learned predictive model of human motion. We achieve this by searching a symbolic state space combined with a human motion prediction model, and use trajectory optimization for finding motion trajectories fulfilling the planned action sequence. By iterative replanning, the plan can be changed to adapt to the updated human prediction.

6.2 Affordances for Intent Prediction

In this section we will discuss an intent prediction framework and combine it with our controllable human predictive model from Chapter 4, in order to predict full-body motion trajectories. Further details are available in our paper [12].

6.2.1 Overview

Figure 6.1 gives an overview of the framework. Offline, we use our dataset \mathcal{D} and train multiple neural networks on it. We train two probabilistic object affordance models, one for graspability and one for placeability. Additionally, we train a PVRED for full-body motion prediction. While the affordance models are trained on human data and scene data, the full-body prediction model is only trained on human data.

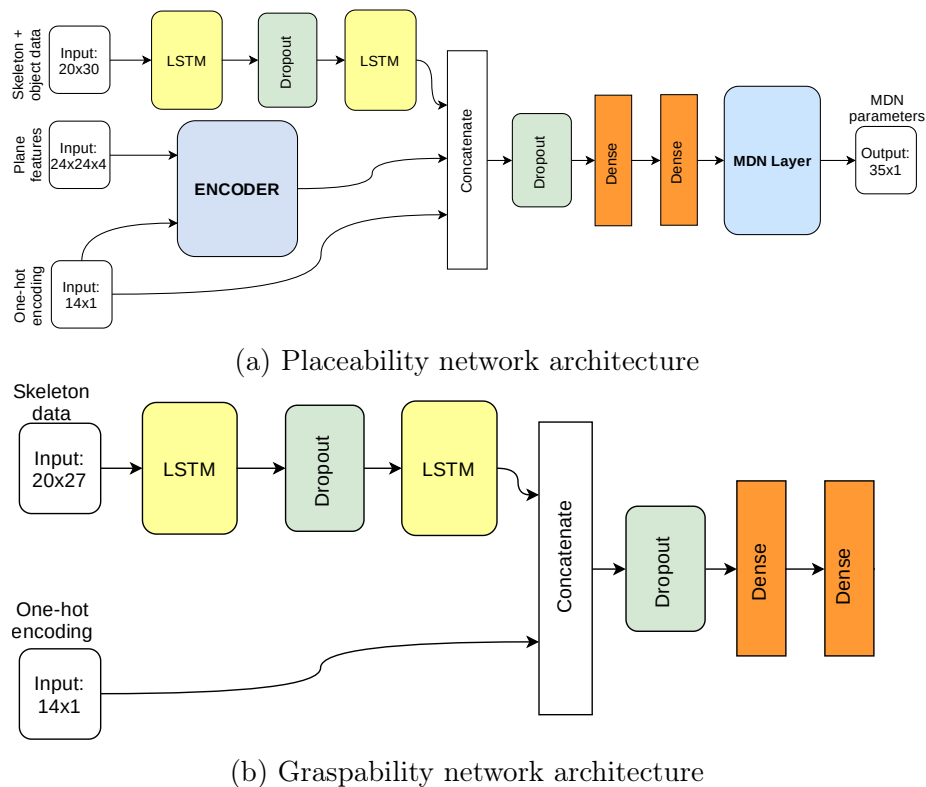


Figure 6.2: Affordance network architectures. The placeability network has additional inputs for plane occupancy.

At prediction time, the affordance model is used to extract a goal position. Using the goal position, we use trajectory optimization to adapt the human prediction with respect to a goal constraint. Finally, the framework returns a full-body prediction trajectory which takes the intent of the human into account.

6.2.2 The Affordance Networks

Affordances describe how the human would interact with an object. Intent is a combination of the affordance and the prediction of the object or the part of the scene the human will interact with.

Here, we aim to find a probabilistic model:

$$p_{o,a}(x|\mathbf{x}_{0:k-1}^H, e) \quad (6.1)$$

where o is an object, a is the action, in our case $a \in (\text{pick}, \text{place})$, and e is the environment. The interaction possibilities x , for example, can be the hand location

of the human while placing. For instance, $p_{\text{table, place}}$ would give a probability over possible place locations on the table, while $p_{\text{jug, grasp}}$ would give us a probability over possible wrist locations for grasping a jug.

Placeability

The first affordance we look into is the *placeability* affordance which we define as a probability distribution over possible place locations on a surface.

As placeability is multi-modal, we use a Mixture Density Network (MDN) to model the distribution [136]:

$$p(x|d) = \sum_{i=1}^m \alpha_i \phi_i(x|d) \quad (6.2)$$

where m indicates the count of the components in the mixture model, α_i are the mixing coefficients, and ϕ_i are functions representing conditional densities for the i^{th} kernel.

We model the placeability affordance using the neural network architecture shown in Figure 6.2a. The inputs d to the model are the human skeleton and object states in positions over a trajectory of one second (20 timesteps), a 14 dimensional one-hot encoding of both, the object type the human has in the hand and the surface we compute the affordance for, and a grid that covers the plane state.

The network additionally takes plane features as input which are a 24×24 grid consisting of a binary occupancy map, a two-dimensional position of the planes reference frame and a SDF (see Figure 6.3).

The network is trained using a Neg-Log Likelihood (NLL) loss with the two-dimensional place position on the surface as ground truth.

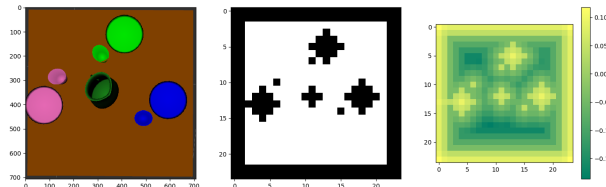


Figure 6.3: Plane features. From left to right: A visualization of the table with several objects, the corresponding binary occupancy map, a visualization of the signed distance field.

Graspability

We model graspability with the von Mises-Fisher (vMF) distribution. The intention of the vMF formulation is to model grasp points as a distribution on a two-dimensional manifold defined on the surface of a sphere:

$$p_{\text{vMF}}(x; \mu, \kappa) = C_p(\kappa) \exp(\kappa \mu^T x) \quad (6.3)$$

where $\mu \in \mathbb{R}^p$, $\|\mu\| = 1$ is the mean direction, with $p = 3$ and $\kappa \geq 0$ is the concentration parameter which defines the spread of the distribution on the surface of the hypersphere in the direction of μ . $C_p(\kappa)$ is the normalizing constant given by

$$C_p(\kappa) = \frac{\kappa^{p/2-1}}{(2\pi)^{p/2} I_{p/2-1}(\kappa)}, \quad (6.4)$$

where I_s denotes the modified Bessel function of the first kind at order s .

The base structure of our base graspability model is shown in Figure 6.2b. The output of the network are the parameters for the vMF distribution. The network is trained on two loss functions simultaneously: NLL of the vMF distribution evaluated at ground truth direction and mean squared loss for the distance parameter.

6.2.3 Combination with the Controllable Model for Full-Body Prediction

Now we make use of the affordance based intent prediction model for conditioning the full-body prediction. The aim is to predict a likely future human trajectory $\mathbf{x}_{k:T}^H$ with prediction horizon T , based on a previously observed trajectory $\mathbf{x}_{0:k-1}^H$ and based on the affordance prediction.

In order to do this, we use our controllable human motion model from chapter 4. We add a goal constraint to the optimizer so that the end state \mathbf{x}_T^H fulfills a sample from $p_{o,a}(x|\mathbf{x}_{0:k-1}^H, e)$. For example, a constraint can be that the hand of the human ends up at the predicted grasp point, or over the predicted place position.

6.2.4 Experiments

In the following we want to test the prediction framework with the intention prediction module on real motion data. Our hypothesis is that the prediction of a standard

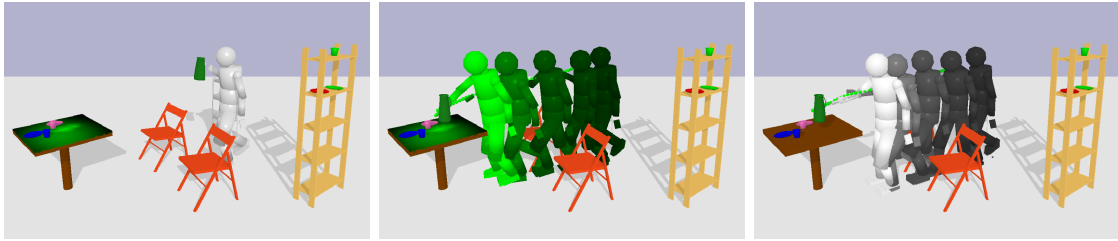


Figure 6.4: Prediction for placing a jug on the table. A placement affordance is predicted as a probability density function on the table (a), depicted in green a full-body motion is optimized (b), which is compared to ground truth motion (c).

PVRED can be improved by an additional optimization step, without having to retrain the model.

The PVRED is trained on the training subset of the MoGaze dataset (see section 7.2). From the test part of the dataset, we extract 27 trajectories of placing trajectories that end up on the table. We use the place affordance model and extract the expected place point p^* from the MDN by taking the most likely point. Also, it would be feasible to sample multiple points in order to achieve multiple possible trajectories. The PVRED is used to predict a trajectory of 1.5s of human motion. We specify a goal constraint for trajectory optimization so that the prediction ends up above p^* .

An example comparison for one of the trajectories of our method to ground truth can be seen in Figure 6.4. On the left side the start frame of the prediction is depicted with an affordance heatmap visualized on the table. The second frame shows several frames of the human trajectory for placing the jug. On the right side the ground truth is depicted. The prediction is similar to the ground truth and only differs slightly in the exact position of the human at the end.

Table 6.1 shows the distance to the ground truth at different timesteps. In the first part of the table, the sum over distances of key joints (wrists, elbows, knees, ankles and pelvis) is shown, in the second part only the distance of the wrist to the ground truth is shown. Values are averaged over the 27 trajectories.

We compare the following baselines:

- **zerovel** keeps the current state as prediction and does not predict for future timesteps.
- **initial** unrolls the recurrent neural network without taking the goal into account

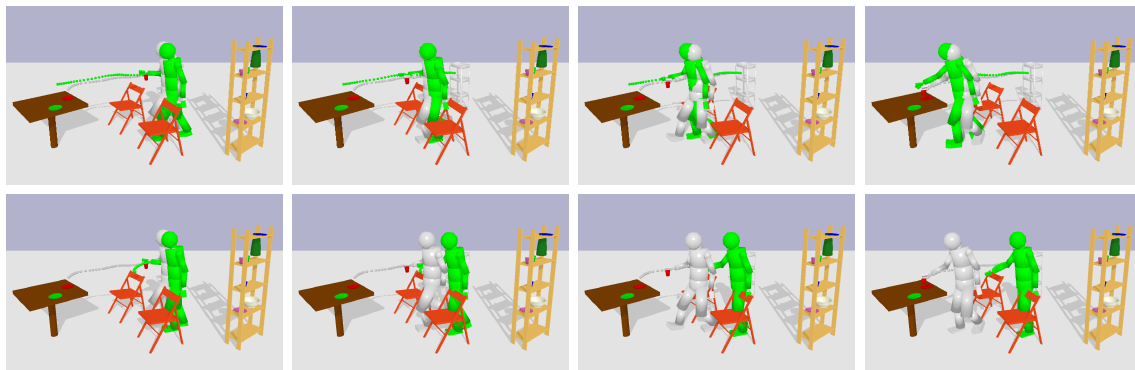


Figure 6.5: Full-body prediction example. From left to right the trajectories after 33ms, 66ms, 100ms and 133ms. Prediction is depicted in green, ground-truth in gray. The top row shows the trajectory with goal optimization towards the affordance, the bottom trajectory is the trajectory by the *initial* baseline without goal optimization.

- **ours** uses the goal extracted from the place affordance. The trajectory is optimized to end up at p^*
- **oracle** uses the true goal position from the wrist obtained from the dataset and optimizes the trajectory to end up at the exact position.

It can be seen that the oracle prediction performs best which is not surprising, as it uses information that is not available at prediction time. Our method using the place point prediction performs second best and outperforms the prediction without any optimization at all time steps.

