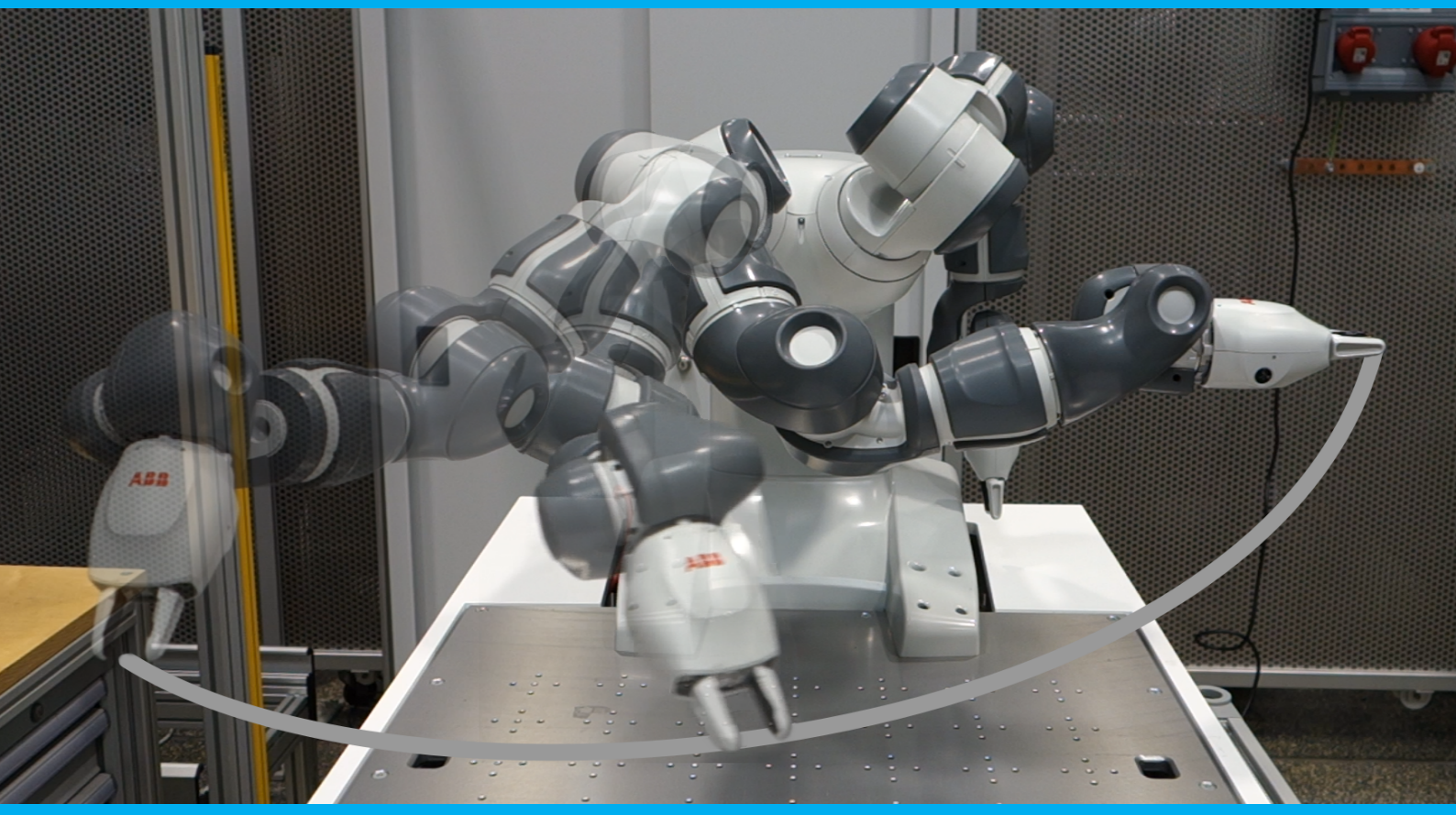
Inverse Optimal Control for robot arm motions

Improving human-robot collaboration by incorporating human characteristics into optimal motion generation

Eert Hoogerwerf

MSc Thesis - November 2018



TU Delft

Inverse Optimal Control for robot arm motions

Improving human-robot collaboration by incorporating human characteristics into optimal motion generation

by

Eert Hoogerwerf

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Thursday December 6, 2018 at 14:00.

Student number:	4230353	
Project duration:	February 1, 2018 – November 1, 2018	
Supervisors:	Dr. Debora Clever,	ABB Germany
	Dr. Mukunda Bharatheesha,	TU Delft
Thesis committee:	Prof. dr. ir. Martijn Wisse,	TU Delft
	Dr. Erik Steur,	TU Delft
	Ir. Linda van der Spaa,	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

This thesis is part of the master's degree program of Mechanical Engineering at the TU Delft. For the last year I have been working on this project at the Corporate Research Center of ABB in Ladenburg, Germany. It started with a three month internship followed by the research project detailed in this thesis. At ABB I found a pleasant working environment with excellent supervision, mainly by my direct supervisor Debora Clever. I'm grateful for all the help and feedback I got, also from the other people working at the research center. Besides that I liked working alongside the other students there, they made this experience so much more enjoyable and I have made some valuable friendships. I also want to thank my supervisor from the university, Mukunda Bharatheesha, who gave me helpful guidance during this project. Lastly I would like to thank all my friends and family for supporting me during the journey towards the master's degree.

Eert Hoogerwerf
Delft, November 2018

Abstract

Human-robot collaboration can be improved if the motions of the robot are more legible and predictable. This can be achieved by making the motions more human-like. It is assumed that humans move optimal with respect to a certain objective or cost function. To find this function an inverse optimal control approach is developed. It uses a bilevel optimization for finding what linear weighted combination of physically interpretable cost functions best mimics human point-to-point motions. The upper level compares the optimal result of the lower level with a reference motion. Two depth cameras are combined in a motion capture setup to record this reference motion.

The human arm is modeled as a seven degrees of freedom manipulator, similar to a robot arm model of the YuMi. The bilevel optimization is done with both models, resulting in two differently weighted cost functions. The cost function that was derived using the robot model is then used to generate new motions for the corresponding real robot arm. The results of an experiment show that humans experience these motions as more anthropomorphic and feel at least equally as safe compared to existing motion planning strategies. This proves the validity of the inverse optimal control method and shows it is a step forward for better collaboration between humans and robots.

Contents

List of Figures

List of Tables

1 Introduction

In an environment where humans and robots are working or living together a mutual understanding of each other is crucial. One way to improve this is by humanizing Human-Robot Interaction (HRI), which means that the robot not only tries to understand the human's actions and intentions, but also that it uses human-like features to communicate its own. This makes sense because we are highly trained in interacting with other humans, even if this nonverbal communication happens unconsciously. It does not mean the robot has to completely imitate the human, but the objective is to make its actions legible and predictable [35].

There are so called anthropomorphic robots arms that have a kinematic structure similar to the human arm, for example the YuMi in Figure 1.1. These robots are physically safe enough to work close by or next to a human coworker. Their motions however, can come across as intimidating and are often far from human-like. To improve this we could directly copy human motion paths, but it would make more sense to learn the underlying design principles and use them or adapt them for the robot.

Figure 1.1: YuMi robot by ABB [1]

It is considered that humans plan their motions in an optimal way with respect to an objective function [13]. Optimal control is then a logical choice for achieving the same on a robot, as here as well a minimum of an objective function is found, while satisfying the constraints. But which objective function humans use has to be determined and depends on the task requirements. For this inverse optimal control comes into play, where the aim is to find the objective function that best resembles an optimal reference. This objective function can then be used to generate motions towards new targets, but the question remains how humans experience these motions and if there is an improvement over existing methods.

1.1. Related works

Here an overview is given on the existing work done on the topics regarding optimized motion generation and inverse optimal control. More related works are also covered in the corresponding chapters itself.

1.1.1. Optimized motion generation

With optimized motion generation an objective is minimized or maximized while constraints ensure the resulting motion is feasible. This can be done over the whole time horizon, like with optimal control. Alternatively, Model Predictive Control optimizes over a small timespan into the future. Lastly, it can be done in a so-called step-by-step manner, where the optimum is found for every single time-step.

Trajectory optimization

In robotic manufacturing tasks often only an end-effector target is specified and the path towards the target has to be determined. This causes a high degree of redundancy because an infinite number of paths are possible. To find the path that is optimal with respect to a certain cost function, an optimization over the whole time horizon needs to be done. This is the approach used in this thesis, also referred to as optimal control and explained in chapter 4.

Model Predictive Control

Trajectory optimization is not suitable for online implementation as it optimizes the motion over the whole time horizon. An online method based on optimal control is Model Predictive Control (MPC). It uses a prediction horizon receding into the future where an optimal control sequence is calculated based on the current inputs, outputs and states. Only the first control step is applied and for every next step a new control sequence is calculated. Constraints on controls and states are possible, but it can be difficult to avoid instabilities. The final solution is also not optimal when considering the entire time-span and the computational requirements can be high [26].

Step-by-step optimization

For step-by-step optimization only the controls and states of one individual time-step are considered. There still has to be some degree of redundancy in order to perform this optimization. This is the case for a redundant manipulator, where there are more degrees of freedom than needed to reach a required end-effector target. A Jacobian-based method can find an optimal arm configuration for an end-effector target. The Jacobian relates the joint angles and end-effector target on velocity level

$$\dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \quad \boldsymbol{J}(\boldsymbol{q}) = \left(\frac{\partial x_i}{\partial q_j}\right)_{i,j}, \tag{1.1}$$

where $\boldsymbol{x}$ represents the end-effector target in Cartesian space and $\boldsymbol{q}$ the joint angles. For a redundant manipulator this Jacobian is non-square and the inverse cannot be calculated directly. Different methods have been developed to deal with this problem, including the pseudoinverse, or more specifically the Moore-Penrose inverse [30], and the augmented Jacobian [20]. Both methods can incorporate cost functions that are projected into the nullspace of the Jacobian, which means they have no influence on the end-effector position or orientation. These methods are used by Yoshikawa [42] to avoid singularities, by Liegeois [22] to avoid joint limits or by Maciejewski and Klein [24] to avoid obstacles.

These methods can be implemented in a real-time manner, where a feedback term can be added to move towards a target or to account for noise and other inaccuracies. Similar to MPC the final trajectory will not optimal, but the computational requirements are relatively low.

1.1.2. Inverse Optimal Control

The methods explained before are used to find an optimal motion based on a certain objective function. When this function is unknown inverse optimal control can be used to find this objective function from an

optimal reference. For finding the objectives of humans, recorded motion can be used. Inverse optimal control can be done in multiple ways, most notably are the bilevel and the KKT-based approaches. Chapter 5 gives an overview on the work done on these topics.

The motion objectives humans use can change during a continuous motion. For example the control strategy when moving the hand towards an object might be different than while moving the object. To discover these changing objective functions a sliding window approach can be used, where the optimization is done over a sliding timespan of fixed width instead of over the whole time horizon. However, the focus of this thesis is on robotic manufacturing tasks, where motions are often point-to-point and separated into different motion commands. The recorded reference motions will also be point-to-point and will have no interaction with the environment. For this reasons optimal control will be used and the objective function is assumed to be constant over the time horizon.

1.2. Research goal

The research goal of this thesis is two-fold. First inverse optimal control is used to find what linearly weighted combination of physically interpretable cost functions best mimics human point-to-point motions. With this combination of cost functions new motions can be generated. Secondly the goal is to validate this method for generating motions by determining how humans experience them compared to motions from existing methods commonly used in robotics.

1.3. Contribution

In this thesis an inverse optimal control method is developed for motions of a seven degrees of freedom arm. It uses a bilevel optimization to identify the cost functions from human captured motion. An optimal control method using a multiple shooting algorithm is developed for the lower level of the bilevel optimization and for generating motions towards new targets. To record the reference motions in the form of trajectories a motion capture setup is created. Two depth cameras are combined to capture the motion of a human arm and hand in three-dimensional space.

The resulting cost functions are used to generate new optimal motions for a robot arm. To see if the optimal motions are beneficial to human-robot interaction an experiment is performed with human participants. The motions are compared with two existing methods on perceived anthropomorphism and feeling of safety.

1.4. Structure

In the next chapter a model of the human arm is defined and compared with a robot arm. The models will be used later in the optimizations. In chapter 3 the motion recording system is explained, as this is needed to generate reference motions. Chapter 4 explains the different methods for optimal control and the implementation used for this thesis. Optimal control is used for the lower level of inverse optimal control, which is explained in chapter 5. In the last chapter newly generated motions are validated by implementing them on a real robot and performing an experiment with human participants.

2

Arm models

2.1. Introduction

For the analyses of the next chapters a model of the human arm and the robot arm are needed. This chapter shows how the kinematic and dynamic parameters of the human arm model are chosen and how this differs from an anthropomorphic robot arm. A comparison is made with a model of the YuMi robot, which has two arms with each seven degrees of freedom. The arm angle is suggested as parameter to describe the redundant dimension and to compare motions between both arms.

2.2. Human model

The human arm can be modeled as a seven degrees of freedom manipulator, Figure 2.1a shows the kinematic structure. The shoulder joint can be approximated as a ball-and-socket joint [37] and is modeled with three revolute joints with zero offset. q_1 represents abduction/adduction, q_2 the flexion/extension and q_3 the rotation along the longitudinal axis. For the elbow joint the flexion/extension is represented by q_4 and the pronation/supination of the lower arm by q_5. The wrist is also modeled as a sperical joint with q_6 for the flexion/extension and q_7 for the deviation of the wrist. Table 2.1 shows the DH-parameters of the model, the offsets for θ_i are chosen for correspondence with the robot model. d_3 is the length of the upper arm, d_5 of the lower arm and a_7 of the hand.