Table 6.1: Error of state prediction per time step for whole body (b) and right wrist (w). Reported values are in meters. For the whole body, the sum distance of nine key joints is shown.

	ms	250	500	750	1000	1250	1500
zerovel (b)		2.38	5.18	7.76	9.68	11.09	11.94
initial (b)		0.88	1.70	2.82	4.01	5.27	6.30
ours (b)		0.86	1.43	2.00	2.42	2.70	2.80
oracle (b)		0.86	1.44	1.84	2.03	2.19	2.18
zerovel (w)		0.26	0.56	0.86	1.09	1.29	1.39
initial (w)		0.09	0.18	0.30	0.44	0.60	0.73
ours (w)		0.10	0.19	0.28	0.31	0.28	0.28
oracle (w)		0.09	0.19	0.24	0.22	0.15	0.08

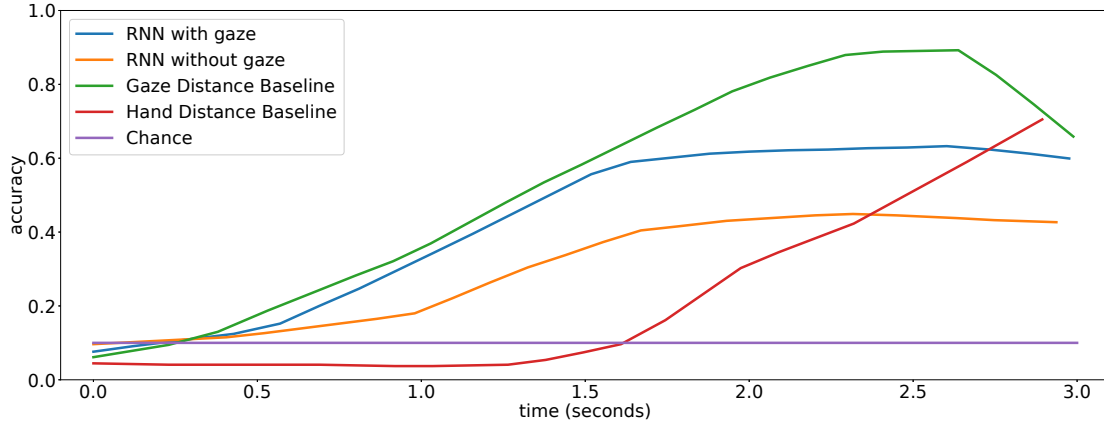


Figure 6.6: Gaze for Intent Prediction. Prediction accuracy for several baselines over a time-span of three seconds is shown.

Figure 6.5 shows an example trajectory for predicting motion to place a cup. The top row shows our method, the bottom row shows an uninformed prediction using the *initial* baseline. It can be seen that our method is very close to the ground truth, while the *initial* baseline predicts that the human moves forward slightly only and keeps position afterwards. This is because the original prediction is unaware of the human end position. Clearly, our method improves the prediction.

6.3 Gaze for Intent Prediction

In this section we investigate the advantage of using eye-gaze for intent prediction. Again, we use the MoGaze dataset (see section 7.2) and compare different intent prediction models with and without gaze features.

We extract grasping trajectories from MoGaze and divide the data into a test and a train set. In this case, We formulate intent prediction as a classification task of predicting the object the human is going to pick.

We compare the following methods:

- **RNN with gaze** is a recurrent neural network that uses gaze distance and human skeleton positions as input.
- **RNN without gaze** is a similar model but without the gaze features.
- **Gaze Distance Baseline** outputs the closest object to the gaze-ray

- **Hand Distance Baseline** outputs the closest object to the hand

Figure 6.6 depicts the prediction accuracy of the methods and the accuracy for selecting a random object (chance) on the test set. The accuracies are given over a time-span of three seconds with the grasp action occurring at second three. It can be seen that the *RNN without gaze* performs worse than the network with gaze features. The *Gaze Distance Baseline* outperforms other models that are based on both: human skeleton positions and gaze features. Note that the accuracy for the gaze baseline goes up to over 0.8 at around two seconds, which shows that the human usually looks at the object that he or she will grasp, and only in the beginning of the time period looks at other objects. This explains the good performance of the gaze baseline.

An interesting observation is that the gaze baseline performs strongest about half a second before a pick action happens. We assume that the human has finished to plan the grasp at that time and already starts to look at other objects he or she needs to interact with in the future, or at nearby objects on the table to avoid collisions.

The strong performance of the simple gaze baseline shows that leveraging gaze features for intent prediction is very promising. The strong performance of the gaze baseline on the MoGaze dataset is confirmed by [137].

6.4 Combining Hierarchical Motion Prediction with Task and Motion Planning

Considering scenarios, where human and robot not only share space, but additionally share a task, the robot needs to partially execute a task the human is busy with.

In such scenarios, maximal support from the robot while minimally intervening with the human is required. An example scenario can be seen in Figure 6.7. The human and the robot need to jointly setup the table.

In this section we introduce a system for combining hierarchical motion prediction with TAMP. Further details are available in our paper [18].

6.4.1 Hierarchical Motion Prediction

Human motion for a task naturally consists of multiple hierarchy levels. In our case, we abstract two levels. The low-level performs full-body motion prediction, the

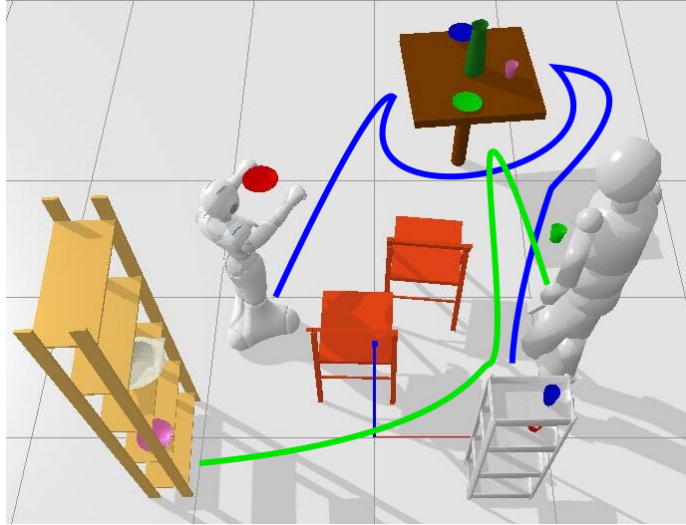


Figure 6.7: Pepper and a human jointly setup the table.

high-level is specifying high-level actions, such as pick or place of an object.

For the low-level policy we use a goal-conditioned PVRED model. For the high-level policy we use Maximum Entropy Inverse Reinforcement Learning (MaxEnt IRL) [138] to retrieve a policy.

MaxEnt IRL is based on state frequency calculations. An example of a high level trajectory for the MoGaze dataset can be seen in Table 6.2. Here, we defined the discretized state by the number of objects on a location and the human position as follows: $s^H = (\text{cups-table}, \text{cups-shelf1}, \text{cups-shelf2}, \text{plates-table}, \text{plates-shelf1}, \text{jugbowl-table}, \text{jugbowl-shelf1}, \text{jugbowl-shelf2}, \text{humanPos})$. The action space is discretized similarly.

The full prediction is obtained by combining the goals obtained from the high-level policy with the goal-conditioned full-body prediction model. In order to compute exact goals, we use heuristics, such as the closest point on the table not occupied by

Table 6.2: Example high-level trajectory

Start State	(0, 4, 0, 1, 0, 3, 1, 0, 1, 2)
Actions	Go to white shelf Pick up cup Go to table Place
End State	(1, 3, 0, 1, 0, 3, 1, 0, 1, 0)

Algorithm 6.1 Dynamic Task and Motion Planning

Input: states, goal set $\mathcal{S}_{\text{goal}}$
Deduce symbolic state s_0
Search for skeletons leading to goal $\Gamma_0(s_0, \mathcal{S}_{\text{goal}}, I)$
Set $\kappa = a_{0:k_0}^R \in \Gamma_0$ as best feasible skeleton
Execute current action of the skeleton κ
while $\mathcal{S}_{\text{goal}}$ not reached **do**
 Wait till Δt elapses, $t \leftarrow t + 1$
 Update system kinematics and human position
 Deduce current symbolic state s_t
 if κ is infeasible **then**
 Search $\Gamma_t(s_t, \mathcal{S}_{\text{goal}}, I)$
 Update $\kappa = a_{t:k_t}^{(R)} \in \Gamma_t$
 end if
 Optimize NLP of κ
 Execute current action of the skeleton κ
end while

an object.

6.4.2 Task and Motion Planning

For planning the robot motion, we also follow two hierarchies similar as we did for the human motion. We introduce a symbolic state s_t and a symbolic action a_t , at each timestep t . Human and robot have to perform actions in order to reach a goal state from the goal set $\mathcal{S}_{\text{goal}}$. In order to plan the high-level task and the low-level motions leading to the task, we use a method similar to LGP [132]. As the human prediction can deviate from the real motion, we introduce a dynamic version that is able to replan.

Algorithm 6.1 shows a high-level overview of the algorithm. First, possible symbolic paths leading to a goal state are generated using the Dijkstra algorithm and ranked by simple distance heuristics. We then define a NLP and solve it until a feasible solution is found, starting with the best skeleton. After that, the action for the current timestep is executed. At each timestep, the human predictions are updated and the feasibility is checked. If the skeleton becomes infeasible due to the human prediction, a new skeleton is searched.

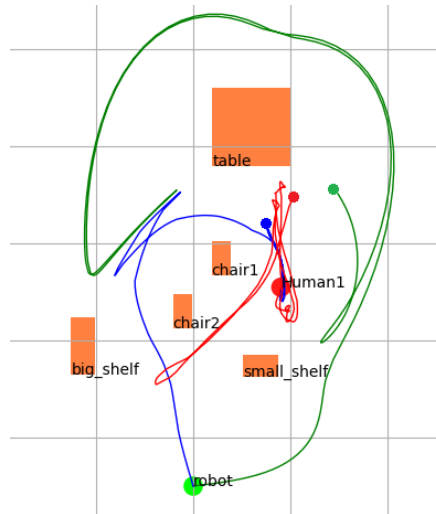


Figure 6.8: Example Human Robot Trajectory with TAMP. Trajectories of single planning (green), dynamic planning (blue) and actual human trajectory (red) on the workspace.

6.4.3 Experiments

We run experiments on our MoGaze dataset. We compare a single planning baseline, where the motion for the robot is planned only once, with our replanning algorithm.

Results can be seen in Table 6.3. The dynamic planning needs slightly more computation time. However, the results show that dynamic planning has higher success rates and produces shorter paths.

Figure 6.8 shows a comparison between trajectories. It can be seen that the dynamic mode plans a much shorter trajectory. This results from the replanning being able to use a path between the chair and the table that was blocked by the human beforehand.

Table 6.3: Dynamic LGP with Human Prediction

	Single planning	Dynamic planning
Success rate	91.2%	100%
Symbolic plan time	$0.0005 \pm 0.0001(sec)$	$0.0006 \pm 0.0002(sec)$
Task time reduction	0.298 ± 0.078	0.300 ± 0.100
Path ratio	1.000	0.626 ± 0.155
LGP replan count	-	3.0 ± 0.87

6.5 Summary

In this chapter we discussed possibilities for predicting longer time horizons. One possibility to improve prediction for longer horizons is the use of intent prediction. Thus, in the first part of this chapter, we addressed RQ4: “How can the intention of a human be estimated?”. We presented two approaches: an approach using ideas from affordances as well as an approach based on the idea of using eye-gaze. Our experiments show that both approaches are helpful for predicting intentions. Furthermore, we showed how the affordances approach can be combined with the full-body prediction from Chapter 4.

In the second part of this chapter, we tackled the follow up question RQ5: “Which challenges arise from longer time horizons and how can they be addressed for planning and prediction?”. For planning and prediction of long-term tasks, we proposed a hierarchical structure with a symbolic level. We showed an experiment where we combine TAMP and a hierarchical human motion prediction approach.

Datasets

As we focus on modeling human motion through a data-driven approach by learning a prediction of future human movements given previous human motion, the data used for training and evaluating the human model is a major component of the presented work. The focus of this thesis lies on robot assistance for humans in everyday scenarios. Thus, a dataset that reflects natural motions of humans as they would appear in everyday manipulation tasks is important.

For training a neural network motion model, as in Chapter 3, many coherent sequences of the same domain are preferred. Thus, the described dataset in this thesis focuses on such motion sequences. We concentrate on pick and place motions, as they are important in many everyday scenarios. Furthermore, we capture the objects in the scene and include the workspace geometry in the dataset. This enables us to develop methods for adapting a prediction to a goal constraint or avoid collisions with the environment, as we do in Chapter 4. Another topic of this thesis is intent prediction (Chapter 6). Related work suggests that eye-gaze is useful for intent prediction [120, 121]. In order to investigate that, we use an eye-tracker to capture the gaze direction of a human.

In this chapter we present the MoGaze dataset, a novel dataset that includes full-body motions of long coherent tasks, eye-gaze captured with a mobile eye-tracking device, and scene geometry. We explain the capturing procedure and focus on the design considerations and best practices for capturing motion datasets (RQ6).