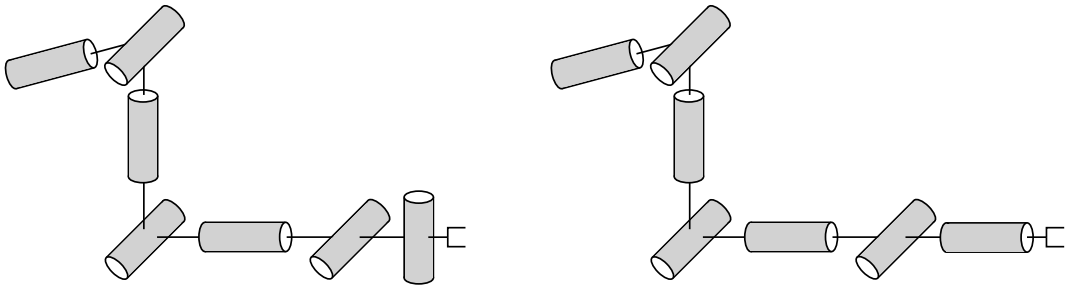
Table 2.1: DH-parameters of the human arm (classic convention)

axis	a_i	d_i	α_i	θ_i
1	0	0	$\pi/2$	q_1
2	0	0	$-\pi/2$	q_2
3	0	d_3	$\pi/2$	q_3
4	0	0	$\pi/2$	$q_4+\pi/2$
5	0	d_5	$-\pi/2$	q_5
6	0	0	$\pi/2$	$q_6-\pi/2$
7	a_7	0	0	q_7

The mass, position of centre of mass (COM) and radius of gyration of the human arm segments are based on the work by De Leva [12]. The mass and COM are given in reference to the joint centers. The arm segment inertia around a specific axis is calculated as

$$I_{jj} = m(k_j)^2, \tag{2.1}$$

with m the mass of the segment and k the radius of gyration around the specific axis. De Leva [12] expresses the mass as a percentage of the total mass of the body, noted as $\bar{m}$. The radius of gyration is expressed as percentage of the length of the segment, $\bar{k}$. So the formula for calculating the inertia of a segment around a



(a) Human arm (b) Robot arm

Figure 2.1: The kinematic structure of the two models. The last joint has a different orientation, the differences in offsets are not shown here

principal axis is specified in the paper as

$$I_{jj} = M\bar{m}(l \cdot \bar{k}_j)^2, \tag{2.2}$$

with M the total mass of the body and l the segment length.

For a male subject with a body mass of 70 kg the mass of the upper arm is 2.63% of the total mass. The COM is positioned at 57.63% of the longitudinal length, which is in this case 28 cm. The inertia matrix is then

$$\boldsymbol{I} = \begin{bmatrix} 0.0113 & 0 & 0 \\ 0 & 0.0035 & 0 \\ 0 & 0 & 0.0101 \end{bmatrix} [kg \cdot m^2] \tag{2.3}$$

The lower arm weights 1.5% of the total body weight. The COM is positioned at 45.67% of the longitudinal length, which is 27 cm. The inertia matrix is

$$\boldsymbol{I} = \begin{bmatrix} 0.0056 & 0 & 0 \\ 0 & 0.0009 & 0 \\ 0 & 0 & 0.0052 \end{bmatrix} [kg \cdot m^2] \tag{2.4}$$

The hand weights 0.6% of the total weight.

The inertia of a single arm segment is constant and invariant to its corresponding joint rotation. But this is not the case for the stiffness and damping factors of the joints as they can be changed by neural activity of the brain and the reflexes. In Piovesan et al. [31] it was found that stiffness and damping vary during movements.

In this thesis a simplified human arm model will be used where the stiffness and damping are ignored. It would be more realistic to model the arm with the individual muscles and their neural inputs, but this would be out of the scope of this project. Also acceleration based control will be used for the optimal control of chapter 4, meaning the dynamic parameters of the model only influence the cost functions.

2.3. Robot model

The YuMi robot has seven joints that have a structure similar to Hollerbach's proposed design of a seven degrees of freedom manipulator [18], see Figure 2.1b. Compared to the human, the main difference is that where the human arm has spherical joints, the robot has revolute joints with offsets between them. Also the last wrist joint has a different orientation, as q_7 is again pronation/supination.

In the dynamics the differences between the two arms is larger due to the differences in actuation and materials. The mass of the YuMi arm is considerably larger than that of the human arm and they have different centers of mass and inertia. The YuMi robot will be controlled on position and velocity level, as torque based control is not supported. The internal controller will calculate the correct motor commands. The dynamic parameters are still needed for the cost functions used later, as they are based on the mass and inertia parameters of the arm.

2.4. Implementation

The models are implemented with the Robotic Toolbox [11] in Matlab. The Toolbox has functions for calculating the arm Jacobian and can give joint torques or accelerations for a specified configuration. The dynamic equation to calculate the joint torques in the arm is

$$\boldsymbol{Q} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{F}(\dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}), \tag{2.5}$$

where $\boldsymbol{M}$ is the inertia matrix, $\boldsymbol{C}$ the Coriolis and centripetal coupling matrix, $\boldsymbol{F}$ the friction matrix and $\boldsymbol{G}$ the gravity matrix. To calculate the torques the Toolbox uses an efficient recursive Newton-Euler scheme [39].

2.5. Arm angle

A seven degrees of freedom manipulator has one degree of redundancy for tasks that are defined with six position and orientation variables. The *arm angle* can be used to describe this last degree of redundancy which can simplify the comparison of motions of two manipulators with small kinematic differences. It was shown that humans percieve motions of a human and a robot arm as similar if the same arm angle is used [40]. In Kreutz-Delgado et al. [20] the arm angle is defined as the angle between a vertical reference plane and a plane though the shoulder, elbow and wrist, as pictured in Figure 2.2. It is calculated as function of the joint angles as

$$\psi = \text{atan2}(\hat{\boldsymbol{w}}^T(\hat{\boldsymbol{z}} \times \boldsymbol{p}), \hat{\boldsymbol{z}}^T \boldsymbol{p}), \tag{2.6}$$

where $\boldsymbol{p} = (\boldsymbol{I} - \hat{\boldsymbol{w}}\hat{\boldsymbol{w}}^T)\boldsymbol{e}$, $\hat{\boldsymbol{w}}$ is the unit vector from shoulder to wrist, $\boldsymbol{e}$ is the vector from shoulder to elbow and $\hat{\boldsymbol{z}}$ is the unit vector in the vertical direction of the base frame. The arm angle is undefined if $\boldsymbol{e}$ and $\boldsymbol{w}$ are co-linear.

In the YuMi controller the arm angle is calculated with reference to the y-axis. So instead of $\hat{\boldsymbol{z}}$ the unit vector in direction of $\boldsymbol{y}$ of the base frame has to be used.

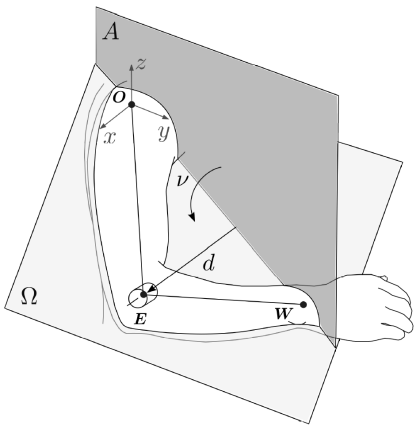


Figure 2.2: Arm angle, from Zanchettin [43]

2.6. Motion comparison

In a natural position of the arm both models have similar the end-effector position and orientation as shown in Figure 2.3. But for other joint angles this is not the case, as shown in Figure 2.4. This is mainly caused by the different orientation of the last joint. When giving both models the same trajectory described with joint angles the arm angle is similar, see Figure 2.5. This is because the arm angle depends on the position of the wrist and not on the position or orientation of the end-effector.

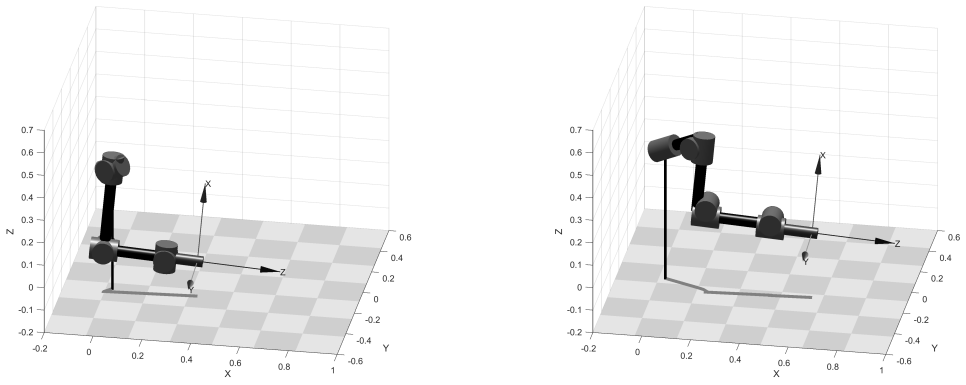


Figure 2.3: In this situation the human (left) and YuMi models with the same joint angles have a similar end-effector orientation

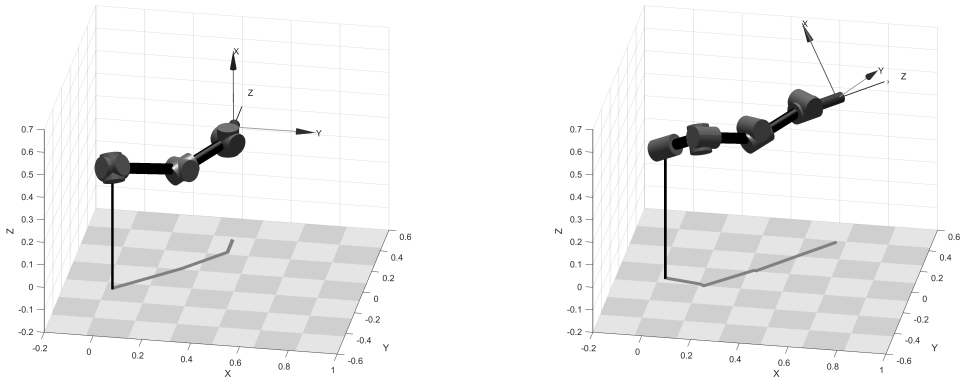


Figure 2.4: In this situation the human and YuMi models with the same joint angles have a different end-effector orientation, mainly due differences in the wrist

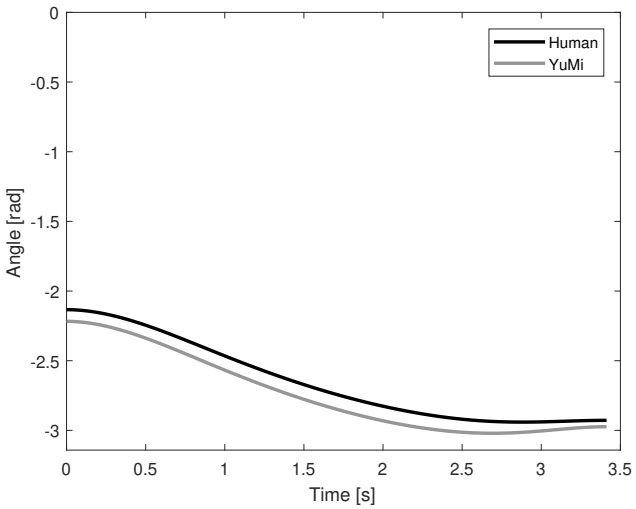


Figure 2.5: Arm angle for the human and the YuMi model for the same joint angle trajectory

3

Motion capture

3.1. Introduction

To find out how humans plan their motions, a motion capture system is needed. This way a reference motion can be recorded which can be used for the inverse optimal control method of chapter 4. This reference motion consists of a trajectory over time of the hand position and orientation in the body reference frame and the arm angle.

Often a marker-based system, like the Vicon system, is used for motion capture, because of its high accuracy and framerate. But this system is expensive and time-consuming to set up. Therefore a markerless system using two depth cameras is developed. The two cameras both have their optimal range and combining them leads to more accurate tracking of arm motions compared to what the cameras can achieve on their own.

3.2. Depth cameras

Conventional cameras record colors which can be saved in RGB format. Depth cameras add an extra value for distance, indicated as RGB-D. Two depth cameras were chosen for the motion recording setup. The Kinect v2 offers a large range for measuring body positions but its accuracy is limited for the hand and fingers. To solve this issue the Leap Motion is added, which cannot measure the whole human body, but offers better hand and finger tracking.

(a) Kinect v2 from [17]

(b) Leap Motion from [19]

Figure 3.1: The two depth cameras used in the motion recording setup

3.2.1. Kinect

The Microsoft Kinect v2 (Figure 3.1a) was developed for the Xbox game console, but has found great attention from researches in robotics and human sciences because of its low price and high performance compared to

other depth cameras. It uses Time-of-Flight to measure distances based on the time between emitting and receiving infrared light. With this information a depth map of the environment can be created and humans can be better detected than with only RGB data. Internally it uses a classifier to pixel-wise segment different parts of the body based on the depth image data and from this segmentation 3D locations of the joints are determined [36].

Compared to the first generation, the Kinect v2 offers higher resolution images and improved body tracking. For the arm joints the accuracy was found to be within 100 mm [41]. Table 3.1 shows the specification of the camera. Its range and Field of View enable body tracking of a human sitting or standing in front of the camera. Figure 3.2 shows which parts of the body are tracked, the included SDK gives access to the three-dimensional positions of these body parts with respect to the camera frame.

Schlagenhauf et al. [34] find that the Kinect v2 can reliable track human motion for the upper-body, but self-occlusions hurt the tracking performance. A dual-Kinect setup is proposed to deal with this problem, which performs well compared to a Vicon system. In the setup of this thesis only one Kinect is used but a different depth camera is added as it is specialized in hand tracking, an important part of the reference recording.

Table 3.1: Camera specifications

	Kinect v2	**Leap Motion**
RGB resolution	1920 x 1080	-
IR resolution	512 x 424	640 x 240 (2x)
IR Framerate	30 fps	20 to 200 fps
Field of View	70^o H and 60^o V	150^o H and 120^o V
Range	0.5 to 4.5 meters	25 to 600 millimeters
Depth sensing	Time-of-Flight	Stereo vision

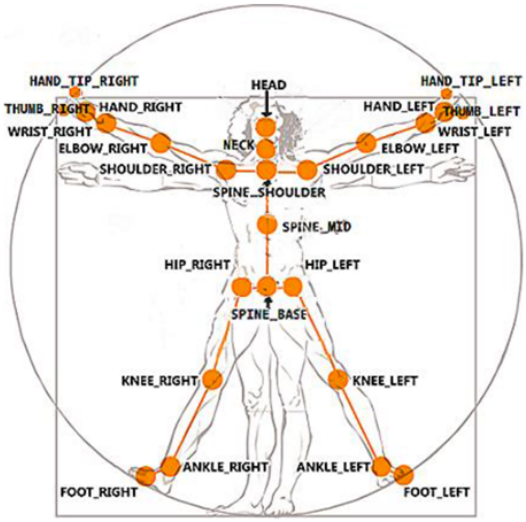


Figure 3.2: Body locations tracked by the Kinect v2

3.2.2. Leap Motion

The Leap Motion (Figure 3.1b) uses two fish-eye IR cameras to calculate depth using stereo vision with a wide field of view. The shifts between features on the two camera images are used to calculate the distances and to create a three-dimensional representation of the environment. Table 3.1 shows the specifications, it is meant to be used at close range positioned on a table or mounted on a VR headset. The SDK offers hand and finger tracking, where the accuracy is high enough to enable tracking of individual bone segments of fingers. This makes the camera more suitable for measuring the position and orientation of the hand and lower arm.

3.3. Implementation

The software for the motion capture system is programmed on Windows in C++, because this is supported by the SDKs of both cameras. The two cameras are placed in a setup as illustrated in Figure 3.3. The body positions of the Leap Motion and the Kinect need to be transferred to the same coordinate frame in order for them to be combined.

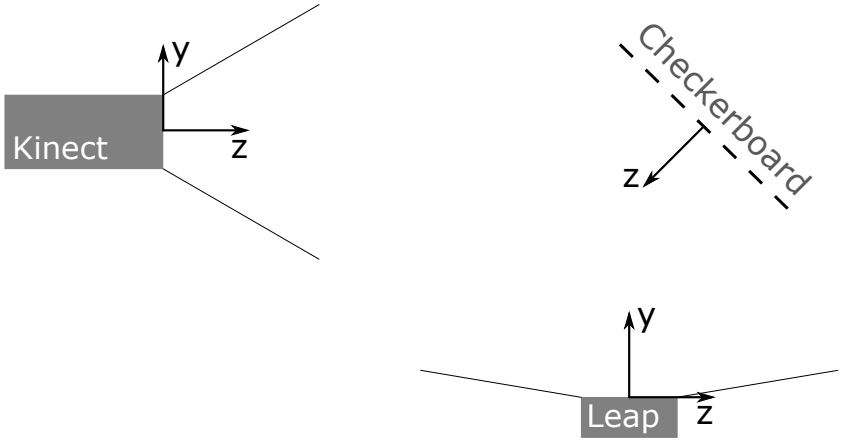


Figure 3.3: Kinect and Leap setup

To calculate the transformation between the two cameras a printed checkerboard pattern is used. The checkerboard is detected using the `cv::findChessboardCorners()` function from OpenCV [9]. If a cameras sees the checkerboard the function `cv::solvePnP()` calculates the position and orientation of the board based on the positions of the checkerboard corners in the image and the camera's intrinsic and extrinsic parameters. From this the transformation matrix from camera to checkerboard is calculated.

If both cameras detect the board at the same time, the transformation matrix from Leap to Kinect can be calculated as

$$\boldsymbol{T}_{LtoK} = \boldsymbol{T}_{LtoBoard}\boldsymbol{T}^{-1}_{KtoBoard}. \tag{3.1}$$

This transformation matrix can be used to get the Leap body positions in the Kinect reference frame

$$\boldsymbol{p}_K = \boldsymbol{T}_{LtoK}\boldsymbol{p}_L. \tag{3.2}$$

If both cameras are in a fixed position, this calibration only needs to be done one time.

The final position of the positions of the lower arm and hands are based on the confidence level provided by the Leap Motion. It is a value between 0 and 1 based on how well the camera sees the hand

$$\boldsymbol{p}_{final} = (1 - \text{confidence})\ \boldsymbol{p}_K + \text{confidence}\ \boldsymbol{p}_L. \tag{3.3}$$

This ensures that the hand is tracked even outside of the range of the Leap Motion without making sudden jumps.

3.4. Body frame

The positions of the arm need to be measured in reference to the human body. For the body reference frame the position `SpineBase` is taken as origin. The axes are calculated as

$$\begin{aligned}
\boldsymbol{z} &= \boldsymbol{p}_{SpineMid} - \boldsymbol{p}_{SpineBase} \\
\boldsymbol{x} &= \boldsymbol{z} \times (\boldsymbol{p}_{ShoulderRight} - \boldsymbol{p}_{ShoulderLeft}) \\
\boldsymbol{y} &= \boldsymbol{z} \times \boldsymbol{x}
\end{aligned}$$

This is then used to calculate the transformation matrix from the Kinect to the body.

3.5. TCP

The position of the Tool Center Point (TCP) of the human hand is defined as the middle between the tip of the index finger and the tip of the thumb as shown in Figure 3.4. The TCP-axes are calculated as

$$\boldsymbol{z} = \frac{1}{2}(\boldsymbol{p}_{index} - \boldsymbol{p}_{thumb}) - \boldsymbol{p}_{hand}$$
$$\boldsymbol{y} = \boldsymbol{z} \times (\boldsymbol{p}_{hand} - \boldsymbol{p}_{wrist})$$
$$\boldsymbol{x} = \boldsymbol{y} \times \boldsymbol{z}$$

The orientation with respect to the body coordinate frame is expressed in Quaternion form.

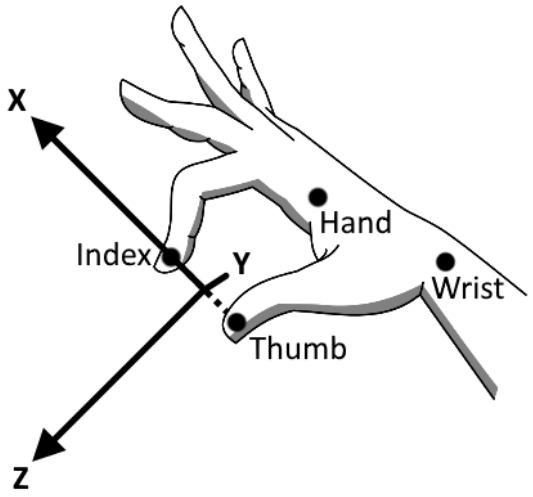
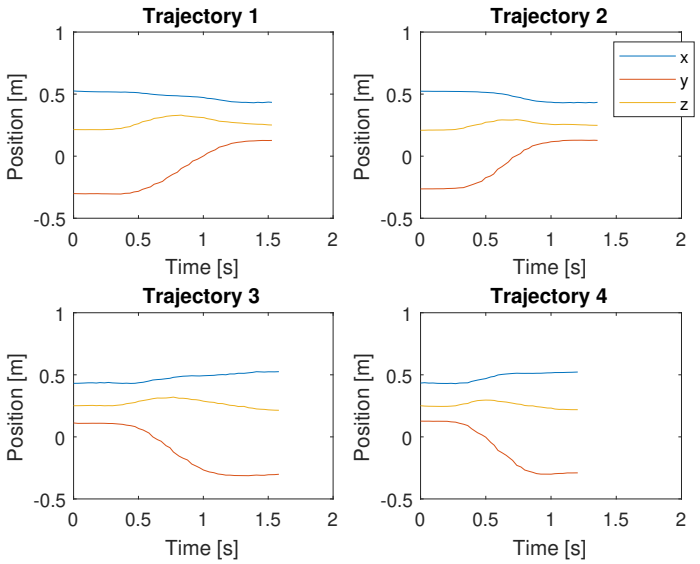


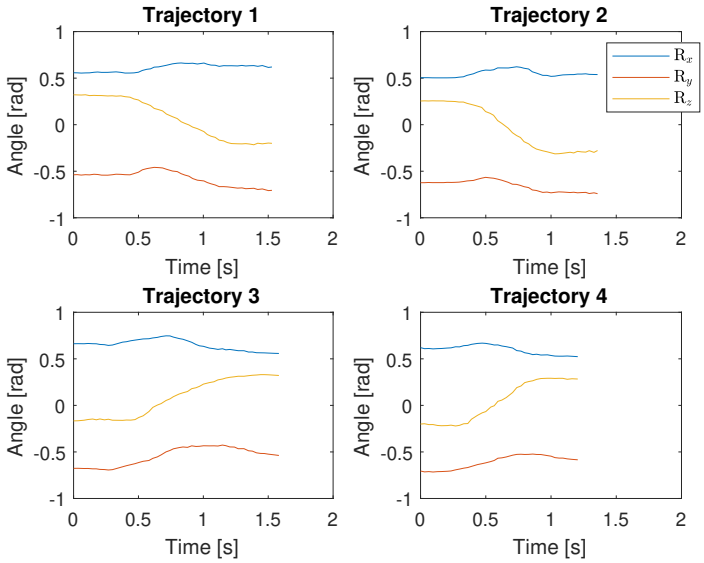
Figure 3.4: TCP of the human hand

3.6. Recording

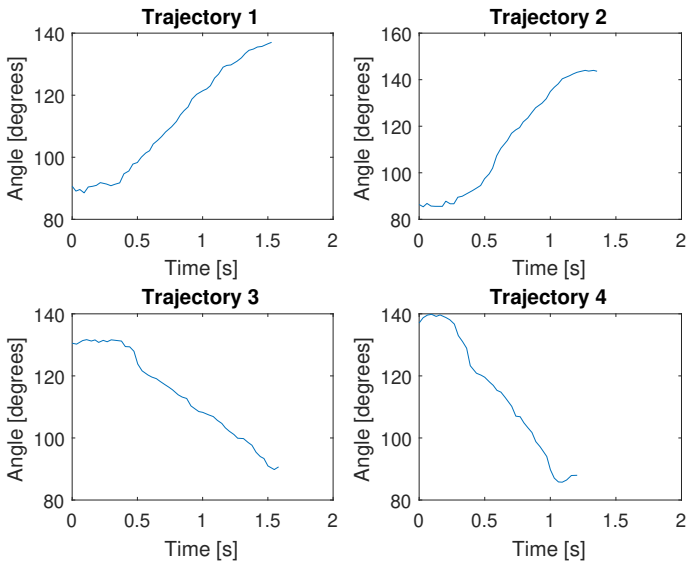
To apply the inverse optimal control of chapter 5 human recorded reference motion is needed. For this four point-to-point motions are recorded with the setup. The arm angle is calculated from the body positions with Equation 2.6. The recording is done by writing the measured TCP position and orientation, the arm angle and the corresponding time to a textfile. The file can then be imported into Matlab. Figure 3.5 shows the four recorded reference motions.



(a) TCP position



(b) TCP orientation



(c) Arm angle

Figure 3.5: Recorded reference motions

4

Optimal Control

4.1. Introduction

This chapter explains the method for calculating motions that are optimal with respect to a known objective or cost function. The implementation will be used for the lower level of the bilevel optimization in chapter 5 and for generating new optimal motions in chapter 6 based on the objective function derived with inverse optimal control.

Human and robot manipulation tasks often involve moving the hand to a target that is described by a position and an orientation

$$\boldsymbol{x} = \begin{bmatrix} x & y & z & \theta_x & \theta_y & \theta_z \end{bmatrix}^T. \tag{4.1}$$

For an arm to move from a starting pose to this target there are an infinite number of paths to choose from. The objective here is to find the motion that is optimal with respect to a cost function. This is done by minimizing the cost, which is calculated over the whole time horizon of the motion, while satisfying the constraints. The minimization will be done with a direct optimization method using multiple shooting.

The starting and final target points are defined as the position and orientation of the end-effector in Cartesian space: $\boldsymbol{x_0}$ and $\boldsymbol{x_f}$. The starting and final joint angles are undetermined, but the starting and final joint angle speeds have to be zero to simulate rest-to-rest motions as for example in pick-and-place tasks. Another possibility would be to define the starting joint angles of the arm, because usually a movement is started from a known arm configuration, but that case is not considered here.

4.1.1. Indirect versus direct optimization

Indirect optimization methods first construct the necessary conditions for optimality, for example with Pontryagin's Maximum Principle [29]. This leads to an expression of the optimal trajectory that can be solved numerically as a boundary value problem. On the other hand direct methods first discretize the states and controls and then use these as optimization parameters. This transforms the problem into a finite dimensional nonlinear programming problem that can be solved with numerical optimization methods.

The two approaches both have their advantages and disadvantages. For the indirect method the optimality conditions depend on the quality of the gradients of the cost and constraint functions, so these usually have to be determined analytically. For the direct method these can be calculated numerically, giving it more flexibility. Secondly the direct method can more easily deal with inequality constraints [14], as the constraints can directly work on the optimization variables. Finally, the indirect optimization tends to be highly sensitive to the initial guesses of state and adjoint variables [6]. This makes the method less robust compared to the direct method, but it has the advantage of higher accuracy of the solution. Because of its higher flexibility and because accuracy is not critical, here the direct method is used.