Parts of this chapter are based on the following publication:

- Kratzer, P., Bihlmaier, S., Midlagajni, N. B., Prakash, R., Toussaint, M., and Mainprice, J. "Mogaze: A dataset of full-body motions that includes workspace geometry and eye-gaze". *Robotics and Automation Letters*, 6(2), 367-373.
© 2020 IEEE

7.1 Related Work on Datasets of Human Motion

For robots that share space with humans and want to successfully interact with them, the ability to understand and learn from human motion is an important key. The research topics of full-body motion prediction and intent prediction are getting more important in robotics research. Algorithms for motion or intent prediction, such as the ones used in this thesis, often rely on data-driven machine learning techniques. Those techniques require sophisticated datasets for training and evaluation.

Data Sets for Pedestrians

Many datasets target prediction or tracking of human motion in crowded spaces. For example, Robicquet et al. captured the Stanford Drone Dataset [139]. The Stanford Drone Dataset contains videos and images of various types of agents in different scenarios, such as a university campus or in the streets. A different dataset focused on autonomous robot navigation is the JRDB dataset by Martín-Martín et al. [140]. It includes data in human environments collected with a mobile manipulator. Simulating pedestrian crowds is an important topic for robot navigation in human scenarios. For example, Curtis et al. propose a framework for pedestrian simulation in crowds [141]. Similarly, Fan et al. proposed a navigation framework that specifically addresses problems for robot navigation in crowds [142].

Data Sets for Autonomous Driving

In the field of autonomous driving, there are also datasets recorded from the perspective of a moving car, such as [143, 144, 145, 146]. Those datasets are usually multi-modal and mainly do not contain human data. Datasets for pedestrians movement in crowded spaces and autonomous driving usually only focus on low-dimensional

pedestrian motion and do not contain full-body motion trajectories which are required for full-body tasks, such as pick and place motion.

Data Sets with Full-Body Motion

Full-body motion received a lot of attention from the computer graphics and computer vision communities. Related tasks in computer graphics and computer vision are generation of motion e.g. for animating characters, inferring human pose e.g. from video data, and forecasting human motion. A widely used dataset is the Carnegie Mellon University motion capture database (Mocap-CMU) [147]. Mocap-CMU provides data by a various number of actors that perform short motions, such as, dancing or jumping. Another commonly used dataset is the Human3.6m dataset recorded by Ionescu et al. [148]. The Human3.6m dataset is a large scale dataset with eleven individual actors that perform 17 scenarios, such as, discussion, smoking or taking photo. Another collection of human motion trajectories is the KIT Whole-Body Human Motion Database [149], which can be used with a unified representation [150]. This database contains a large collection of short motion trajectories and objects. A problem with those datasets is that they often only contain very short motions and motions that are “acted” and would not naturally occur in real world manipulation tasks. For human motion prediction in robotics long, coherent motions of real scenarios are more useful.

A more recent dataset by Marcard et al. contains three-dimensional human poses captured by using a single hand-held camera [151]. For body shape estimation, Ghorbani et al. presented a dataset, which contains motions as well as videos [152]. Maurice et al. captured a dataset with industry-related scenarios, such as screwing or manipulation loads [153].

An issue related to dataset representation is that motion data is usually captured using different systems and marker sets. Consequently, data across different datasets come with different parameterizations. Combining different sources in one learning algorithm is challenging. However, training networks on as many data as possible, is often desired in nowadays deep learning techniques. To tackle this issue, Mahmood et al. propose a dataset that aggregates several datasets into a common parameterization called AMASS [109].

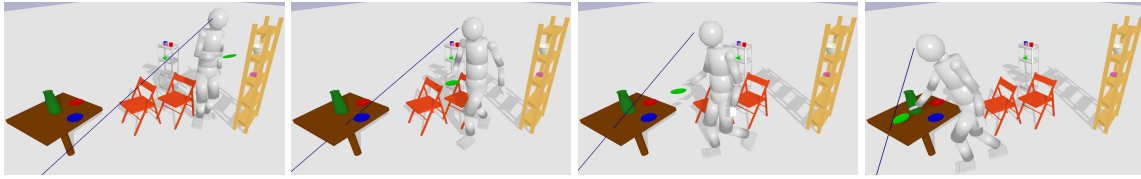


Figure 7.1: A human grasped a plate from the shelf and places it on the table. It can be observed how the gaze shifts towards the actual placing position even before the human reaches the table.

Summary

In this chapter we explain our own dataset, MoGaze, that was captured and used throughout this thesis. In contrast to other datasets, we aim to capture realistic, coherent whole-body human motion data of pick and place scenarios. In order to be applicable for robotics tasks, we additionally capture the objects in the scene. Moreover, we capture the eye-gaze of the human which is, for example, useful for intent prediction.

7.2 The *MoGaze* Full-Body Dataset

We present the *MoGaze* dataset, a dataset of full-body pick and place motions in table-setup and cleaning tasks [13]. The dataset is the first to our knowledge that includes both, full-body motion data and gaze.

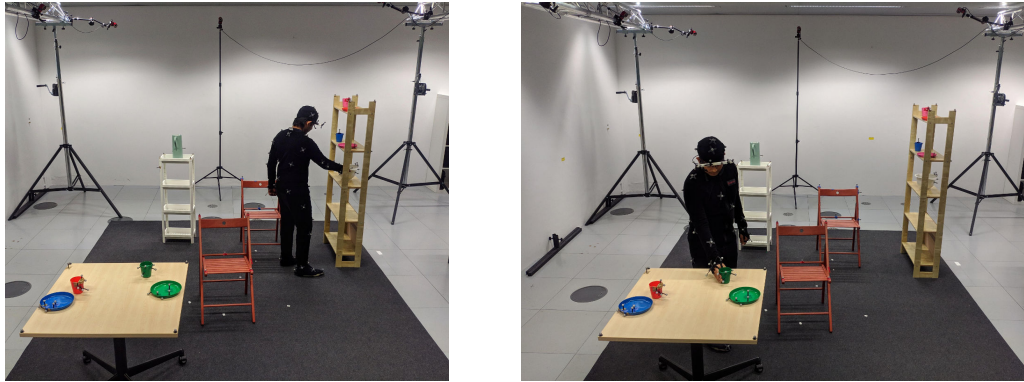
MoGaze contains 1627 pick and place actions performed within 180 minutes of motion capture data of seven different actors. It contains full-body motion data of seven actors and eye-gaze data for six of the actors. The data includes multiple objects, such as chairs, tables, shelves, plates, and cups.

An example trajectory of the dataset can be seen in Figure 7.1. The human model (depicted gray) starts in front of the shelf and walks towards the table in order to place a plate (depicted green).

7.2.1 Setup for Recording

For tracking the human body and the objects, we use an OptiTrack¹ motion capture system. The system consists of twelve cameras tracking an area of approximately three by four meters. The setup can be seen in Figure 7.2.

¹<https://www.optitrack.com/>



(a) Human placing a bowl

(b) Human grasping a cup

Figure 7.2: Dataset capture setup. The scene with the human and multiple objects is seen.



Figure 7.3: Eye-tracker with an attachment for placing additional motion capture markers.

In order to capture the human movement, the participants wore a motion capture suit. We attached 50 reflecting markers to the motion capture suit following the full-body tracking marker convention of the Motive software². Consequently, we are able to capture full-body motion trajectories for the human skeleton.

In order to obtain motion data for the objects in the scene, we place between four and six reflective markers on each object. This allows tracking the position and orientation of the objects as rigid bodies.

For eye tracking, we use a wearable headset from PupilLabs [154]. We add a removable attachment to the headset and place motion capture markers on it (see Figure 7.3). This allows accurate tracking of its position and orientation in world

²<https://www.optitrack.com/products/motive/>

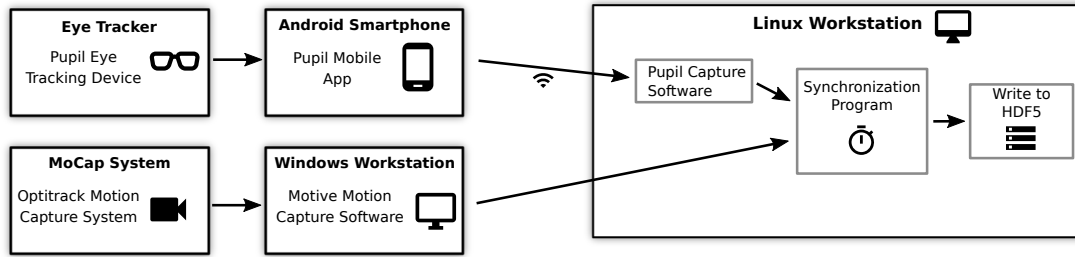


Figure 7.4: Overview over the data capture setup. The eye tracker is connected to a smartphone which streams the data to a Linux workstation via Wi-Fi. This enables the human to freely walk in the area.

coordinates. The eye tracker uses two eye cameras pointing inwards to estimate the pupils' motion and a world camera to calibrate the estimated gaze direction with respect to the headset.

Figure 7.4 shows a schematic overview of the data capture system. The eye tracker is connected to an Android smartphone via USB-C. The Android smartphone runs the *Pupil Mobile App* from PupilLabs. The app uses Wi-Fi to send the video of the eye trackers' cameras to the Pupil Capture Software running on a Linux computer. Due to the use of Wi-Fi, the human is not hindered by an Ethernet cable and can freely walk in the scene.

The motion capture system is connected to a Windows workstation which runs the Motive motion capture software. The Windows machine streams the motion data to the Linux workstation via Ethernet. The frames are stamped with the clock of the Linux workstation. The mean delay between the Windows machine and the Linux machine is 0.00055 seconds, which is negligible. Furthermore, using scripts provided by PupilLabs, the eye-tracker's internal clock is synchronized with the Linux workstation. This way, we account for the longer delay through streaming video data from the smartphone to the Linux workstation via Wi-Fi.

The gaze data is captured with a frequency of 200Hz and the motion data is captured with a frequency of 120Hz. When multiple gaze data frames correspond to one motion capture frame, we use the frame with a higher reported confidence value from the Pupil software.

On the Linux machine we run programs for recording the data. The live data with the corresponding timestamps is written to HDF5 files.

7.2.2 Calibration of Motion Capture and Eye-Tracker

For calibrating the individual systems, routines by the manufacturers are provided. For the motion capture system, markers in a predefined distance are mounted on a calibration wand. The wand is moved inside the motion capture volume till enough samples are taken. The individual cameras track the markers and a routine can compute their relative positions from the recorded samples.

The calibration of the eye-tracker works by using visual markers. The participants are advised to look on a specific marker which additionally is tracked by the world camera of the headset. The eye cameras of the headset track the pupils position. The marker is moved in the visual field of the participant. After enough samples, the software can map the pupils position to a gaze point in the image plane.

Relative Position of the Eye Tracker in World Coordinates

We use markers on the headset's world camera in order to track the global coordinates of the eye tracker (see Figure 7.3). However, we need to find the relative offset between the estimated pose of the headset obtained from the motion capture system and the true pose of the camera. To minimize errors, we directly calibrate the gaze of the participant in global coordinates. This is done by letting the participant look at a predefined motion capture marker from different angles. Using the gaze ray obtained from the eye tracker and the pose obtained from the motion capture system, we use a least squares fit to calculate the position and rotation offset of the headset's world camera.

7.2.3 The Human Model

The human data is modeled using a simple kinematic model with 21 joints. The joints can be seen in Table 7.1. The base joint has a three-dimensional translation and a three-dimensional orientation to offset the human model from the world frame.

Table 7.1: The joints of the kinematic model of the human that we used.

base	pelvis	torso	neck	head	rToe
lElbow	lWrist	rinnerShoulder	rShoulder	rElbow	rWrist
lKnee	lAnkle	lToe	rHip	rKnee	rAnkle
lShoulder	lHip	linnerShoulder			

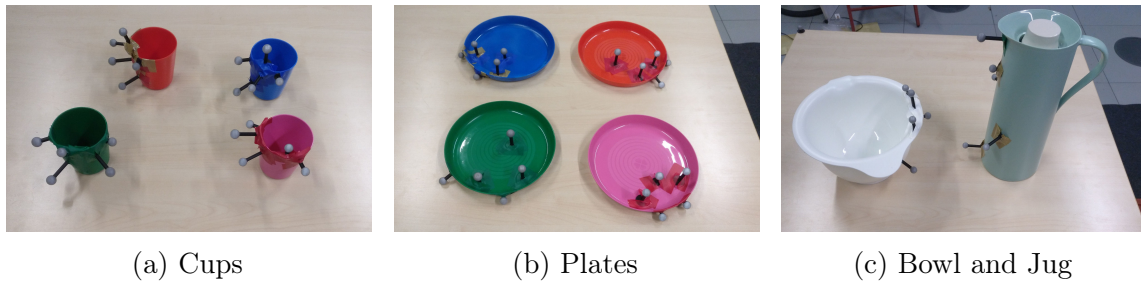


Figure 7.5: Movable objects in the MoGaze dataset. Each of the objects has several motion capture markers placed on it.

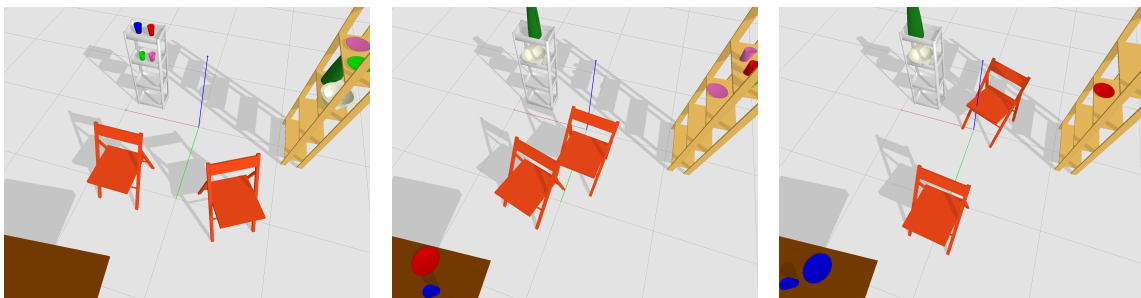


Figure 7.6: Different configurations of the chairs

The other joints also have translations and rotations to offset from the parent joint. However, the translations only change for different humans and are kept fixed during a recording session. Those are used for scaling the model to different humans by adapting the sizes, for example, the length of the forearm.

In total, the human data has 66 degrees of freedom. For rotation of the 21 joints, 63 degrees of freedom are used, three more degrees of freedom are used for the base translations.

7.2.4 Objects

The scene contains the following objects: 4 plates and 4 cups (colored red, green, blue and pink), 1 jug, 1 bowl, 2 chairs, a big shelf, a small shelf, and a table. We measured the dimensions of the objects and created triangular meshes for visualization of the objects in a simulator.

During the recording, the table, the big shelf, and the small shelf were kept on fixed locations. The other objects were moved around by participants. The movable objects are the cups, the plates, the bowl, and the jug and can be seen in Figure 7.5.

The chairs are treated as special objects. The participants were instructed to change the positions of the chairs three times throughout the recording. This way, the chairs serve as obstacles that prevent participants from always taking the same paths. As a result, more variations of motions can be collected.

The configurations can be seen in Figure 7.6. The first and the last configuration allow the human to walk in between them. In the second configuration the chairs are placed close to each other so that the human is not able to walk in between. The same three configurations are used for each participant. We chose three configurations because we found that it is a good trade-off between getting more variation into the walking behavior and having sufficient data per configuration.

7.2.5 Tasks

To achieve varying motion trajectories, the participants were instructed to perform simple manipulation tasks. A full list of the tasks can be seen in Table 7.2. The tasks include table setup tasks for a specific number of people, clearing the table, and putting a specific group of objects to a specific shelf.

Table 7.2: Instructions for participant tasks. The used instructions can be seen, paired with the probability of the task being assigned to the participant

Probability	Task
0.081	Set the table for 1 person
0.081	Set the table for 2 persons
0.081	Set the table for 3 persons
0.081	Set the table for 4 persons
0.032	Clear table
0.161	Put the jug and the bowl on small shelf
0.161	Put all cups on small shelf
0.032	Put blue and pink objects on big shelf
0.032	Put blue and red objects on big shelf
0.032	Put blue and green objects on big shelf
0.032	Put pink and red objects on big shelf
0.032	Put pink and green objects on big shelf
0.032	Put red and green objects on big shelf
0.032	Put all cups on big shelf
0.032	Put bowl and jug on big shelf
0.032	Put all cups on big shelf
0.032	Put all plates on big shelf

For the table setup tasks, the participant had to place a plate and a cup for each person, and either the jug or the bowl, on the table. For clearing the table the participant had to move all objects away from the table in order of it to be empty. The other tasks directly specify which objects have to be moved and to which location.

In order to better distribute the locations to which objects are moved, a probability was added to each of the tasks (see Table 7.2). This way, moving objects to the big shelf, moving object to the small shelf, and moving objects to the table, is selected with equal probability. This ensures that the objects are more equally distributed in the area.

We use a simple instruction program that runs on the Linux workstation. The program randomly selects a task according to the corresponding probabilities. The program shows the task to the conductor of the experiment who decides if the task is possible with the current object locations. A text-to-speech system is used to read the task out to the participant after it is confirmed by the conductor.

The instruction program stores the tasks during data capturing and stamps it with the current time. This allows knowledge of which task was performed at which timestep.

7.2.6 Recording Procedure

First, the participants received a short introduction on the capture setup and the aim of the dataset. Furthermore, the tasks were explained in detail. We advised the participants to only move one object at a time and to always use the same hand.

After the introduction, the participants put on the motion capture suit and we ensured that the reflective markers were adjusted correctly. The participant put on the eye-tracker and we connected it to the smartphone. As soon as everything was connected, we started the calibration routine for calibrating the eye-tracker with the motion capture system. Before starting the main capture session, we performed a short test recording to get the participants used to the capture setup and the tasks. In case of recording issues the session was interrupted and resumed.

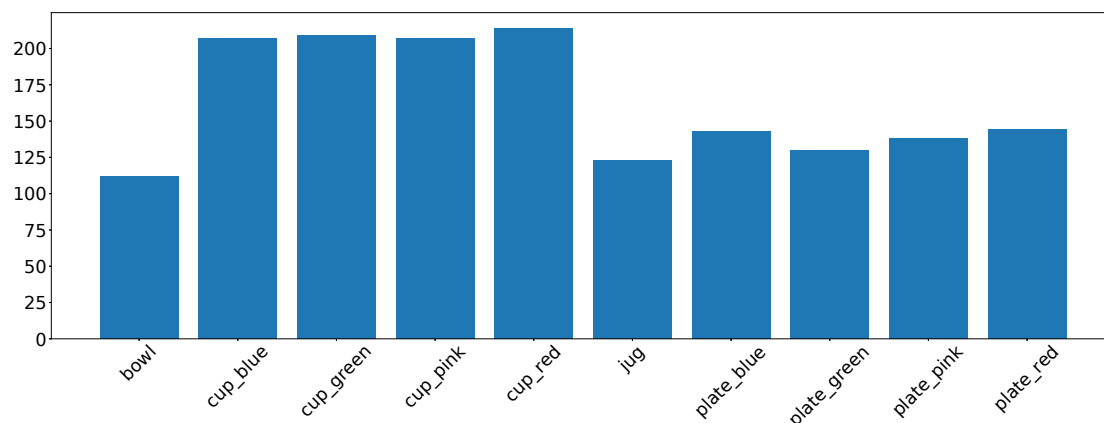


Figure 7.7: Number of pick and place actions in the dataset per object.

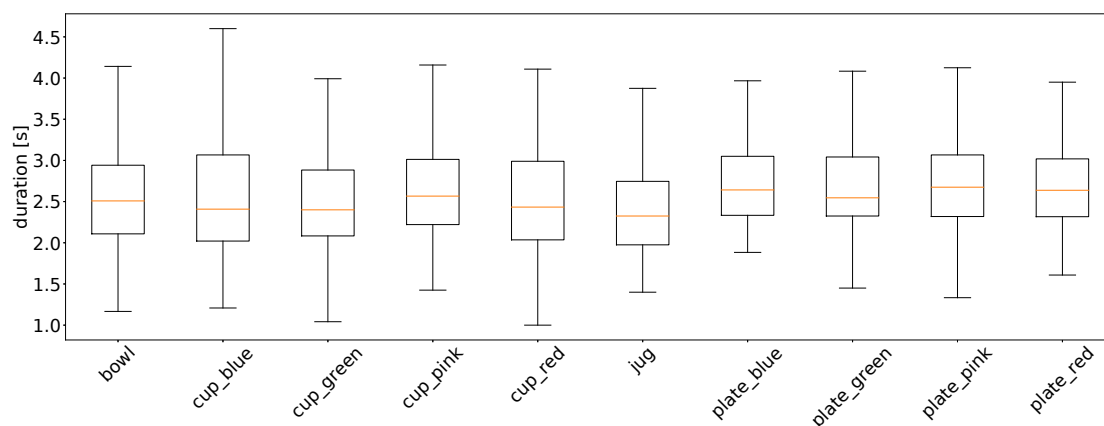


Figure 7.8: Duration of a placing action per object, from object is grasped till object is placed.

7.2.7 Data Labeling

For a number of applications of motion datasets, such as intent prediction, it is interesting to be able to obtain specific trajectories, such as trajectories for picking or placing movements. Thus, we automatically label the data by computing the start and stop point of object movements. Typically, when an object starts moving, a grasp action has been performed by the human, and when the movement stops, a place action has been performed.

The labeling is computed by comparing the positions of the object to previous timesteps. If the object distance between two timesteps is larger than a threshold, the object is labeled as moving. We validated the labeling by a human expert.

In total MoGaze contains 1627 pick and place actions. The number of pick and place actions per object can be seen in Figure 7.7. Most often the cups have been moved which makes sense since they occur in most of the tasks. The least often moved object is the bowl. All objects have been moved more than 100 times.

Figure 7.8 depicts the duration of the grasping action, which is the duration from when an object is picked until it is placed. It can be seen that the timespan of an action is usually between two and three seconds. This holds for all the objects in the dataset.

7.2.8 Discussion and Limitations

Our dataset is a sophisticated dataset with full-body motion, eye-gaze, and scene geometry. It has been successfully used for many experiments during this thesis. Furthermore, it has already been used by other research groups [137, 155, 156].

A challenge with capturing motion data is the setup of the cameras. While the cameras were adjusted to cover the full scene, due to the various number of objects and shelves, occlusions may rarely occur. This can lead to small jumps of objects in the MoGaze dataset. The effect of occlusions is even higher when multiple humans are recorded at the same time. Thus, we only recorded preliminary data with two humans with our setup (see section A.1.1).

Another limitation with the specific setup of simultaneously capturing full-body motion and eye-gaze, is that the eye tracker needs to be calibrated for a specific distance to the eyes. Opposed to many other applications for eye tracking, the human is moving in the scene so that the distance between eyes and objects changes. We calibrated the eye-tracker for a distance of approximately one meter because we found that this is a good compromise for our tasks.

7.3 Summary

In this chapter we presented a sophisticated dataset that has been created during this thesis. We discussed the considerations, formats and recording procedures, important for creating human motion datasets (RQ6). The MoGaze dataset is a full-body datasets which aims to close gaps of previous human full-body motion dataset, which often only contain very short motions without scene context, and thus are not ideal for motion prediction during human-robot interaction. Instead, the MoGaze dataset

provides long sequences of everyday pick and place motions with scene geometry and eye gaze.

Additional datasets we used in this thesis are explained in section A.1 and contain walking motion and synthetic data for testing with lower dimensional examples.

Conclusions

In this thesis we investigated the use of predictive models for human motion with the aim of a human-aware framework for planning robot motion in a HRC setting. We first investigated data-driven predictive models that are able to forecast an observed motion trajectory (RQ1, see Chapter 3). We analyzed several approaches and found that RNNs can work well with large datasets. We compared different modifications of the RNN architecture, a basic network, a network with residual connection, and a network with additional velocity connections and an augmented state. Our experiments showed that the network with velocity connections works best. Furthermore, we compared different rotation representations. We found that a continuous six-dimensional representation outperforms networks with exponential map representations, which have been state-of-the-art.

While forecasting of an observed motion is an important topic, we identified that in many scenarios, there is the need to be able to constrain a prediction to external criteria. Constraining the prediction of a predictive model while retaining the predictive capabilities of still finding a likely trajectory, leads us to the notion of a controllable and predictive human model (RQ2, see Chapter 4). In this thesis, a novel framework for controlling the predictive model by only slightly changing the model inputs, was proposed. We added control parameters to the input states of the decoder of the RNN for motion prediction and we formulated the prediction problem as a NLP with the control parameters as decision variables. This made it possible to use gradient-based optimization to solve the NLP while accounting for additional user specified constraints. For example, such constraints can arise from the environment,

the underlying task, or an intention prediction model. We conducted experiments on real motion data with goal-set constraints and constraints for collision avoidance. The experiments showed that our method is able to change predictions while retaining the predictive capabilities through minimizing the change made to the neural network inputs. Furthermore, we found that our method significantly improves over the pure human prediction without such environmental constraints. Thus, our method can be used to control the trajectories produced by a predictive human model.