4.2. Optimization

For the direct method the optimization problem is formulated as

$$\min_{z,u,\tau,T} \int_0^T \Big[\sum_{i=1}^{n} \alpha_i U_i(\boldsymbol{z}(t), \boldsymbol{u}(t)) \Big] dt, \tag{4.2}$$

$$\begin{aligned}
\text{subject to:} \quad & \textit{System equation: } \dot{\boldsymbol{z}} = \Omega(t, \boldsymbol{z}(t), \boldsymbol{u}(t)) && (4.3)\\
& \textit{Forward kinematics BC: } f(\boldsymbol{q}(0)) = \boldsymbol{x}_0;\ f(\boldsymbol{q}(T)) = \boldsymbol{x}_e && (4.4)\\
& \textit{Joint angular speeds BC: } \dot{\boldsymbol{q}}(0) = 0;\ \dot{\boldsymbol{q}}(T) = 0 && (4.5)\\
& \textit{Interval times: } \sum \boldsymbol{\tau}_i = T,\ \tau_i > 0,\ T > 0 && (4.6)
\end{aligned}$$

The states $\boldsymbol{z}$ are the seven joint angles and joint angular speeds and the joint acceleration is chosen as control variable

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix}, \quad \boldsymbol{u} = \ddot{\boldsymbol{q}}. \tag{4.7}$$

The system equation Ω is

$$\dot{\boldsymbol{z}} = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{u} \end{bmatrix}. \tag{4.8}$$

This differential equation is solved to ensure an actual motion is performed. In section 4.3 torque is used as control, which means the system equation will be different as well.

An interpolation function is used to transform the discretized controls into a continuous function, see section 4.2.5.

The cost is chosen as a linearly weighted combination of convex functions. For example the speeds and controls squared, with the final time added to minimize the time of the motion

$$U = \alpha_1 T + \alpha_2 \cdot \dot{\boldsymbol{q}}^T \dot{\boldsymbol{q}} + \alpha_3 \cdot \boldsymbol{u}^T \boldsymbol{u}. \tag{4.9}$$

The parameters α_i are used to give a weighting to the cost functions. The cost is calculated based on the simulated trajectory.

The boundary constraints of Equation 4.4 are enforced with a function that outputs a vector as

$$W_{ceq} = \begin{bmatrix} f(\boldsymbol{q}(0)) - \boldsymbol{x}_0 \\ f(\boldsymbol{q}(T)) - \boldsymbol{x}_e \end{bmatrix}. \tag{4.10}$$

The zero starting and final speeds of Equation 4.5 are enforced with linear constraints. The linear constraint of Equation 4.6 ensures that the sums of the interval times are equal to the final time of the motion. This final time T is variable.

4.2.1. Single and multiple shooting

Shooting methods are used to numerically solve boundary value problems. Multiple shooting was introduced by Bock and Plitt [8] as a method to solve optimal control problems.

Single shooting

For single shooting the system is simulated starting from $\boldsymbol{z}_0$. This is an ordinary differential equation that can be solved using a variable step-size method, in this case with the `ode45` function from Matlab. The result is a trajectory, which also depends on the controls, see Figure 4.1 for the case of a single joint.

In this case the controls $\boldsymbol{u}_i$ make up the control function by linear interpolation and the interval times $\boldsymbol{\tau}_i$ can be varied as well. The state $\boldsymbol{z}_0$ and resulting final state $\boldsymbol{z}(T)$ are used in the constraint function. So the

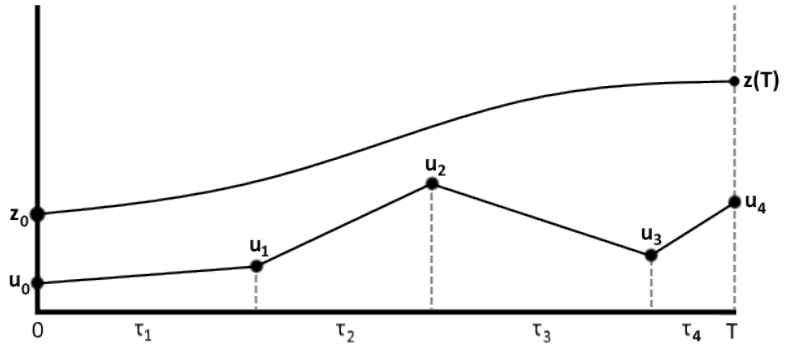


Figure 4.1: Single shooting with piecewise linear interpolation of the controls

parameters that have to be optimized to find the minimum of the objective function are

$$\boldsymbol{\omega} = \begin{bmatrix} \boldsymbol{u}_0 & \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_n \\ \boldsymbol{\tau}_1 & \boldsymbol{\tau}_2 & \cdots \; \boldsymbol{\tau}_n & \begin{bmatrix} T \\ 0 \end{bmatrix} \\ \boldsymbol{z}_0 & 0 & \cdots & 0 \end{bmatrix}, \tag{4.11}$$

where n is the number of control intervals. Every joint has his own set of discretized controls with their own interval times. The sum of these intervals all have to add up to the final time T, which is variable as well. So this leads to seven linear constraints for a seven degrees of freedom arm

$$\sum_{i=1}^{n} \boldsymbol{\tau}_i = T. \tag{4.12}$$

Multiple shooting

For multiple shooting the timespan is divided into a fixed number of intervals and on each of those intervals the simulation is done, see Figure 4.2. The states $\boldsymbol{z}_i$ are sometimes referred to as *shooting nodes*.

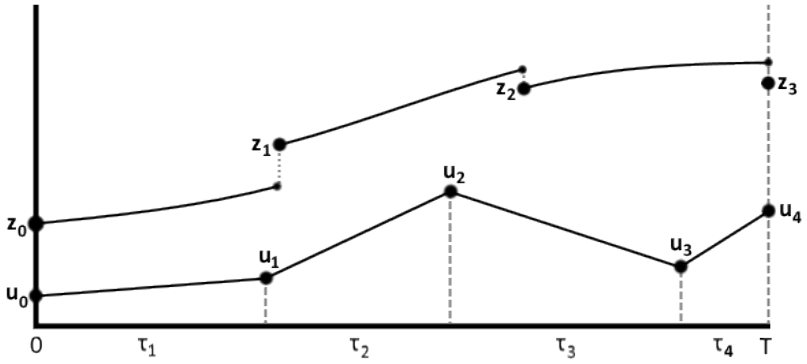


Figure 4.2: Multiple shooting with piecewise linear interpolation of the controls and three shooting intervals

To ensure that the trajectory stays continuous, the differences between the final state on the interval and the initial state of the next interval are added to the constraint function. The intervals are equally spaced over the timespan and their length depends on final time T.

The optimization variables are then

$$\boldsymbol{\omega} = \begin{bmatrix} \boldsymbol{u}_0 & \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_n & (0) \\ \boldsymbol{\tau}_1 & \boldsymbol{\tau}_2 & \cdots \; \boldsymbol{\tau}_n & \begin{bmatrix} T \\ 0 \end{bmatrix} & (0) \\ \boldsymbol{z}_0 & \boldsymbol{z}_1 & \cdots & \boldsymbol{z}_m & (0) \end{bmatrix}, \tag{4.13}$$

where n is the number of control intervals and m the number of shooting intervals. Zeros are added at the end if the row sizes are not equal.

The shooting intervals make the optimization more stable and therefore less sensitive to the initial guess. This is because the shorter intervals decrease the chance of the system going unstable exponentially, after which it would be difficult to find a stable trajectory. Furthermore the nonlinearities of the system are spread out over the continuity constraints instead of being accumulated at the end constraint [14].

4.2.2. SQP

To solve the optimization the Sequential Quadratic Programming (SQP) algorithm is used. It uses the Lagrangian function

$$\mathscr{L}(\boldsymbol{\omega}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = G(\boldsymbol{\omega}) - \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{\omega}) - \boldsymbol{\mu}^T \boldsymbol{h}(\boldsymbol{\omega}), \tag{4.14}$$

with the objective function

$$G(\boldsymbol{\omega}) = \int_0^T \Big[\sum_{i=1}^{n} \alpha_i U_i(\boldsymbol{z}(t), \boldsymbol{u}(t)) \Big] dt \tag{4.15}$$

and equality constraints $\boldsymbol{g}(\boldsymbol{\omega})$ and inequality constraints $\boldsymbol{h}(\boldsymbol{\omega})$.

According to the KKT-conditions for a local optimum the optimal parameters $\boldsymbol{\omega}^*$, $\boldsymbol{\lambda}^*$ and $\boldsymbol{\mu}^*$ have to furfill these necessary conditions

$$\nabla_\omega \mathscr{L}(\boldsymbol{\omega}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) = 0 \tag{4.16}$$

$$\boldsymbol{g}(\boldsymbol{\omega}^*) = 0 \tag{4.17}$$

$$\boldsymbol{h}(\boldsymbol{\omega}^*) \geq 0, \; \boldsymbol{\mu}^* \geq 0, \; \boldsymbol{h}(\boldsymbol{\omega}^*)^T \boldsymbol{\mu}^* = 0 \tag{4.18}$$

The general method of SQP solves the following sub-problem

$$\min_{\Delta\omega} \; \frac{1}{2} \Delta\omega^T H_k \Delta\omega + \nabla_\omega G(\omega_k)^T \Delta\omega \tag{4.19}$$

$$\begin{aligned} \text{subject to:} \quad & g(\omega_k) + \nabla_\omega g(\omega_k)^T \Delta\omega = 0 \\ & h(\omega_k) + \nabla_\omega h(\omega_k)^T \Delta\omega \geq 0 \end{aligned}$$

The matrix H_k is a positive definite approximation of the Hessian matrix of the Lagrangian. The solution is used to iteratively find the optimum as

$$\omega_{k+1} = \omega_k + \Delta\omega. \tag{4.20}$$

4.2.3. Gradient calculation

The SQP algorithm relies on the gradients of the objective function and the nonlinear constraint functions. The gradient is the transposed derivative towards all optimization variables

$$\nabla_\omega G = \Big(\frac{\partial G_i}{\partial \omega_j} \Big)_{i,j}. \tag{4.21}$$

This gradient can be calculated analytic or approximated numerically with a finite difference method.

Objective function

The objective function depends on the solution of the integration of the system equation and the interpolation of $\boldsymbol{u}$. The gradient is therefore calculated numerically using a finite difference approximation

$$\nabla F = \frac{F(x+\delta) - F(x)}{\delta}. \tag{4.22}$$

Here δ is chosen based on the square root of the double precision spacing: $\delta = \sqrt{\texttt{eps}} = 1.4901 * 10^{-8}$.

Boundary constraints

The equality constraint at the beginning and end of the motion based on the forward kinematics is $f(\boldsymbol{q}_b) - \boldsymbol{x}_b = 0$. This has both a translational and a rotational part, where it was chosen to represent rotation in rotation matrix form. If the two poses have an infinitesimally difference between them, the translational and a rotational difference is

$$\boldsymbol{\delta} = \begin{bmatrix} \boldsymbol{t}_x - \boldsymbol{t}_b \\ \text{vex}(\boldsymbol{R}_x \boldsymbol{R}_b^T - \boldsymbol{I}_{3\times 3}) \end{bmatrix}. \tag{4.23}$$

This uses the property $\boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I}_{3\times 3}$ and vex() transforms a skew-symmetric matrix to a vector. At the start of the optimization the two poses will not be infinitesimally close to each other, but the closer it gets, the more accurate this pose difference will be.

For the gradient of this constraint with respect to $\boldsymbol{q}$ the geometric Jacobian can be used, because $\boldsymbol{\delta}$ is a representation of spacial velocity. The Jacobian relates joint velocities with the Cartesian velocities of the end-effector and is defined as

$$\dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}, \quad \boldsymbol{J}(\boldsymbol{q}) = \left(\frac{\partial x_i}{\partial q_j}\right)_{i,j}. \tag{4.24}$$

The Jacobian is relatively easy to calculate and because it is an exact expression it leads to fast convergence. This constraint only depends on $\boldsymbol{q}_b$, so the gradient is zero for all other variables.

The second boundary constraint ensures zero joint speeds at the beginning and end of the motion: $\dot{\boldsymbol{q}}_b = 0$. This is a linear constraint, so the gradient does not have to be provided.

Continuity constraints

Like the objective function, the continuity constraints depend on the integrated system equation and the interpolation of $\boldsymbol{u}$. The gradient is therefore calculated numerically as well.

4.2.4. Initial guess

The minimization needs an initial guess of the controls and the states. For the controls this can be all zero. The states can be based on an initial guess of the start and final joint angles and the states in between can be taken with linear interpolation. In Figure 4.5a the start and final joint angles are based on an inverse kinematics approach and Figure 4.5b shows the resulting solution of the optimization.

Another option is to use a natural pose of the arm as initial guess for start and final joint angles

$$\boldsymbol{q}_0 = \boldsymbol{q}_m = \begin{bmatrix} 120 & 120 & -30 & 0 & 0 & 0 & 0 \end{bmatrix}^T. \tag{4.25}$$

The initial trajectory of the joint angles is therefore constant as shown in Figure 4.6a. The resulting solution is shown in Figure 4.6b and is practically the same as in Figure 4.5b. This shows that the optimization, for this case, is not dependent on its initial guess. However if the initial angles are chosen far from the solution, the result is different or the optimization does not converge.

4.2.5. Control interpolation

The interpolation of the discretized controls can be done in different ways, as shown in Figure 4.3.

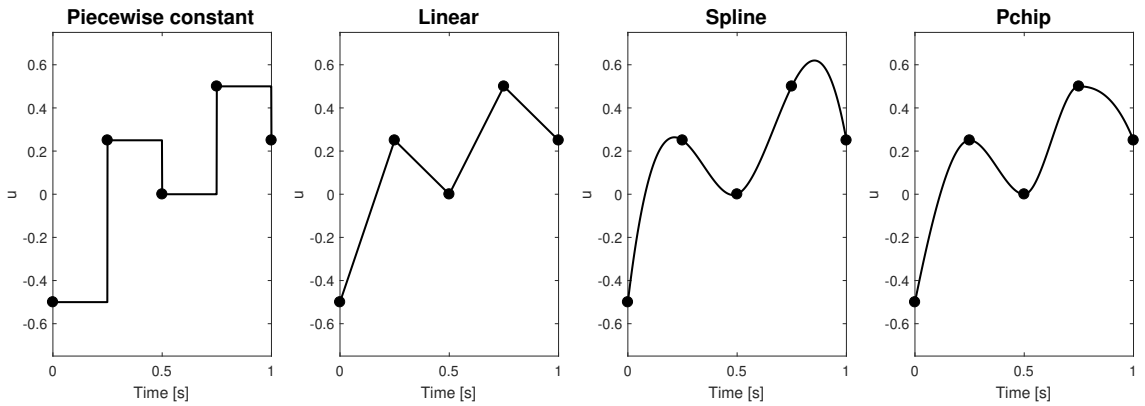


Figure 4.3: Interpolation methods

Piecewise constant The closest discrete point to the left is taken as value for the interpolated points.