For HRC, we want to plan human-aware robot motion. For this reason, we proposed to incorporate the predictive human model into motion planning (RQ3, see Chapter 5). We identified the cyclic dependency of predicting human motion and planning robot motion as a challenge, since the human would adapt his behavior dependent on what the robot is doing and vice-versa. Thus, we proposed an extension of our controllable motion prediction framework that incorporates two agents, a human and a robot. This novel formulation allows incorporating coordination constraints that act on both agents simultaneously, such as handovers or coordinated collision avoidance. Our experiments on handovers, collision avoidance, and pick or place tasks showed that our framework is able to plan coordinated motion, where both, human and robot, adapt to each other. Moreover, in a comparison to baselines, our method showed superior distribution of the paths between human and robot, with a possibility to influence the strength an agent is adapting to the other by using hyperparameters.

In the field of long-term motion prediction, the challenges of multi-modality and uncertainty about the human’s future goal make accurate predictions very difficult. In order to improve predictions for longer horizons, we looked into possibilities for intention prediction (RQ4, see Chapter 6). We discussed two approaches, the possibility to incorporate object related interaction capabilities, so called object affordances, in order to predict human intention, and the possibility to incorporate eye-gaze. Both approaches showed promising results. In experiments with eye-gaze we found that the gaze is a very strong indicator for identifying which object a human is going to pick. We integrated an intention prediction module based on affordances within our controllable human prediction model and showed that this improves the prediction performance of the model.

Planning robot motion for long task sequences is challenging (RQ5, see Chapter 6). We introduced a hierarchical structure with two levels. The high level introduces

a symbolic abstraction which contains high-level actions, corresponding to which object should be picked or where it should be placed. The low level is the full-body prediction of the human, and the motion trajectory for the robot, respectively. We proposed an algorithm for hierarchical planning of human motion and used TAMP to coordinate a robot trajectory accordingly.

A very central part of human predictive models is the underlying dataset for motion prediction. Since many existing datasets did not contain long motion sequences of pick and place tasks – scenarios that are required for robotics, we captured our own dataset (RQ6, see Chapter 7). Our dataset contains 180 min of motion capture data with 1627 pick and place actions. Additionally, we captured the eye-gaze of the participants using mobile eye-tracking glasses. Experiments with our dataset throughout this thesis show the high value of the dataset for the motion prediction experiments in the context of HRC.

8.1 Future Work

This thesis lays a solid foundation for understanding the use of modern, data-driven human predictive models in robot planning. We want to highlight our novel framework that allows to adapt the human prediction to external criteria and enables co-optimizing human and robot motion.

For the underlying model of human motion we see room for potential improvements on the side of the dataset. Modern deep learning methods, such as the RNNs used in this thesis, scale well to extremely large dataset. At the same time, data-driven methods are limited to predict motion that is similar to those of the dataset. While our captured dataset contains a wide range of full-body, pick and place trajectories, not all possible interactions are covered, obviously. Future work on dataset creation can accumulate multiple datasets, investigate a wider range of interactions, and even cover human-human or human-robot interaction within the dataset. We started to extend our data with human-human data (see section A.1.1) but have been limited by hardware capabilities.

For our human-aware planning framework for robot motion, we think that the next step is to take it to real world human-robot interaction. Challenges on the real robot require the ability to obtain the human state from sensors, the ability to replan when the human motion deviates from the prediction, and the ability to

8. CONCLUSIONS

run the algorithms in real time which needs computation time optimized algorithms. While making a big step towards human-aware motion planning, a holistic real robot human-aware framework for human-robot interaction is out-of-scope for this thesis.

We finally demonstrated our work on long-term motion prediction and planning. Human movement for tasks that include many motion sequences, such as pick and place sequences, are challenging to predict since there are multiple options, and multiple execution orders of action sequences that can lead to the same goal. We think that long term motion prediction is a promising research field. Future work in this area could potentially improve the consideration of the multi-modality of human motion.

Bibliography

- [1] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015.
- [2] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. In *Robotics: Science and Systems (RSS)*, 2017.
- [3] IFR presents world robotics 2021 reports. <https://ifr.org/ifr-press-releases/news/robot-sales-rise-again>, 2021.
- [4] Richard M Scheffler, James Campbell, Giorgio Cometto, Akiko Maeda, Jenny Liu, Tim A Bruckner, Daniel R Arnold, and Tim Evans. Forecasting imbalances in the global health labor market and devising policy responses. *Human Resources for Health*, 16(1):1–10, 2018.
- [5] Warnung vor dramatischem Personalmangel. <https://www.tagesschau.de/inland/aerztetag-warnung-personalmangel-101.html>, 2021.
- [6] Shirine El Zaatari, Mohamed Marei, Weidong Li, and Zahid Usman. Cobot programming for collaborative industrial tasks: An overview. *Robotics and Autonomous Systems*, 116:162–180, 2019.
- [7] Julieta Martinez, Michael J Black, and Javier Romero. On human motion prediction using recurrent neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

- [8] Dario Pavlo, David Grangier, and Michael Auli. Quaternet: A quaternion-based recurrent model for human motion. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2018.
- [9] Daniel Holden, Taku Komura, and Jun Saito. Phase-functioned neural networks for character control. *ACM Transactions on Graphics*, 36(4), 2017.
- [10] Philipp Kratzer, Marc Toussaint, and Jim Mainprice. Towards combining motion optimization and data driven dynamical models for human motion prediction. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 202–208, 2018.
- [11] Philipp Kratzer, Marc Toussaint, and Jim Mainprice. Prediction of human full-body movements with motion optimization and recurrent neural networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1792–1798, 2020.
- [12] Philipp Kratzer, Niteesh Balachandra Midlagajni, Marc Toussaint, and Jim Mainprice. Anticipating human intention for full-body motion prediction in object grasping and placing tasks. In *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 1157–1163, 2020.
- [13] Philipp Kratzer, Simon Bihlmaier, Niteesh Balachandra Midlagajni, Rohit Prakash, Marc Toussaint, and Jim Mainprice. Mogaze: A dataset of full-body motions that includes workspace geometry and eye-gaze. *IEEE Robotics and Automation Letters*, 6(2):367–373, 2020.
- [14] Philipp Kratzer, Marc Toussaint, and Jim Mainprice. Planning coordinated human-robot motions with neural network full-body prediction models. *arXiv preprint arXiv:2210.13317*, 2022.
- [15] Simon Tobias Bihlmaier. Gaze based intent prediction for human robot collaboration using neural networks. B.S. thesis, University of Stuttgart, 2019.
- [16] Niteesh Balachandra Midlagajni. Learning object affordances using human motion capture data. Master’s thesis, University of Stuttgart, 2019.

- [17] Simon Hagenmayer. Hierarchical adversarial imitation learning from motion capture data. Master’s thesis, University of Stuttgart, 2020.
- [18] An T Le, Philipp Kratzer, Simon Hagenmayer, Marc Toussaint, and Jim Mainprice. Hierarchical human-motion prediction and logic-geometric programming for minimal interference human-robot tasks. In *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 7–14, 2021.
- [19] Philipp Kratzer. Robot assistance for collaborative task execution. Master’s thesis, University of Stuttgart, 2017.
- [20] Ruth Schulz, Philipp Kratzer, and Marc Toussaint. Building a bridge with a robot: a system for collaborative on-table task execution. In *Proceedings of the International Conference on Human Agent Interaction (HAI)*, pages 399–403, 2017.
- [21] Ruth Schulz, Philipp Kratzer, and Marc Toussaint. Preferred interaction styles for human-robot collaboration vary over tasks with different action types. *Frontiers in Neurorobotics*, 12:36, 2018.
- [22] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.
- [23] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.
- [24] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- [25] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91(1), 1991.
- [26] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

- [27] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1310–1318. PMLR, 2013.
- [28] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- [29] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.
- [30] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.
- [31] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.
- [32] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 27, 2014.
- [33] Kai Henning Koch, Katja Mombaur, and Philippe Soueres. Optimization-based walking generation for humanoid robot. In *Proceedings of the IFAC International Symposium on Robot Control (SYROCO)*. Elsevier, 2012.
- [34] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 489–494, 2009.
- [35] Oskar Von Stryk and Roland Bulirsch. Direct and indirect methods for trajectory optimization. *Annals of Operations Research*, 37(1):357–373, 1992.
- [36] John T Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, 1998.

-
- [37] Matthew Kelly. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, 59(4):849–904, 2017.
- [38] Marc Toussaint. A tutorial on newton methods for constrained trajectory optimization and relations to slam, gaussian process smoothing, optimal control, and probabilistic inference. In *Geometric and Numerical Foundations of Movements*, pages 361–392. Springer, 2017.
- [39] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.
- [40] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.
- [41] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Dariu M Gavrila, and Kai O Arras. Human motion trajectory prediction: A survey. *The International Journal of Robotics Research*, page 0278364920917446, 2019.
- [42] Arthur D Kuo and J Maxwell Donelan. Dynamic principles of gait and their clinical implications. *Physical Therapy*, 90(2):157–174, 2010.
- [43] Friedl De Groote and Antoine Falisse. Perspective on musculoskeletal modelling and predictive simulations of human movement to assess the neuromechanics of gait. *Proceedings of the Royal Society B*, 288(1946):20202432, 2021.
- [44] Danny Driess, Heiko Zimmermann, Simon Wolfen, Dan Suissa, Daniel Haeufle, Daniel Hennes, Marc Toussaint, and Syn Schmitt. Learning to control redundant musculoskeletal systems with neural networks and sqp: exploiting muscle properties. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 6461–6468, 2018.
- [45] Christian Beyaert, Rajul Vasa, and Gunilla E Frykberg. Gait post-stroke: pathophysiology and rehabilitation strategies. *Neurophysiologie Clinique/Clinical Neurophysiology*, 45(4-5):335–355, 2015.
- [46] Bastien Berret, Enrico Chiovetto, Francesco Nori, and Thierry Pozzo. Evidence for composite cost functions in arm movement planning: an inverse optimal control approach. *PLoS computational biology*, 7(10), 2011.

- [47] Jim Mainprice, Rafi Hayne, and Dmitry Berenson. Goal set inverse optimal control and iterative replanning for predicting human reaching motions in shared workspaces. *IEEE Transactions Robotics*, 32(4):897–908, 2016.
- [48] Maren Bennewitz, Wolfram Burgard, Grzegorz Cielniak, and Sebastian Thrun. Learning motion patterns of people for compliant robot motion. *The International Journal of Robotics Research*, 24(1):31–48, 2005.
- [49] Dana Kulić, Christian Ott, Dongheui Lee, Junichi Ishikawa, and Yoshihiko Nakamura. Incremental learning of full body motion primitives and their sequencing through human motion observation. *The International Journal of Robotics Research*, 31(3):330–345, 2012.
- [50] Jos Elfring, René Van De Molengraft, and Maarten Steinbuch. Learning intentions for improved human motion prediction. *Robotics and Autonomous Systems*, 62(4):591–602, 2014.
- [51] Andreas M Lehrmann, Peter V Gehler, and Sebastian Nowozin. Efficient nonlinear markov models for human motion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1314–1321, 2014.
- [52] Hema S Koppula and Ashutosh Saxena. Anticipating human activities using object affordances for reactive robotic response. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):14–29, 2016.
- [53] Aaron P Shon, Keith Grochow, and Rajesh PN Rao. Robotic imitation from human motion capture using gaussian processes. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 129–134, 2005.
- [54] Carl Henrik Ek, Philip HS Torr, and Neil D Lawrence. Gaussian process latent variable models for human pose estimation. In *International Workshop on Machine Learning for Multimodal Interaction*, pages 132–143. Springer, 2007.
- [55] Jack M Wang, David J Fleet, and Aaron Hertzmann. Gaussian process dynamical models for human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):283–298, 2008.