Piecewise linear Linear interpolation between the discrete points.

Piecewise cubic This function, called `spline` in Matlab, uses cubic interpolation with a continuous first and second derivative. This makes the interpolated function smooth, but it can have overshoot.

Shape preserving piecewise cubic For `pchip`, the slopes at the interior points are based on a weighted harmonic mean of the slopes of the piecewise linear interpolant. Compared to `spline` it is more similar to the linear interpolated function, but it has jumps in the second derivative.

The required amount of discrete points is based on the expectation that the velocity trajectories will have a bell-shaped profile for a point-to-point motion. The control, which is in this case the acceleration, will then have a sine or cosine-like profile. To describe this five discrete points are sufficient and it was found that using more would not lead to different results.

4.2.6. Computation

The most time-consuming part of the optimization is the calculation of the gradients. It was found that parallel computation made the gradient calculation approximately four times faster and therefore considerately increased the speed of the optimization.

To avoid having to simulate the system for both the cost and the constraint function, the two are combined into one function. This makes the numerical gradient calculation considerably faster, as the system has to be simulated half as many times.

4.2.7. ODE optimization

Optimizing a system with a differential equation has a number of challenges. First of all variable ode solvers change their step size depending on the accuracy of the solution. For the numerical gradient calculation this means that the small offset δ can lead to a different final step size and because of that a large change in the solution, what leads to an incorrect gradient estimation. Also any errors within the derivative function accumulate over time, so when numerically calculating the gradient this can lead to large errors. The consequence of these problems is that the optimization can take long to converge and the first-order optimality, which is a measure on how close the solution is to optimal, will not get low.

Furthermore, adaptive ode solvers tend to have problems with integrating non-smooth functions. So for the piecewise constant and linear interpolation this leads to problems as their derivative is non-continuous.

Therefore the cubic interpolation methods are used. The shape preserving version is found to perform better, because there is less change of overshoot.

The composite cost function needs to be integrated over the timespan, so it can be added to the state function and be solved at the same time. This way the accuracy of the solution of the total cost is also incorporated in the error control of the ode-solver

$$\begin{bmatrix} \dot{\boldsymbol{z}} \\ \dot{U} \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{u} \\ \sum_{i=1}^{n} \alpha_i U_i(\boldsymbol{z}, \boldsymbol{u}) \end{bmatrix}. \tag{4.26}$$

4.3. Torque-based control

Until now joint acceleration is considered as control variable, which is useful for trajectory generation where the internal controller of the robot calculates the correct motor commands. If instead torque based control is required the dynamic equation of the arm has to be used, see Equation 2.5.

With torque as control variable ($\boldsymbol{u} = \boldsymbol{Q}$) the system equation becomes

$$\dot{\boldsymbol{z}} = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{M}(\boldsymbol{q})^{-1}\big(\boldsymbol{u} - \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{F}(\dot{\boldsymbol{q}}) - \boldsymbol{G}(\boldsymbol{q})\big) \end{bmatrix}. \tag{4.27}$$

This makes the optimization much more taxing, as for every simulation step the four matrices need to be derived and a matrix has to be inverted. An optimal motion was generated with the YuMi model, see Figure 4.4. This shows that the optimal control method works with torques as control as well. But for the inverse optimal control of the next chapter, torque based control will not be used because of how time consuming it is and because the YuMi robot will not be controlled with torques anyway.

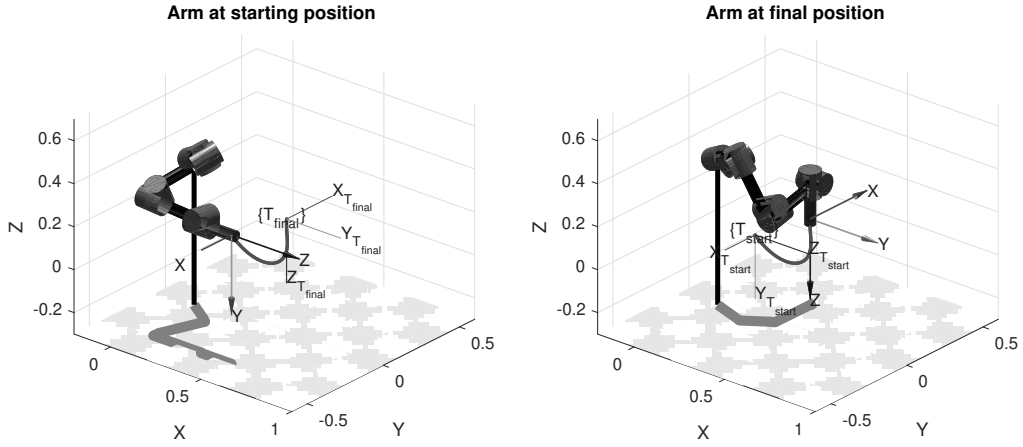
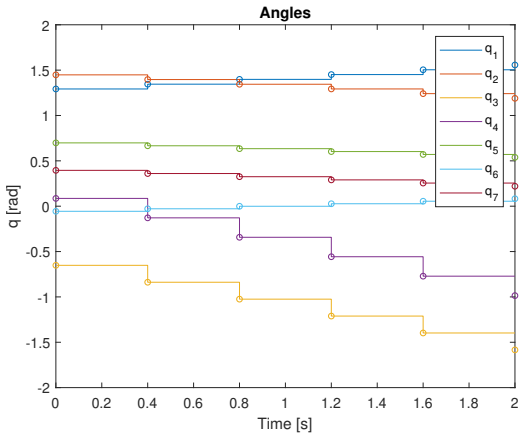
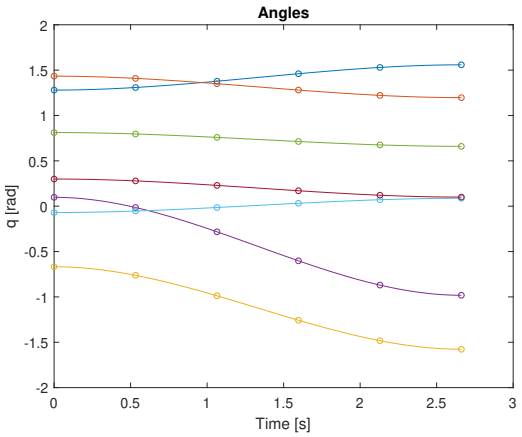


Figure 4.4: Trajectory from torque-based optimal control

Figures

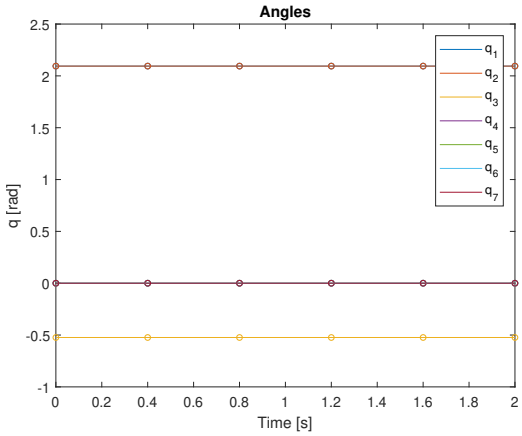


(a) Initial guess based on an inverse kinematics start and end position and linear interpolation of the states. The circles represent the *shooting nodes* and the lines represent the simulated trajectory

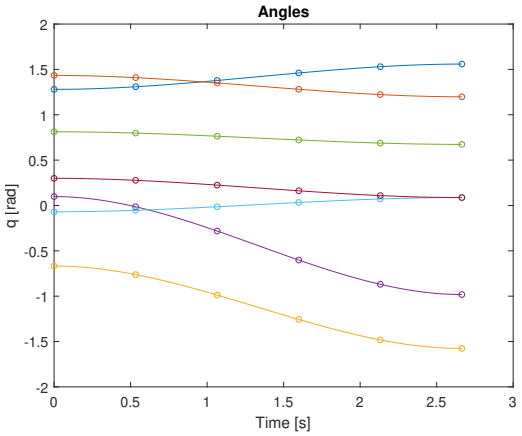


(b) Optimal result with inverse kinematic derived angles as initial guess

Figure 4.5: Optimal initial pose



(a) Initial guess based on a natural pose of the arm



(b) Optimal result with natural pose as initial guess

Figure 4.6: Natural initial pose

5

Inverse Optimal Control

5.1. Introduction

Where optimal control is used to find a motion that is optimal with respect to a certain objective or cost function, inverse optimal control can be used to identify that function based on an optimal reference motion. To help understand what objectives humans use when planning their motions, motion recorded as described in chapter 3 will be used as reference. A bilevel optimization is done to find which weighted combination of cost functions best replicates the recorded motion.

First an overview on related work done on this topic is given, then the bilevel and KKT-based methods are explained. What follows is an explanation on the choice of cost functions and finally the results of the bilevel optimization are given.

5.2. Related works

The bilevel approach from Mombaur et al. [27] splits up the optimization into two levels. The upper level minimizes the difference between the reference and the derived motion by adjusting certain parameters that are transferred to the lower level. The lower level then finds the optimal motion based on these parameters and sends it back to the upper level. This optimal motion is compared with the reference motion and if needed adjusted parameters are again sent to the lower level until the whole optimization convergences. The bilevel approach is applied to a locomotion task where optimal walking paths are found based on recorded walking paths of human participants.

Berret et al. [5] use the bilevel approach from Mombaur et al. [27] for a pointing task in a horizontal plane. The upper level optimization finds the weights of a number of cost functions and the lower level finds the optimal joint trajectory based on these weighted cost functions. The participants had to point on a vertical bar and were free to choose the path and the final location on the bar. The cost functions that have the biggest contributions are those that minimize energy consumption and angle acceleration. The Geodesic, angle jerk and hand jerk criteria have relative small contributions, while the other cost function have no contribution at all.

Sylla et al. [38] apply a similar bilevel approach to a human arm reaching task. It is found that the criteria minimizing energy consumption had the highest contribution, followed by the Geodesic criteria. The functions minimizing Cartesian and angle jerk, angle acceleration, torque and torque change all have low contributions to the final cost function. In this case a four degrees of freedom manipulator and movement in the three dimensional plane is considered, which makes the movement more realistic than the planar task in [5]. It is argued that the results are still in correspondence with this previous research, as in both cases energy spending is minimized while joint smoothness is maximized. The acquired weights lead to movements similar to what was recorded but the inverse optimization does require high computational power, taking around 30 hours to complete the optimization.

Arm reaching motions in collaboration tasks are researched by Mainprice et al. [25]. Two participants performing a number of pick-and-place tasks while standing next to each other were recorded. For the inverse optimal control cost functions are used based on distances to the other participant and functions achieving smoothness. The trajectories planned with the recovered weights are found to better resemble human motion than with manually tuned weights.

Albrecht et al. [2] transform the bilevel problem into a standard nonlinear as optimization problem

$$\min_{\alpha,x,\lambda} \text{dist}(p^{ref}, p^{comp}(x))^2, \tag{5.1}$$

subject to

$$\nabla_x G(x,\alpha) + \nabla_x h(x)\lambda = 0 \tag{5.2}$$

$$h(x) = 0 \tag{5.3}$$

$$\sum_i \alpha_i = 1 \tag{5.4}$$

$$\alpha_i \geq 0 \tag{5.5}$$

With objective function $G(x)$ and constraints function $h(x)$. Equation (5.2) is based on the first order necessary conditions for an optimum corresponding to the minimization of the Lagrangian.

This is used for finding the weights that replicate unconstrained human reaching motions. Only cost functions minimizing hand and joint jerk and change of torques are considered. The performance of cost functions strongly differs between tasks, what also means that the optimal combination of weights is different for each task. In general minimization of joint jerk and change of torques performs better than minimization of hand jerk and, as expected, the optimal combination always gives the best results.

Englert et al. [15] propose the inverse KKT method, also a one-level optimization but instead of the approach of Albrecht et al. [2] the KKT-conditions are used in the cost function. Their composite cost function is made of a weighted sum of squared features. These features can be comparable to what was described before, like torques or accelerations. Or they can be based on environmental variables, like the end-effector position with respect to an object. This makes the method usable for contact and manipulation tasks. Their method can incorporate time variable weights and can automatically handle multiple demonstrations without the need to derive one general reference motion. Their inverse KKT approach is faster and more accurate than the bilevel approach, but the gradient must be modeled explicitly.

5.2.1. Combining cost functions

The idea of inverse optimal control is to select a number of cost functions based on domain knowledge and find which ones best explain the reference data. Usually a combination of cost functions performs better. There are multiple ways of making this combination.

Linear weighting

One way of learning the objective function U can be by assuming that it is a linear combination of cost functions U_i weighted by parameters α_i that do not change over time. The problem of finding the objective function is now reduced to finding the weight factors α_i. The contribution of a cost function to the total cost is calculated as

$$c_i = \frac{\alpha_i U_i}{U(\boldsymbol{\alpha})}, \quad \text{with } U(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i U_i. \tag{5.6}$$

Time-dependent and nonlinear weightings

Lin et al. [23] state that humans use task dependent criteria to optimize their movement. So instead of the whole timespan, a sliding window with fixed width is used to estimate the weights of each cost function. The weights at a given timestep are found by averaging the weights found in the windows that overlap with that timestep. The inverse optimal control is done with the inverse KKT approach. For a squatting task, it was found that humans minimize power consumption during rest and Cartesian acceleration during movement.