-
- [56] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4346–4354, 2015.
- [57] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5308–5317, 2016.
- [58] Partha Ghosh, Jie Song, Emre Aksan, and Otmar Hilliges. Learning human motion models for long-term predictions. In *Proceedings of the International Conference on 3D Vision (3DV)*, pages 458–466, 2017.
- [59] Liang-Yan Gui, Yu-Xiong Wang, Xiaodan Liang, and José MF Moura. Adversarial geometry-aware human motion prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 786–803, 2018.
- [60] Dario Pavlo, Christoph Feichtenhofer, Michael Auli, and David Grangier. Modeling human motion with quaternion-based neural networks. *International Journal of Computer Vision*, pages 1–18, 2019.
- [61] Hongsong Wang, Jian Dong, Bin Cheng, and Jiashi Feng. Pvrred: A position-velocity recurrent encoder-decoder for human motion prediction. *IEEE Transactions on Image Processing*, 30:6096–6106, 2021.
- [62] Maosen Li, Siheng Chen, Yangheng Zhao, Ya Zhang, Yanfeng Wang, and Qi Tian. Dynamic multiscale graph neural networks for 3d skeleton based human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 214–223, 2020.
- [63] Emre Aksan, Manuel Kaufmann, Peng Cao, and Otmar Hilliges. A spatio-temporal transformer for 3d human motion prediction. In *Proceedings of the International Conference on 3D Vision (3DV)*, pages 565–574. IEEE, 2021.
- [64] Du Q Huynh. Metrics for 3d rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35(2):155–164, 2009.
- [65] F Sebastian Grassia. Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools*, 3(3):29–48, 1998.

- [66] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5745–5753, 2019.
- [67] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.
- [68] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in Neural Information Processing Systems*, 28, 2015.
- [69] Zimo Li, Yi Zhou, Shuangjiu Xiao, Chong He, Zeng Huang, and Hao Li. Auto-conditioned recurrent networks for extended complex human motion synthesis. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.
- [70] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. *ACM Transactions on Graphics*, 21(3):473–482, jul 2002.
- [71] Jehee Lee, Jinxiang Chai, Paul SA Reitsma, Jessica K Hodgins, and Nancy S Pollard. Interactive control of avatars animated with human motion data. In *Proceedings of the ACM SIGGRAPH annual Conference on Computer graphics and Interactive Techniques (SIGGRAPH)*, pages 491–500, 2002.
- [72] Manfred Lau and James J Kuffner. Behavior planning for character animation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 271–280, 2005.
- [73] Alla Safonova and Jessica K Hodgins. Construction and optimal search of interpolated motion graphs. *ACM Transactions on Graphics*, 26(3):106, 2007.
- [74] Stelian Coros, Philippe Beaudoin, Kang Kang Yin, and Michiel van de Panne. Synthesis of constrained walking skills. *ACM Transactions on Graphics*, 27(5):113, 2008.
- [75] Libin Liu, KangKang Yin, Michiel van de Panne, Tianjia Shao, and Weiwei Xu. Sampling-based contact-rich motion control. *ACM Transactions on Graphics*, 29(4), 2010.

-
- [76] Jehee Lee and Kang Hoon Lee. Precomputing avatar behavior from human motion data. *Graphical models*, 68(2):158–174, 2006.
- [77] Sergey Levine, Jack M Wang, Alexis Haraux, Zoran Popović, and Vladlen Koltun. Continuous character control with low-dimensional embeddings. *ACM Transactions on Graphics*, 31(4):1–10, 2012.
- [78] Xue Bin Peng, Glen Berseth, and Michiel Van de Panne. Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Transactions on Graphics*, 35(4):1–12, 2016.
- [79] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics*, 37(4):1–14, 2018.
- [80] Jinxiang Chai and Jessica K Hodgins. Constraint-based motion optimization using a statistical dynamic model. *ACM Transactions on Graphics*, 26(3):8–es, 2007.
- [81] Igor Mordatch, Emanuel Todorov, and Zoran Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics*, 31(4):1–8, 2012.
- [82] Igor Mordatch, Jack M Wang, Emanuel Todorov, and Vladlen Koltun. Animating human lower limbs using contact-invariant optimization. *ACM Transactions on Graphics*, 32(6):1–8, 2013.
- [83] Philipp Kratzer, Marc Toussaint, and Jim Mainprice. Motion prediction with recurrent neural network dynamical models and trajectory optimization. In *Workshop on AI and Its Alternatives in Assistive and Collaborative Robotics*, 2019.
- [84] Iain S Duff. Ma57—a code for the solution of sparse symmetric definite and indefinite systems. *ACM Transactions on Mathematical Software*, 30(2):118–144, 2004.
- [85] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *Proceedings of the*

- USENIX Symposium on Operating Systems Design and Implementation*, pages 265–283, 2016.
- [86] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian process for machine learning*. MIT press, 2006.
- [87] Dieter Kraft. Algorithm 733: Tomp–fortran modules for optimal control calculations. *ACM Transactions on Mathematical Software*, 20(3):262–281, 1994.
- [88] Maruan Al-Shedivat et al. Learning scalable deep kernels with recurrent structure. *The Journal of Machine Learning Research*, 18(1):2850–2886, 2017.
- [89] Przemyslaw A. Lasota, Terrence Fong, and Julie A. Shah. *A survey of methods for safe human-robot interaction*. Now Publishers, 2017.
- [90] Arash Ajoudani, Andrea Maria Zanchettin, Serena Ivaldi, Alin Albu-Schäffer, Kazuhiro Kosuge, and Oussama Khatib. Progress and prospects of the human–robot collaboration. *Autonomous Robots*, 42(5):957–975, 2018.
- [91] Jimmy Baraglia, Maya Cakmak, Yukie Nagai, Rajesh Rao, and Minoru Asada. Initiative in robot assistance during collaborative task execution. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 67–74, 2016.
- [92] Przemyslaw A Lasota and Julie A Shah. Analyzing the effects of human-aware motion planning on close-proximity human–robot collaboration. *Human Factors*, 57(1):21–33, 2015.
- [93] Dana Kulić and Elizabeth Croft. Pre-collision safety strategies for human-robot interaction. *Autonomous Robots*, 22(2):149–164, 2007.
- [94] Jim Mainprice and Dmitry Berenson. Human-robot collaborative manipulation planning using early prediction of human motion. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots And Systems (IROS)*, pages 299–306, 2013.
- [95] Przemyslaw A Lasota, Gregory F Rossano, and Julie A Shah. Toward safe close-proximity human-robot interaction with standard industrial robots. In

-
- Proceedings of the IEEE International Conference on Automation Science and Engineering (CASE)*, pages 339–344, 2014.
- [96] Emrah Akin Sisbot and Rachid Alami. A human-aware manipulation planner. *IEEE Transactions Robotics*, 28(5):1045–1057, 2012.
- [97] Thibault Kruse, Amit Kumar Pandey, Rachid Alami, and Alexandra Kirsch. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 61(12):1726–1743, 2013.
- [98] Niclas Evestedt, Erik Ward, John Folkesson, and Daniel Axehill. Interaction aware trajectory planning for merge scenarios in congested traffic situations. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 465–472, 2016.
- [99] Tirthankar Bandyopadhyay, Kok Sung Won, Emilio Frazzoli, David Hsu, Wee Sun Lee, and Daniela Rus. Intention-aware motion planning. In *Algorithmic Foundations of Robotics X*, pages 475–491. Springer, 2013.
- [100] David Fridovich-Keil, Andrea Bajcsy, Jaime F Fisac, Sylvia L Herbert, Steven Wang, Anca D Dragan, and Claire J Tomlin. Confidence-aware motion prediction for real-time collision avoidance¹. *The International Journal of Robotics Research*, 39(2-3):250–265, 2020.
- [101] Emanuel Todorov and Weiwei Li. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proceedings of the American Control Conference (ACC)*, pages 300–306, 2005.
- [102] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4569–4574, 2011.
- [103] John Schulman, Jonathan Ho, Alex X Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. *Robotics: Science and Systems (RSS)*, 9(1):1–10, 2013.
- [104] Marc Toussaint. Newton methods for k-order markov constrained motion problems. *arXiv preprint arXiv:1407.0414*, 2014.

- [105] Jim Mainprice, Nathan Ratliff, Marc Toussaint, and Stefan Schaal. An interior point method solving motion planning problems with narrow passages. In *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2020.
- [106] Waldez Gomes, Pauline Maurice, Eloïse Dalin, Jean-Baptiste Mouret, and Serena Ivaldi. Multi-objective trajectory optimization to improve ergonomics in human motion. *IEEE Robotics and Automation Letters*, 7(1):342–349, 2021.
- [107] Adam Fishman, Chris Paxton, Wei Yang, Dieter Fox, Byron Boots, and Nathan Ratliff. Collaborative interaction models for optimized human-robot teamwork. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots And Systems (IROS)*, pages 11221–11228, 2020.
- [108] Simon Schaefer, Karen Leung, Boris Ivanovic, and Marco Pavone. Leveraging neural network gradients within trajectory optimization for proactive human-robot interactions. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 9673–9679, 2021.
- [109] Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. Amass: Archive of motion capture as surface shapes. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5442–5451, 2019.
- [110] James Jerome Gibson. The senses considered as perceptual systems. 1966.
- [111] J.J. Gibson. *The Ecological Approach to Visual Perception*. Resources for ecological psychology. Lawrence Erlbaum Associates, 1979.
- [112] L. Jamone, E. Ugur, A. Cangelosi, L. Fadiga, A. Bernardino, J. Piater, and J. Santos-Victor. Affordances in psychology, neuroscience, and robotics: A survey. *IEEE Transactions on Cognitive and Developmental Systems*, 10(1):4–25, 2018.
- [113] Mohammed Hassanin, Salman Khan, and Murat Tahtali. Visual affordance and function understanding: A survey. *ACM Computing Surveys*, 54(3):1–35, 2021.
- [114] Anirban Roy and Sinisa Todorovic. A multi-scale cnn for affordance segmentation in rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 9908, pages 186–201, 2016.

-
- [115] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis. Detecting object affordances with convolutional neural networks. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots And Systems (IROS)*, pages 2765–2770, 2016.
- [116] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Learning object affordances: From sensory–motor coordination to imitation. *IEEE Transactions Robotics*, 24(1):15–26, 2008.
- [117] A. Gonçalves, G. Saponaro, L. Jamone, and A. Bernardino. Learning visual affordances of objects and tools through autonomous robot exploration. In *Proceedings of the IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 128–133, 2014.
- [118] Atabak Dehban, Lorenzo Jamone, Adam Kampff, and José Santos-Victor. Denoising auto-encoders for learning of objects and tools affordances in continuous space. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4866–4871, 2016.
- [119] Hema Swetha Koppula, Rudhir Gupta, and Ashutosh Saxena. Learning human activities and object affordances from rgb-d videos. *The International Journal of Robotics Research*, 32(8):951–970, 2013.
- [120] Henny Admoni and Brian Scassellati. Social eye gaze in human-robot interaction: a review. *Journal of Human-Robot Interaction*, 6(1), 2017.
- [121] Chien-Ming Huang and Bilge Mutlu. Anticipatory robot control for efficient human-robot collaboration. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2016.
- [122] Jan Hoffmann, Philipp Kratzer, Marc Toussaint, and Jim Mainprice. Towards activity recognition and anticipation using motion data and gaze. In *Workshop on Robotic Co-workers 4.0: Human Safety and Comfort in Human-Robot Interactive Social Environments*, 2018.
- [123] Nuno Ferreira Duarte, Mirko Raković, Jovica Tasevski, Moreno Ignazio Coco, Aude Billard, and José Santos-Victor. Action anticipation: Reading the intentions of humans and robots. *IEEE Robotics and Automation Letters*, 3(4), 2018.

- [124] Rachid Alami, Aurélie Clodic, Vincent Montreuil, Emrah Akin Sisbot, and Raja Chatila. Task planning for human-robot interaction. In *Proceedings of the ACM Joint Conference on Smart Objects and Ambient Intelligence*, 2005.
- [125] Séverin Lemaignan, Mathieu Warnier, E Akin Sisbot, Aurélie Clodic, and Rachid Alami. Artificial cognition for social human-robot interaction: An implementation. *Artificial Intelligence*, 247, 2017.
- [126] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: theory and practice*. Elsevier, 2004.
- [127] Thierry Siméon, Jean-Paul Laumond, Juan Cortés, and Anis Sahbani. Manipulation planning with probabilistic roadmaps. *The International Journal of Robotics Research*, 23(7-8), 2004.
- [128] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.
- [129] Neil T Dantam et al. An incremental constraint-based framework for task and motion planning. *The International Journal of Robotics Research*, 37(10), 2018.
- [130] Tomás Lozano-Pérez and Leslie Pack Kaelbling. A constraint-based method for solving sequential manipulation planning problems. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots And Systems (IROS)*, 2014.
- [131] Fabien Lagriffoul et al. Efficiently combining task and motion planning using geometric constraints. *The International Journal of Robotics Research*, 33(14), 2014.
- [132] Marc Toussaint. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.
- [133] Mamoun Gharbi, Raphaël Lallement, and Rachid Alami. Combining symbolic and geometric planning to synthesize human-aware plans - toward more efficient combined search. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots And Systems (IROS)*, 2015.

-
- [134] Baptiste Busch, Marc Toussaint, and Manuel Lopes 0001. Planning Ergonomic Sequences of Actions in Human-Robot Interaction. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018.
- [135] Marc Toussaint and Manuel Lopes. Multi-bound tree search for logic-geometric programming in cooperative manipulation domains. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.
- [136] Christopher M Bishop. Mixture density networks. 1994.
- [137] Tomislav Petković, Luka Petrović, Ivan Marković, and Ivan Petrović. Ensemble of lstms and feature selection for human action prediction. In *Proceedings of the International Conference on Intelligent Autonomous Systems (ICoIAS)*, pages 429–441. Springer, 2022.
- [138] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2008.
- [139] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 549–565. Springer, 2016.
- [140] Roberto Martín-Martín, Hamid Rezaatofghi, Abhijeet Shenoi, Mihir Patel, JunYoung Gwak, Nathan Dass, Alan Federman, Patrick Goebel, and Silvio Savarese. Jrdb: A dataset and benchmark for visual perception for navigation in human environments. *arXiv preprint arXiv:1910.11792*, 2019.
- [141] Sean Curtis, Andrew Best, and Dinesh Manocha. Menge: A modular framework for simulating crowd movement. *Collective Dynamics*, 1:1–40, 2016.
- [142] Tingxiang Fan, Xinjing Cheng, Jia Pan, Pinxin Long, Wenxi Liu, Ruigang Yang, and Dinesh Manocha. Getting robots unfrozen and unlost in dense pedestrian crowds. *IEEE Robotics and Automation Letters*, 4(2):1178–1185, 2019.
- [143] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: A benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 304–311, 2009.

- [144] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.
- [145] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nusenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11621–11631, 2020.
- [146] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2446–2454, 2020.
- [147] Cmu graphics lab motion capture database. <http://mocap.cs.cmu.edu>.
- [148] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, 2013.
- [149] Christian Mandery, Ömer Terlemez, Martin Do, Nikolaus Vahrenkamp, and Tamim Asfour. The kit whole-body human motion database. In *Proceedings of the IEEE International Conference on Advanced Robotics (ICAR)*, pages 329–336, 2015.
- [150] Christian Mandery, Ömer Terlemez, Martin Do, Nikolaus Vahrenkamp, and Tamim Asfour. Unifying representations and large-scale whole-body motion databases for studying human motion. *IEEE Transactions Robotics*, 32(4):796–809, 2016.
- [151] Timo von Marcard, Roberto Henschel, Michael Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In *Proceedings of the European Conference on Computer Vision (ECCV)*, sep 2018.

- [152] Saeed Ghorbani, Kimia Mahdaviani, Anne Thaler, Konrad Kording, Douglas James Cook, Gunnar Blohm, and Nikolaus F Troje. Movi: A large multipurpose motion and video dataset. *arXiv preprint arXiv:2003.01888*, 2020.
- [153] Pauline Maurice, Adrien Malaisé, Clélie Amiot, Nicolas Paris, Guy-Junior Richard, Olivier Rochel, and Serena Ivaldi. Human movement and ergonomics: An industry-oriented dataset for collaborative robotics. *The International Journal of Robotics Research*, 38(14):1529–1537, 2019.
- [154] Moritz Kassner, William Patera, and Andreas Bulling. Pupil: An open source platform for pervasive eye tracking and mobile gaze-based interaction. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, Adjunct publication, pages 1151–1160. ACM, 2014.
- [155] Mohamed El-Shamouty and Anish Pratheepkumar. Prednet: a simple human motion prediction network for human-robot interaction. In *Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–7, 2021.
- [156] Haziq Razali and Yiannis Demiris. Using a single input to forecast human action keystates in everyday pick and place actions. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3488–3492, 2022.

All URLs cited were checked in June 2022.

A

Appendix

A.1 Additional Datasets

A.1.1 Additional Human Motion Data

The *MoGaze* data presented in Chapter 7 is a solid basis for long-term human motion interacting with objects. However, we capture additional data in order to evaluate the performance of our framework on other aspects. This includes pure walking data (see Figure A.1a), data with less objects, as seen in Figure A.1b and Figure A.1c, upper-body data (see Figure A.1d), and data with two humans (see Figure A.3).

Walking Dataset

One additional dataset we consider covers pure walking data of one actor (see Figure A.1a). As pure walking is a very repetitive pattern, training and testing on

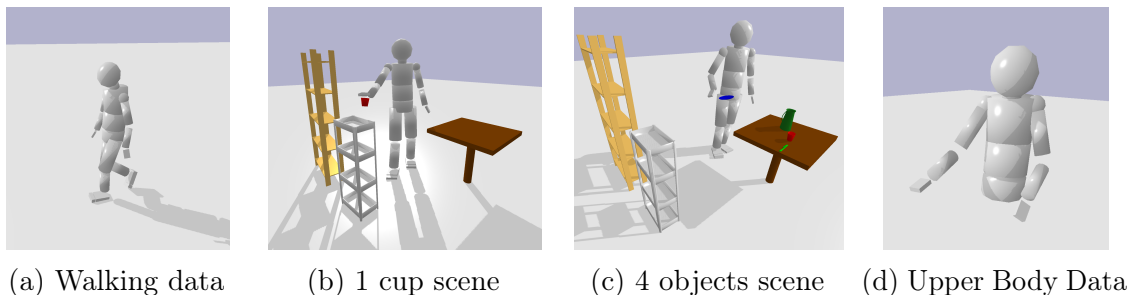


Figure A.1: Additional Datasets of Human Motion.

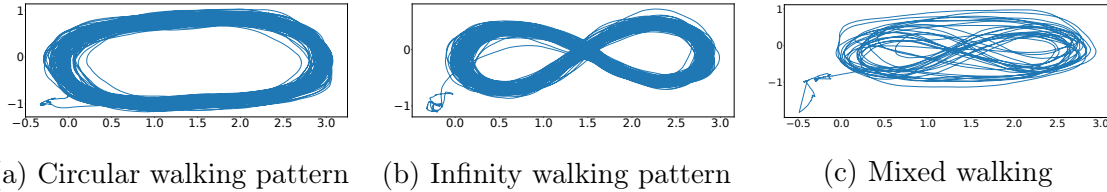


Figure A.2: Different Walking patterns in the *Walking* Dataset.

this dataset is useful for evaluating long-term behavior, for example, with respect to cascading errors.

For this dataset, one actor performed a total of 67 minutes of walking in the motion capture area. The data is distributed in five parts. In the first part the human is walking in a circle, as visible in Figure A.2a, in the second part he is switching the direction. In the third part the human is following an infinity symbol path as depicted in Figure A.2b, with changed direction in part four. In the fifth part, the human is alternating different walking patterns from the first four parts (see Figure A.2c). Parts one to four are used as training set and contain approximately 16 minutes of data each, part five contains only three minutes and is used as a test set.

Data with Less Objects

For the same actor as in the *Walking* dataset, additional pick and place actions with less objects were recorded. This additional data for one actor allows to test predictive behavior of models for only one human, generalizing about different scenes and different types of motion without the need of generalizing across different humans.

This data contains two parts. The first part, which we call *reaching1*, consists of 22 minutes of pick and place full-body motions with only one cup as object. The scene can be seen in Figure A.1b. The second part, which we call *reaching2*, contains pick and place motions with four objects: a knife, a plate, a cup and a jug. The scene can be seen in Figure A.1c. In total, the second part contains 32 minutes of motion data.

Upper-Body Data

Upper-body data (see Figure A.1d) is generally easier to capture and of lower-dimension than full-body data, which can be useful, for example, for evaluating models that do not cope well with very high dimensional data. For capturing upper-

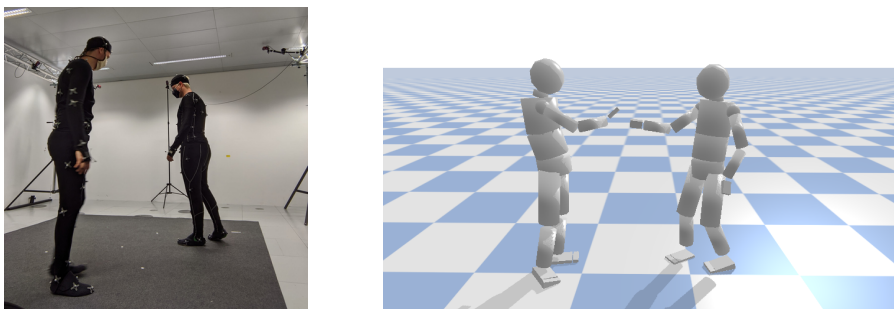


Figure A.3: Data with two humans. Left: capture setup with two people in motion capture suits. Right: two humans in the simulation.

body data, the actor wore a motion capture suit with 25 markers placed on the upper-body of the human. The actor was instructed to perform tasks with objects placed on two different tables in the motion capture area. Possible tasks were placing, drinking, pouring, opening, closing and scrubbing. Each task was preceded by a reaching motion to pick up the objects involved. In total 132 minutes of upper-body motion capture data with two different actors was recorded.

Data with two Humans

In a preliminary setup, we capture full-body motion data with two humans. Both humans are wearing a motion capture suit with 50 reflecting markers each (see Figure A.3). We captured data for two different scenarios:

- *Handover*: One human hands an object to the other human.
- *Collision Avoidance*: The two humans walk towards each other but avoid themselves.

Data with two humans is useful for training collaborative scenarios. Problems arise for capturing the data, as a lot more occlusions of markers occur. Reliable capturing of two humans in our motion capture setup is very difficult due to inclusions. It could be improved by adding more cameras placed on different heights and on the ceiling.

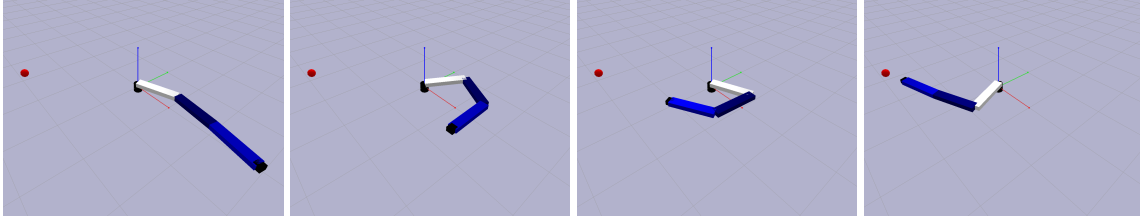


Figure A.4: A simple planar robot with four joints reaches towards the goal (red sphere).

A.1.2 Synthetic Datasets for Testing with Non-Human Data

In some cases it is not tractable for us to perform large-scale tests of our methods on human data because of its high-dimensionality. Additionally, testing on non-human data shows that approaches could be applied in other domains as well and are not limited to the use-case of human motion. In the following we introduce some simple synthetic datasets and explain how they are generated. Those datasets have the advantages that they are 1) lower-dimensional than the human motion data and 2) an arbitrary amount of data can be synthesized.

Four Joint Robot

We created simple discrete motion trajectories using a planar robot arm with four degrees of freedom as seen in Figure A.4. Several trajectories have been generated so that the robot starts from a fixed start state and reaches towards a randomized endeffector goal position. This was done by interpolating the endeffector start and goal positions on a line and using a basic inverse kinematics update, so that the endeffector follows a straight line to the goal. Data can be generated as necessary by also specifying the number of timesteps. The resulting discrete trajectories contain the robot joint states for every timestep.

One-dimensional Data

For testing with one-dimensional data, we simply create trajectories by discretizing sinus curves. Example trajectories can be seen in Figure A.5. The curves are randomly shifted in x -direction. The amplitude of the curves is randomly scaled between 0.8 and 1.2. The curves are randomly shifted by ± 1 in y -direction. We use 30 frames per period. To each point a small Gaussian noise of $\mathcal{N}(0., 0.02)$ is added.