Englert et al. [15] propose a number of alternatives ways to combine cost functions:

- Time-dependent weights, with or without the sliding window as explained. Leads to a high-dimensional parameter space.
- Radial or B-spline basis functions, equally distributed over the time horizon. This can describe the smooth varying cost functions more compactly.
- Gaussian base functions, where the means and variances are nonlinear parameters and the weights are linear.

They conclude that the linear radial basis functions are best for estimating which cost functions are used over time. The linearity makes the inverse KKT problem convex, which makes solving easier than with nonlinear parametrization.

5.3. Bilevel optimization

For the inverse optimal control in this work the bilevel approach is chosen. The advantage is that the gradient of the cost and constraint functions do not have to be derived exactly, making it considerably more flexible, for example when trying out different kinds of cost functions. Moreover, the aim is to identify the objective function used by humans, which is a one-time optimization, and then generating optimal motions offline. If a real-time application is required for the forward optimal control, a KKT-based approach would be a better choice. The cost functions are combined with linear weightings, invariant of time. This is because only short point-to-point motions are considered, with no contact with the environment.

5.3.1. Upper level

The upper level optimization uses the parameters α_i to minimize the difference between the reference and the derived motion in a least-squares manner

$$\min_{\alpha} \sum_{j=1}^{m} ||\boldsymbol{v}_j^*(\boldsymbol{q}^*(t;\boldsymbol{\alpha})) - \boldsymbol{v}_j^M(t)||^2, \tag{5.7}$$

where $\boldsymbol{v}^*$ is based on the solution of the lower level optimization. The vector $\boldsymbol{\alpha}$ is sent to the lower level, which returns an optimal solution in the form of a trajectory of joint angles. This trajecory cannot be directly compared with the recorded motion $\boldsymbol{v}^M$, as there could be differences in kinematics and time. So different measures are used for $\boldsymbol{v}$. First of all the path of the hand is used, based on the forward kinematics: $\boldsymbol{x} = f(\boldsymbol{q})$. For a seven degrees of freedom arm it also makes sense to use the arm angle (θ, see section 2.5), because together with $\boldsymbol{x}$ it uniquely describes the arm configuration and it is similar for different arm or robot models [43]. Lastly the total movement time is used, to make sure that the motions have the same duration

$$\boldsymbol{v} = \begin{bmatrix} f(\boldsymbol{q}) \\ \theta \\ T \end{bmatrix}. \tag{5.8}$$

A higher weighting is given to T, as this is a single variable instead of a trajectory.

Derivative-free optimization

For the upper level optimization a direct search method is used because an analytic gradient of the lower level is not available and the evaluations are expensive and noisy. Direct search methods only use the scalar output of the function that is to be optimized and therefore do not require the gradient. A number of algorithms found in literature are described below:

Nelder-Mead Simplex A geometrical figure in the N-dimensional space consisting of N+1 vertices is made around the initial guess. Depending on the function output at the N+1 points the worst point is replaced.

COBYLA Developed by M. J. D. Powell in 1994 [32] this algorithm makes a linear approximation of the objective and constraint functions. A linear programming problem is solved to make a guess for the next step. The output at each step is used to improve the approximation and the algorithm finishes when the step size is sufficiently small. It can handle (in)equality constraints, which can be used to fix the sum of weights α_i.

BOBYQA Also developed by Powell [33] and short for Bound Optimization BY Quadratic Approximation. Instead of linear, a quadratic approximation of the objective function is made. Mombaur et al. [27] use this for finding cost functions weights as it performs significantly faster than a Nelder-Mead implementation.

For a linear-weighted combination of cost functions, only the relative size of the cost functions is important. To fix this redundancy when optimizing the weights, a constraint can be added that requires the sum of the weights to be constant. Alternatively one of the weights can be fixed, to make the other weights change relative to that. When this one fixed weight ends up to be unimportant, all the other weights will increase.

5.3.2. Lower level

For the lower level the forward optimal control method of the previous chapter is used. For this optimization both the YuMi and the human model can be used. When doing the optimization with the human model the cost functions have a physically meaningful interpretation. But the implementation on the YuMi could lead to unnatural looking motions because of the kinematic and dynamic differences. Previous researches solved this problem by carefully scaling the weights depending on the particular robot as in Clever et al. [10]. On the other hand, when doing the optimization with the YuMi model, the found cost functions directly lead to human-like motions when applying them on the real robot and scaling of the weights is not necessary.

One of the main difficulties with the bilevel optimization is choosing the right stopping criteria of the lower level optimization. The first important parameter to set is the `ConstraintTolerance`, based on the magnitude of output of the constraint functions. It is found that below 1e-2 the output for the continuity constraints is low enough to have no difference between single shooting and multiple shooting. This means that for single shooting on the whole trajectory with the same controls, the arm follows the same trajectory as for the multiple shooting. To make sure the constraint for the end-effector start and final target is also sufficiently small, the tolerance is set at 1e-3. The second important parameter is the `OptimalityTolerance`, based on the first-order optimality measure calculated from the gradient of the Lagrangian, see Equation 4.17. This gives a measure on the optimality of the solution. Because the ode-solver accumulates errors and a part of the gradients is calculated numerically, the first-order optimality is set at 1e-2.

5.4. Choice of cost functions

The cost functions are selected based on previous research. Table 5.1 shows the functions that were chosen.

Table 5.1: Cost functions used for the inverse optimal control

Angle acceleration	$\int_0^T \ddot{\boldsymbol{q}}^{\mathrm{T}} \ddot{\boldsymbol{q}} dt$
Angle jerk	$\int_0^T \dddot{\boldsymbol{q}}^{\mathrm{T}} \dddot{\boldsymbol{q}} dt$
Hand acceleration	$\int_0^T \ddot{\boldsymbol{x}}^{\mathrm{T}} \ddot{\boldsymbol{x}} dt$
Geodesic	$\int_0^T \dot{\boldsymbol{q}}^T \boldsymbol{M}(\boldsymbol{q}) \dot{\boldsymbol{q}} dt$
Energy	$\int_0^T \sum \lvert \dot{q}_i \tau_i \rvert dt$
Time	$T = \int_0^T 1 dt$

5.4.1. Smoothness

Flash and Hogan [16] achieve smoothness of the motion by minimizing speed, acceleration and jerk of the hand and the joint angles. Albrecht et al. [2] uses minimization of hand jerk and joint jerk, together with torque change. Sylla et al. [38] adds the Geodesic criterion minimizing joint velocities squared times the inertia matrix as

$$U_{geodesic} = \dot{\boldsymbol{q}}^T \boldsymbol{M}(\boldsymbol{q}) \dot{\boldsymbol{q}}. \tag{5.9}$$

It prefers the shortest path in joint space and in their work it is shown to be a good alternative for acceleration and jerk minimization because it also maximizes joint smoothness and it has a large contribution to the total cost of the solution.

5.4.2. Energy

Minimization of energy was shown to well describe human arm trajectories in combination with a smoothness criterion [5] [38]. Energy usage from torques is calculated as

$$W = \int_0^T \dot{\boldsymbol{q}}^T \boldsymbol{\tau} dt. \tag{5.10}$$

This can lead to a negative value if the joint speed has the opposite direction of the joint torque, meaning energy is absorbed instead of consumed. It was found that this is unrealistic for a human, as moving in the opposite direction of the torque also consumes energy. The joint torques for free arm motions are mainly caused by gravity and there is probably a difference in working with or against gravity, but for now this is assumed to be equal so the absolute work of torques is used

$$W_{abs} = \int_0^T \sum_{i=1}^{n} |\dot{q}_i \tau_i| dt. \tag{5.11}$$

The absolute makes the function non-smooth and the derivative at $\dot{q}_i \tau_i = 0$ does not exist. This makes the optimization highly unstable and therefore difficult to converge. To solve this the absolute function can be approximated by

$$|x| \approx \sqrt{(x)^2 + \mu^2}. \tag{5.12}$$

An alternative approximation was proposed by Bagul [3]

$$|x| \approx (x) \cdot \tanh\left(\frac{x}{\mu}\right). \tag{5.13}$$

Figure 5.1 shows both approximations and their derivative. The latter was chosen because of its better approximation in terms of accuracy, especially for $x = 0$. The error with respect to the absolute function was shown to be

$$\left| |x| - x \cdot \tanh\left(\frac{x}{\mu}\right) \right| < \mu. \tag{5.14}$$

A value of $\mu = 0.01$ was chosen as this would lead to sufficient fast performance while lower values for μ would not lead to different results.

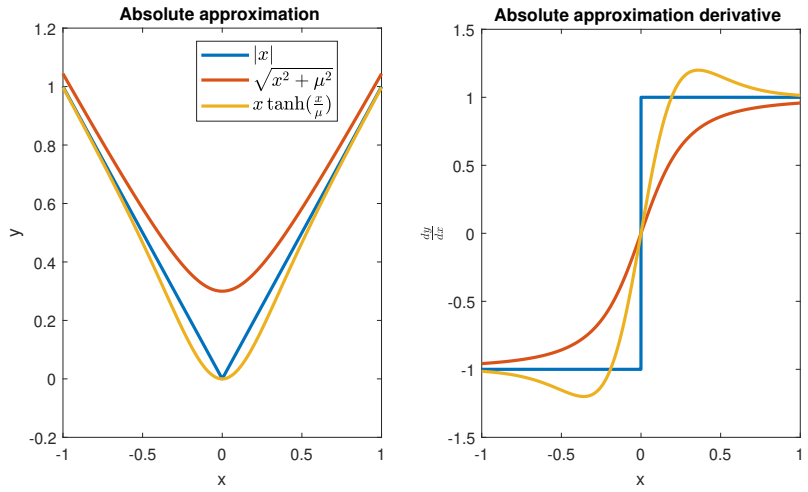


Figure 5.1: Approximation of the absolute function for $\mu = 0.3$. Both functions approximate $|x|$ for $\mu \to 0$.

5.4.3. Time

Sylla et al. [38] use a fixed end time, which is sensible because according to Biess et al. [7], the geometric and temporal properties of human arm motion are decoupled. But including the movement duration into the optimization makes the final optimal motions more realistic, as they will have the correct duration and velocities. It was found that the cost functions with a quadratic term tends to increase the time indefinitely, as the square function grows faster than the integrated time. For this reason a time minimization criteria is included.

5.5. Optimization results

This chapter shows the results of the bilevel optimization for both the YuMi and the human model. For both cases first the COBYLA algorithm is used as it is less dependent on the initial guess and it can include a equality constraint, in this case used for fixing the sum of weights to a constant value. Then the optimization is ran again with the initialization based on the best result of the previous run, but now the BOBYQA algorithm is used and the value of the first weight is fixed. The four recorded trajectories of chapter 3 are used, the first two with a right to left motion and the other two from left to right.

5.5.1. YuMi model

The bilevel optimization is first performed using the YuMi model.

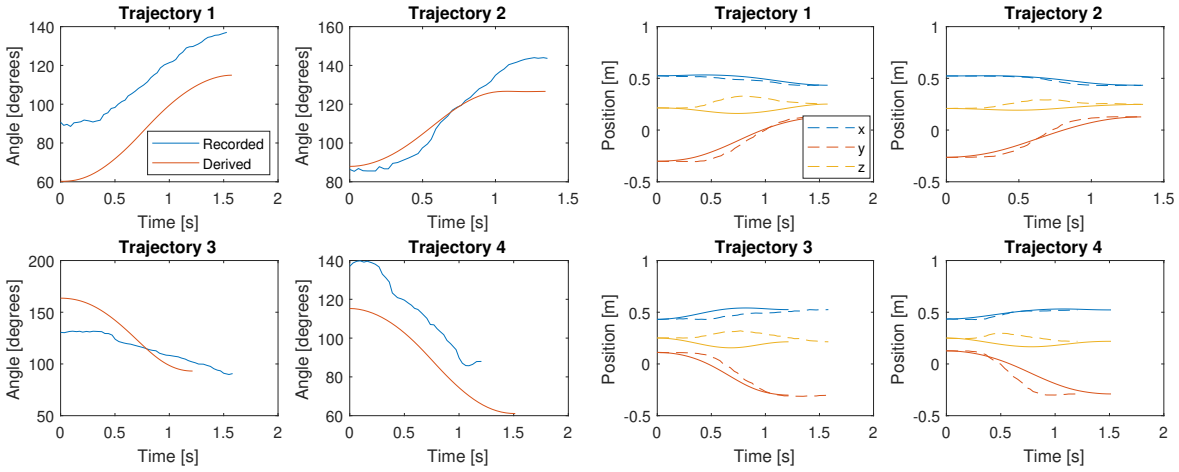
COBYLA

The weights are all initialized as one and the COBYLA algorithm is used for the upper level. Table 5.2 shows the weights found for the four trajectories. The last column shows the error with the recorded trajectory. As explained in subsection 5.3.1 this error is based on the difference in end-effector position/rotation, arm angle and time.

Table 5.2: Resulting weights for the trajectories, with the YuMi model, COBYLA and $\boldsymbol{\alpha}_0 = [1,1,1,1]^T$

	T	U_{energy}	$U_{geodesic}$	$U_{\ddot{x}}$	RMSE
1	1.3750	0.8794	0.8314	0.9478	17.4452
2	1.8024	0.0065	0.9177	1.2734	6.6303
3	2.1774	-0.0201	1.1383	0.7044	21.4045
4	1.3750	0.8750	0.8750	0.8750	18.2898

The results differ greatly between the four trajectories. Figure 5.2 shows the trajectories of the solutions compared to their reference recording.



(a) Arm angle

(b) TCP path, with the recorded reference shown as dashed lines

Figure 5.2: Closest fit solution as found by the bilevel optimization, with the YuMi model, COBYLA and $\boldsymbol{\alpha}_0 = [1,1,1,1]^T$.

The arm angle follows the correct direction of the reference, being lower when the arm is on the right and higher when the arm is on the left. A lower value for the arm angle indicates the elbow is in a higher position, see chapter 2.5. For trajectory 1 and 4 the arm angle is too low and for trajectory 3 and 4 the duration of the motion is incorrect.

From these results and a number of other trials it is clear that the weights tend to end up in local minima, so it is necessary to give a good initial guess. Trajectory 2 has the closest fit to its reference, with a low weight

for U_{energy} so for the next optimization this weight is initialized as 0.02. It is also found that the energy and Geodesic criteria have no influence on the duration of the motion. This is in correspondence with Biess et al. [7], where it was shown that geometric and temporal properties of human arm motion are decoupled. As expained before, $U_{\ddot{x}}$ lengthens the time, so its weight should be about 0.7 times the weight of T.

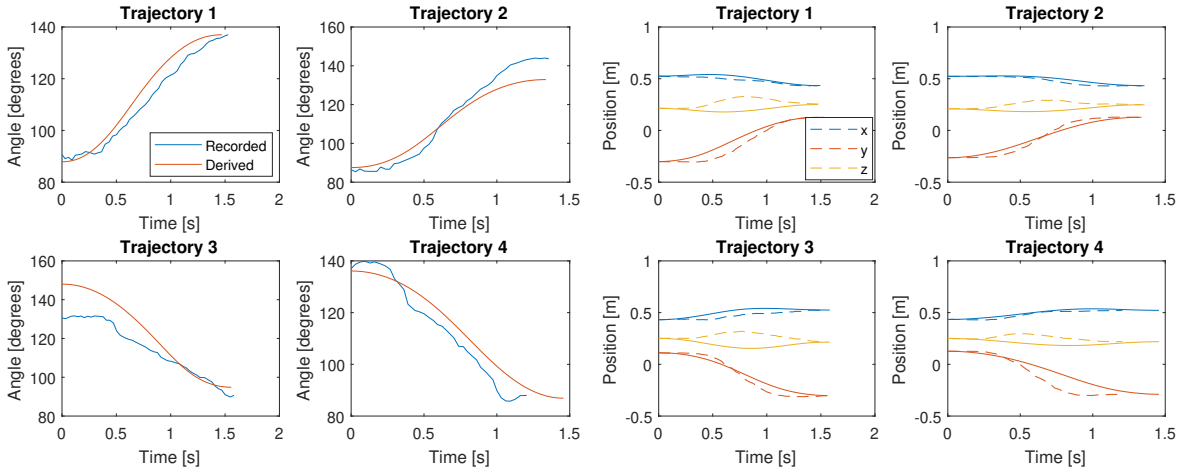
BOBYQA

Now BOBYQA is used and the weights are initialized as $\boldsymbol{\alpha}_0 = [1, 0.02, 1, 0.7]^T$, where the first weight is fixed. The results are as displayed in Table 5.3.

Table 5.3: Resulting weights for the trajectories, with the YuMi model, BOBYQA and $\boldsymbol{\alpha}_0 = [1, 0.02, 1, 0.7]^T$

	T	U_{energy}	$U_{geodesic}$	$U_{\ddot{x}}$	RMSE
1	1	0.0110	1.0304	0.7001	7.0847
2	1	0.0200	0.9691	0.5207	4.6230
3	1	0	1.0957	0.7222	10.1591
4	1	0.0196	1.2866	0.7035	7.5174

The results are now more consistent and the RMSE is lower. Figure 5.3 shows the arm angle and TCP trajectories of the final solution.



(a) Arm angle

(b) TCP path, with the recorded reference shown as dashed lines

Figure 5.3: Closest fit solution as found by the bilevel optimization, with the YuMi model, BOBYQA and $\boldsymbol{\alpha}_0 = [1, 0.02, 1, 0.7]^T$.

These results show a much better fit with the reference data. Table 5.4 shows the contributions of the weighted cost functions to the total cost.

Table 5.4: Contributions, YuMi model

	T	U_{energy}	$U_{geodesic}$	$U_{\ddot{x}}$
1	0.6079	0.0160	0.1721	0.2040
2	0.6013	0.0325	0.1769	0.1893
3	0.6260	0	0.1725	0.2016
4	0.5717	0.0276	0.2062	0.1944

Final result

Taking the average of the weights for the four trajectories has no physical meaning because the costs are relative to each other. Therefore the weights of 2 are chosen, because its low RMSE. The weights are also tested on all other trajectories and this one performs best on average. The final composite cost function is therefore

$$U = T + 0.020\, U_{energy} + 0.97\, U_{geodesic} + 0.52\, U_{\ddot{x}}. \tag{5.15}$$

Figure 5.4 shows a trajectory for a new set of starting and ending targets using this cost function. Figure 5.5 shows the corresponding joint angles and angular speeds. What stands out are the asymmetrical velocity profiles, which is a characteristic of human motions [28].

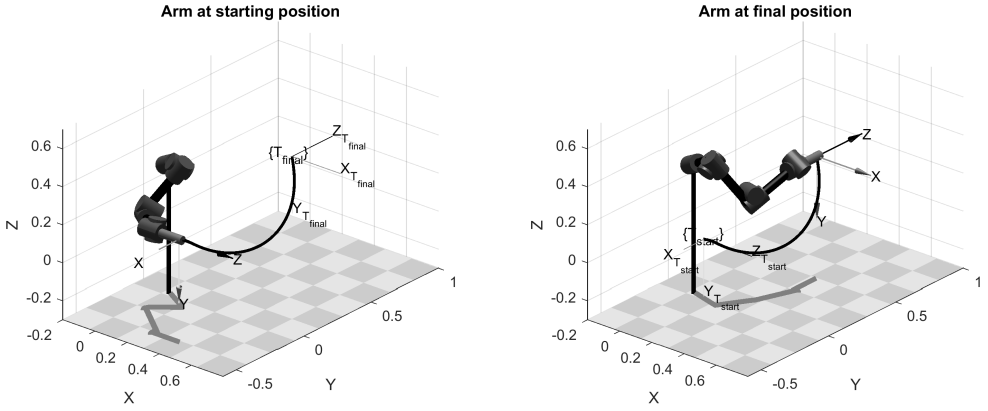


Figure 5.4: An optimal trajectory from the final composite cost function.

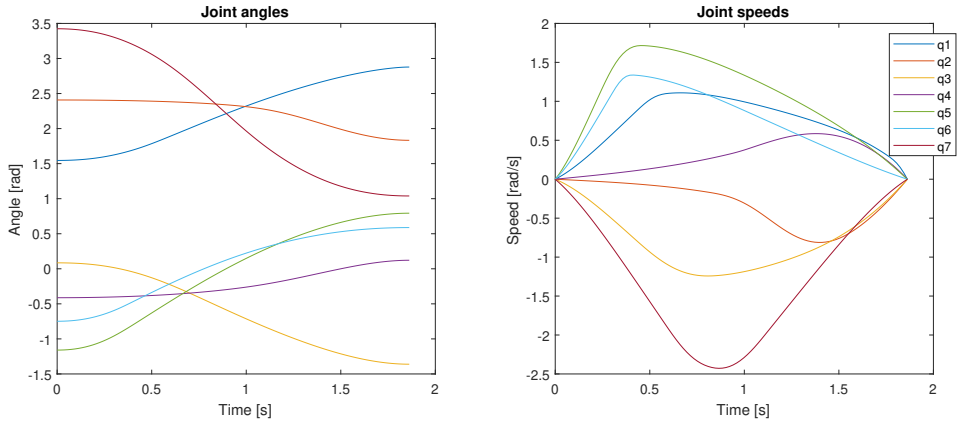


Figure 5.5: Joint angle and angular speeds optimal trajectory from the final composite cost function.

5.5.2. Human model

The bilevel optimization using the human model is also done with both COBYLA and BOBYQA for the upper level.

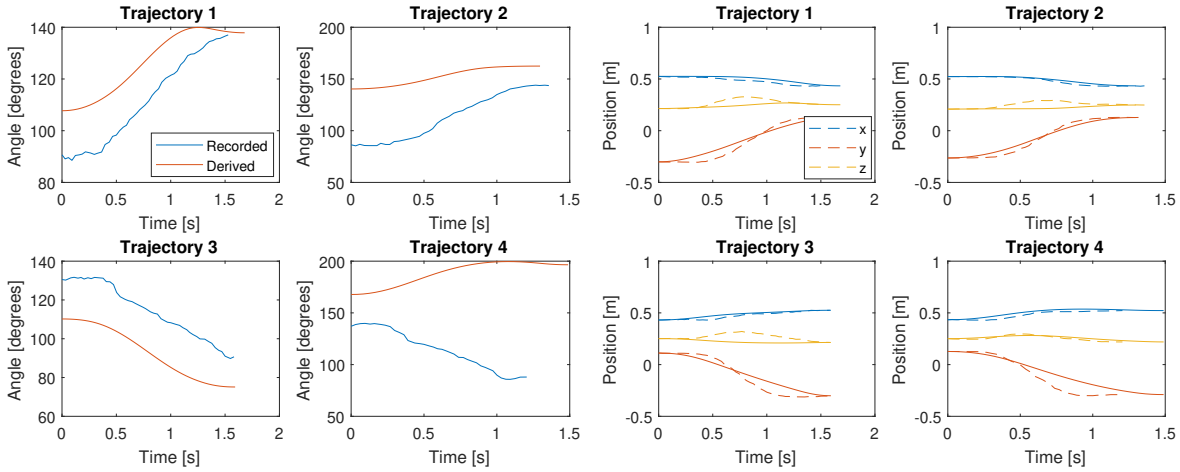
COBYLA

The result from using COBYLA and initializing the weights as $\boldsymbol{\alpha}_0 = [0.13, 0.4, 0.4, 0.1]^T$ are shown in Table 5.5. The initialization was chosen because for $\boldsymbol{\alpha}_0 = [1,1,1,1]^T$ the arm angle would always go to 180°. This is probably caused by the lower mass of the human arm in comparison to the YuMi arm. U_{energy} and $U_{geodesic}$ are both dependent on the mass, so a higher weighting is necessary compared to T and $U_{\ddot{x}}$.

Table 5.5: Resulting weights for the trajectories, with the human model, COBYLA and $\boldsymbol{\alpha}_0 = [0.13, 0.4, 0.4, 0.1]^T$

	T	U_{energy}	$U_{geodesic}$	$U_{\ddot{x}}$	RMSE
1	0.1052	0.2753	0.5928	0.0568	7.0292
2	0.1654	0.0465	0.7541	0.0640	20.7094
3	0.1211	0.5132	0.3366	0.0591	12.4175
4	0.1588	0.3863	0.3815	0.1034	72.2099

Figure 5.6 shows the trajectories and their reference.



(a) Arm angle

(b) TCP path, with the recorded reference shown as dashed lines

Figure 5.6: Closest fit solution as found by the bilevel optimization, with the human model, COBYLA and $\boldsymbol{\alpha} = [0.13, 0.4, 0.4, 0.1]^T$.

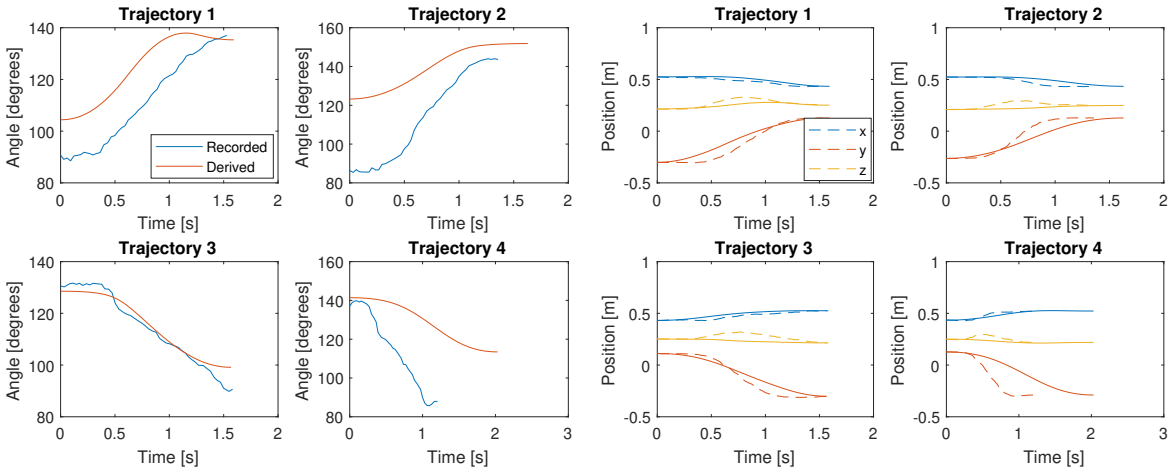
As with the YuMi model, the derived arm angle follows the direction of the reference, except for trajectory 4. There is, however, a relatively large offset for all trajectories.

BOBYQA

Based on the results of trajectories 1 and 3, the weights are now initialized as $\boldsymbol{\alpha}_0 = [0.1, 0.3, 0.6, 0.05]^T$ and BOBYQA is used. The results are shown in Table 5.6 and Figure 5.7.

Table 5.6: Resulting weights for the trajectories, with the human model, BOBYQA and $\boldsymbol{\alpha}_0 = [0.1, 0.3, 0.6, 0.05]^T$

	T	U_{energy}	$U_{geodesic}$	$U_{\ddot{x}}$	RMSE
1	0.1	0.2735	0.6683	0.0109	5.5537
2	0.1	0.0959	0.6004	0.0515	13.7672
3	0.1	0.1195	0.5794	0.0500	3.6880
4	0.1	0.3011	0.6030	0.0797	37.9151



(a) Arm angle

(b) TCP path, with the recorded reference shown as dashed lines

Figure 5.7: Closest fit solution as found by the bilevel optimization, with the human model, BOBYQA and $\boldsymbol{\alpha}_0 = [0.1, 0.3, 0.6, 0.05]^T$.