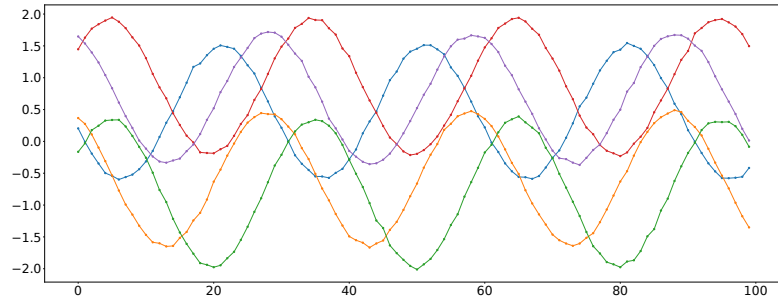


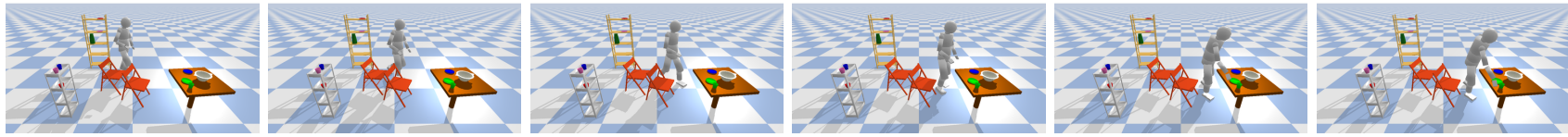
Figure A.5: Example trajectories of the one-dimensional test data.

The idea of the one-dimensional data is to be able to test the time-series prediction data on small input dimensions. As this allows for smaller networks, due to less complexity of the data, training and testing on this data can be done much faster compared to the full-dimensional human data.

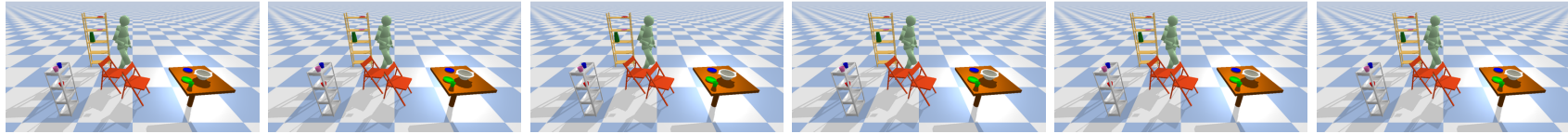
A.2 Goal Constraint Experiment: All Baselines

On the following pages we show example trajectories from different methods and baselines for our method with goal constraint. It is an extension of Figure 4.6.

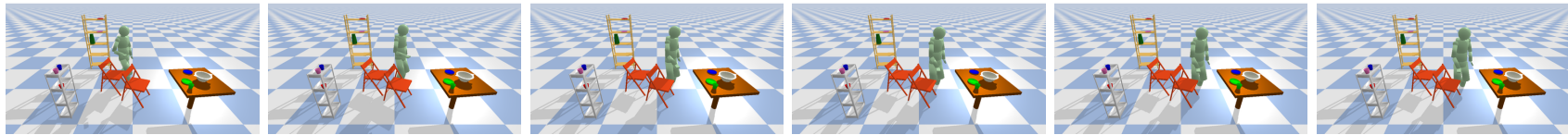
Figure A.6 shows the ground truth, zero velocity baseline, and the initial prediction of the neural network architectures. Figure A.7 shows the ground truth and the sample baselines for the neural network architectures. Figure A.8 shows our optimization based framework for the neural network architectures and an inverse kinematics solution for the zero velocity baseline.



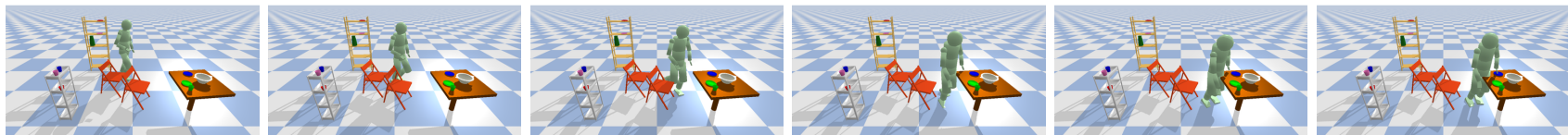
(a) Ground truth



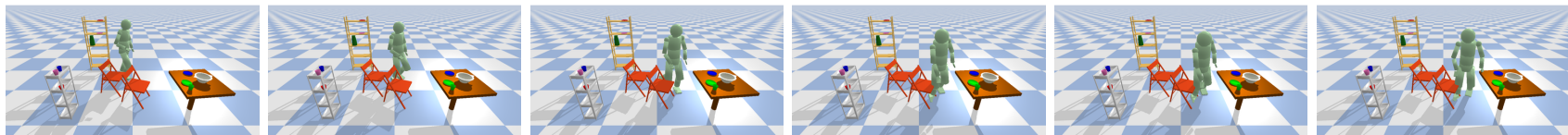
(b) Zerovel



(c) Basic-GRU



(d) Residual



(e) PVRED

Figure A.6: Example trajectories of the initial baseline for different architectures. From left to right: Trajectory after 0s, 0.4s, 0.8s, 1.2s, 1.6s and 2s.

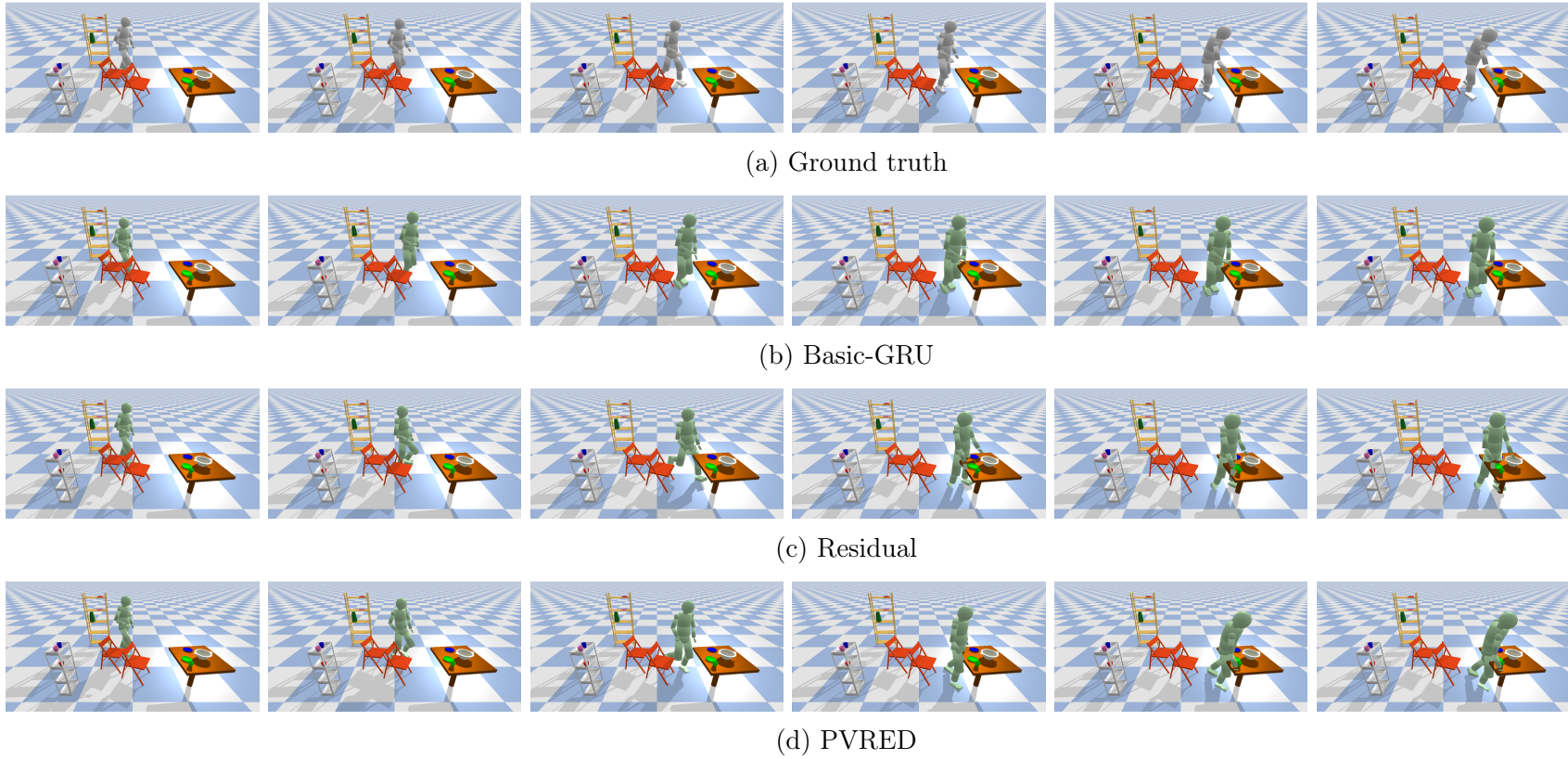
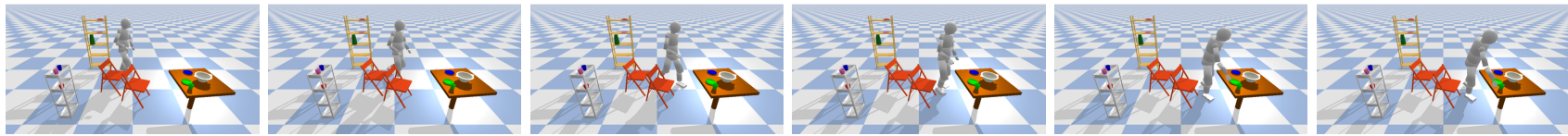
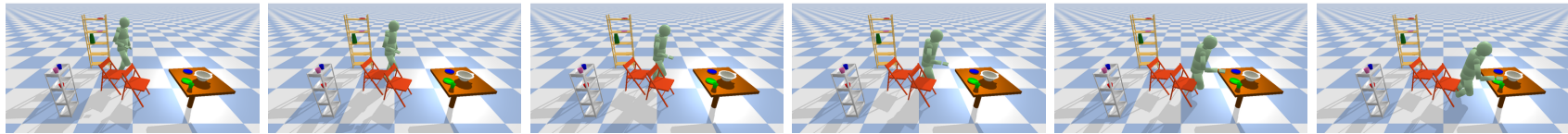


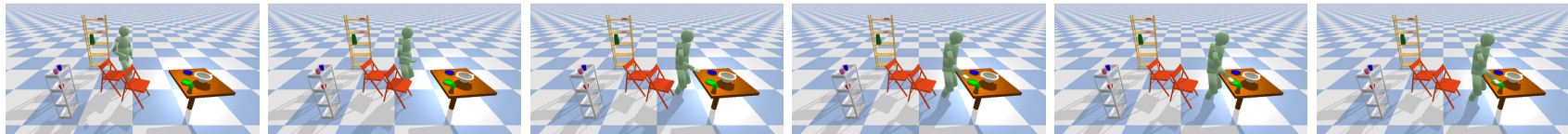
Figure A.7: Example trajectories of the sample baseline with goal distance for different architectures. From left to right: Trajectory after 0s, 0.4s, 0.8s, 1.2s, 1.6s and 2s.



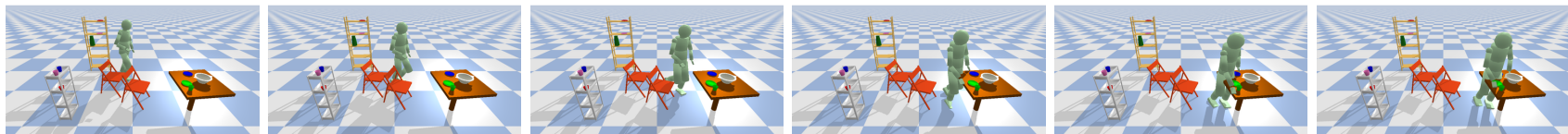
(a) Ground truth



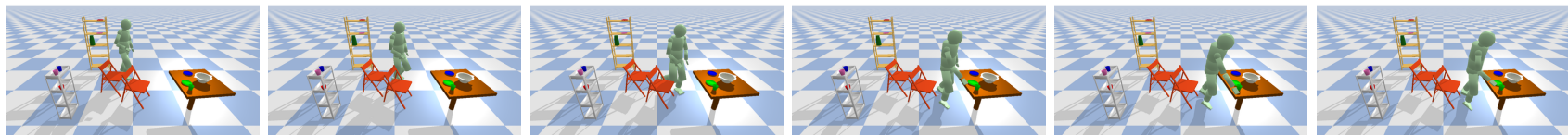
(b) ZeroVel



(c) Basic-GRU



(d) Residual



(e) PVRED

Figure A.8: Example trajectories of our method with goal constraint for different architectures. From left to right: Trajectory after 0s, 0.4s, 0.8s, 1.2s, 1.6s and 2s.

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Dissertation selbstständig und ausschließlich auf der Grundlage der in der Arbeit angegebenen Hilfsmittel und Hilfen verfasst habe.

Philipp Kratzer

Stuttgart, den 22.06.2022