Compared to the COBYLA optimization the RMSE went down for all trajectories. But, except for trajectory 3, there is still not a great fit for the arm angle and there is no noticeable improvement in the TCP path. Table 5.7 shows the contributions, which are more consistent between the trajectories than the weights.

Table 5.7: Contributions, human model

	T	U_{energy}	$U_{geodesic}$	$U_{\ddot{x}}$
1	0.1653	0.5346	0.2876	0.0125
2	0.2611	0.2977	0.3883	0.0529
3	0.2491	0.3349	0.3532	0.0628
4	0.1902	0.5513	0.2319	0.0266

Final result

As the weights found with trajectory 3 lead to the closest fit the final cost function is

$$U = 0.1\ T + 0.12\ U_{energy} + 0.58\ U_{geodesic} + 0.05\ U_{\ddot{x}}. \tag{5.16}$$

5.6. Conclusion

The bilevel optimization explained in this chapter manages to find a close fit with the reference motions when using the YuMi model and the resulting weights are consistent between the four optimizations. The optimization using the human model is less successful. Changing the ratio between the temporal and spatial cost functions improves the results, but a consistent close fit is still not found. A reason for this could be inaccuracies in the human model, but this cannot be stated for certain. It is therefore chosen to continue in the next chapter with the weights found with the YuMi model.

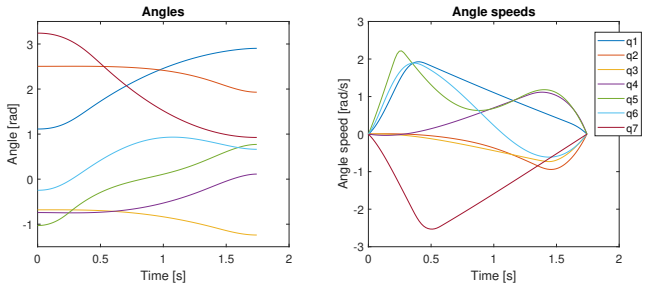
6 Validation

6.1. Introduction

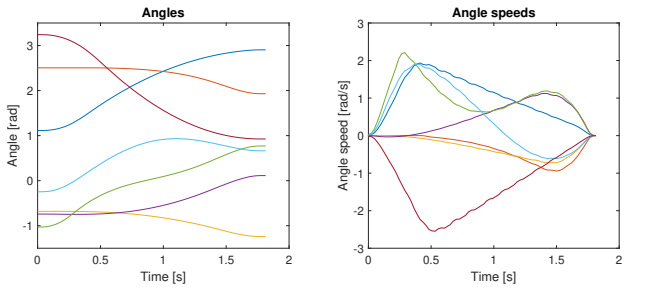
In this chapter the result of the inverse optimal control is validated with an experiment. Based on the reasons explained in section 5.3.2 the cost function that is derived with the YuMi model is used to generate motions towards new targets. The experiment should indicate if these motions are an improvement over the existing motion planning strategies used for the YuMi robot.

6.2. Implementation on the real robot

The YuMi robot is controlled with motion commands specified in a RAPID function. The optimal joint trajectories computed in Matlab are exported to a RAPID file. The `MoveAbsJ` command is used to specify the joint targets. To ensure the correct joint angular speeds the time of each move-command is set. Figure 6.1b shows the trajectory as recorded from the YuMi, it is similar to the derived trajectory of Figure 6.1a, showing that the correct velocities are established.



(a) Derived with optimal control



(b) Joint angles recorded from the YuMi, the joint angle angular speeds are calculated numerically from the joint angles

Figure 6.1: The same trajectory first as derived and then transferred to the YuMi and recorded with the robot controller.

6.3. Human experiment

To determine if the method for generating trajectories does indeed leads to improved collaboration an experiment is performed where it is compared to two existing motion planning strategies. Human participants are asked to rate the shown motions on a number of scales indicating anthropomorphism and feeling of safety.

6.3.1. Questionnaire

In Bartneck et al. [4] questionnaires are developed to measure Human-Robot Interaction (HRI). Five categories are specified: anthropomorphism, animacy, likeability, perceived intelligence and perceived safety. Each category has a number of semantic differential scales that the participant has to give a score to. These questionnaires are aimed to be a standardized tool to measure HRI. They are well tested and the standardization makes it possible to compare results between researches. As the authors state the results of questionnaires do not offer an absolute value for a certain category but can enable a comparison between different robots or different configurations of a robot. For our case they can be used to compare the newly derived optimal motions with an already available motion planning system.

Kulić and Croft [21] perform a similar experiment with an industrial six degrees of freedom robot arm performing motions with different planning strategies. The participants were asked to rate motions on if they felt anxious, calm, surprised and how much it kept their attention. A Likert scale was used with scores from 1 to 5. The participants arousal was also estimated using skin conductance and heart rate measurements. The correlations between the participant-reported responses and the estimated arousal was found to be statistically significant for all four categories, indicating that the questionnaire is in correspondence with the objective measurements.

These works show that a valid comparison between the motion planners can be made with a questionnaire. First of all it was chosen to compare on anthropomorphism, because of the previously made assumption that more human-like motions lead to improved collaboration. Secondly the feeling of safety was estimated, as this is an important factor in a manufacturing setting and a strong indication for comfort. It is important to state that the feeling of safety is only a subjective feeling, while the actual safety of the robot needs to be ensured with other measures.

Appendix A shows one of the questionnaires that was used. The scales *Fake - Natural* and *Moving rigidly - Moving elegantly* are used to determine the anthropomorphism of the motions and *Anxious - Relaxed* and *Quiescent - Surprised* to see how safe the participant feels. Between the participants the order of the scales is mixed.

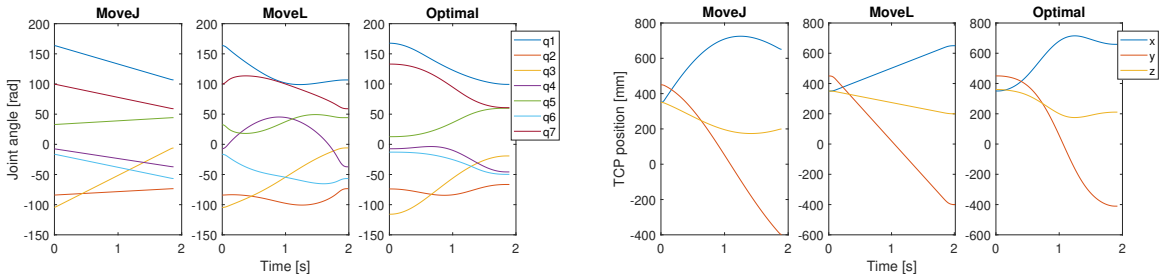
6.3.2. Motions

The YuMi robot at the moment has two implementations for making point-to-point motions: MoveL, with paths linear interpolated in Cartesian space and MoveJ, linear interpolated in joint space. These two are compared with the implementation using the forward optimal control scheme as developed in this thesis, with the cost function identified with inverse optimal control. This motion will be referred to as 'Optimal'.

Figure 6.2a shows the joint angle trajectories of the three motion planning implementations for one of the sets of motions used in the experiment. As expected, MoveJ has linear interpolated paths in joint space and MoveL has curved paths. Interestingly Optimal is somewhat in between. For the Cartesian trajectory of the end-effector MoveL has linear interpolated paths, see Figure 6.2b. MoveJ and Optimal both show curved paths, with Optimal following a less rounded curve.

6.3.3. Experiment procedure

The participant is first given a description of the experiment and asked to sign a consent form. The participant is sitting in front of the robot. First the three different motions are shown one after the other. Then the motions are shown again, but now with time in between to fill in the four scales. This procedure is done for four sets of motions, with different start and end targets for the robot arm, so in total 12 motions are shown. The order of the motions is mixed for every new set of motions.



(a) Joint angles

(b) End-effector positions

Figure 6.2: Comparison of the three motion implementations.

6.3.4. Results and discussion

From the average scores displayed in Figure 6.3 it is clear that the Optimal motions are perceived as more natural and more elegant than the other two implementations. The participants also feel more relaxed and less surprised compared to the MoveL motions. Between the MoveJ and Optimal motions they feel slightly more relaxed and similarly surprised.

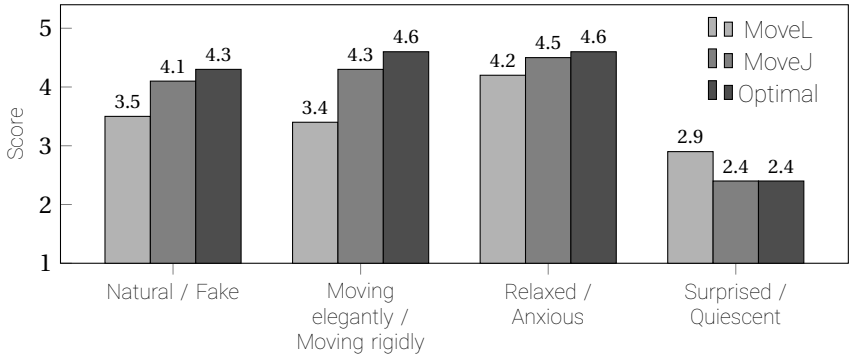


Figure 6.3: Average scores given for the three motion planning strategies.

Table 6.1 shows the percentage of every choice on the scales. Here a similar pattern is visible, with the Optimal motion scoring higher for *natural*, *moving elegantly* and *relaxed*.

For experiments with a questionnaire the Cronbach's alpha is often calculated to measure the consistency between the answers. It is calculated as

$$\alpha = \frac{K}{K-1}\left(1 - \frac{\sum_{i=1}^{K} \sigma_{Y_i}^2}{\sigma_x^2}\right), \tag{6.1}$$

where K is the number of criteria, $\sum_{i=1}^{K} \sigma_{Y_i}^2$ the sum of the variances of the scores between the test subjects and σ_x^2 the variance of the sum of the scores between the subjects. For the responses to the three motion strategies it is 0.50 if we include all four scales. If the last scale, *Quiescent - Surprised*, is omitted the alpha-value increases to 0.78, indicating that for internal consistency in measuring human preference it is better to exclude this scale. This could be because this scale seems flipped with respect to the other scales, leading to a certain level of confusion for the participants. The consistency between the two scales indicating anthropomorphism is even higher at 0.85.

Table 6.2 shows the correlation between the responses to the four scales for all motions. *Natural - Fake* and *Moving rigidly - Moving elegantly* have a strong positive correlation and a motion perceived as natural and elegant leads to a more relaxed feeling for the participants. The consistency of *Quiescent - Surprised* is low with all other scales.

Using an alpha level of 0.05, a dependent-samples t test was conducted to see if the results between the motion strategies are significant. For the *Natural - Fake* scale between MoveL and Optimal there is a significant

Table 6.1: Answers to the scales expressed in percentages

(a) MoveL

	1	2	3	4	5	
Fake	12.5	10.0	22.5	30.0	25.0	Natural
Moving rigidly	7.5	20.0	20.0	30.0	22.5	Moving elegantly
Anxious	0	10.0	12.5	30.0	47.5	Relaxed
Quiescent	22.5	15.0	22.5	30.0	10.0	Surprised

(b) MoveJ

	1	2	3	4	5	
Fake	0	7.5	10.0	45.0	37.5	Natural
Moving rigidly	0	5.0	15.0	27.5	52.5	Moving elegantly
Anxious	0	2.5	2.5	37.5	57.5	Relaxed
Quiescent	35.0	25.0	12.5	22.5	5.0	Surprised

(c) Optimal

	1	2	3	4	5	
Fake	0	2.5	2.5	55.0	40.0	Natural
Moving rigidly	0	0	5.0	27.5	67.5	Moving elegantly
Anxious	0	0	5.0	32.5	62.5	Relaxed
Quiescent	32.5	27.5	7.5	30.0	2.5	Surprised

difference: $t(39) = 4.18, p < 0.05, d = 0.66$. Between MoveJ and Optimal the difference for this scale is not significant. But for *Moving rigidly - Moving elegantly* between MoveJ and Optimal the result is significant: $t(39) = 2.48, p < 0.05, d = 0.39$.

The difference for *Anxious - Relaxed* between MoveL and Optimal is significant as well: $t(39) = 2.38, p < 0.05, d = 0.38$, but not between MoveJ and Optimal. For *Quiescent - Surprised* between MoveL and Optimal the difference is significant: $t(39) = -2.15, p < 0.05, d = 0.34$, but not between MoveJ and Optimal.

6.3.5. Anthropomorphism

The first two scales are measuring anthropomorphism, so if we combine them for MoveJ (M: 4.20, SD: 0.89) and Optimal (M: 4.48, SD: 0.64) we can again do the t test: $t(79) = 2.58, p < 0.05, d = 0.29$, which indicates that Optimal is perceived as significantly more anthropomorphic than MoveJ.

6.3.6. Perceived safety

When it comes to the last two scales the results can be combined to give a measure for perceived safety, if the results of *Quiescent - Surprised* are flipped. The t test shows the difference is significant between the MoveL and Optimal motions: $t(79) = 3.19, p < 0.05, d = 0.36$. The participants, as expected, do not experience a significant difference in safety between MoveJ and Optimal. MoveJ is also a significant improvement over MoveL: $t(79) = 2.54, p < 0.05, d = 0.40$.

Table 6.2: Correlations between the responses

	Fake / Natural	Rigidly / Elegantly	Anxious / Relaxed	Quiescent / Surprised
Fake / Natural	1.0			
Rigidly / Elegantly	0.7421	1.0		
Anxious / Relaxed	0.4488	0.4323	1.0	
Quiescent / Surprised	0.0497	-0.0047	-0.2215	1.0

6.4. Conclusion

The results in this chapter validate the method that was used to generate the optimal motions. The participants experience a significant improvement over the MoveL motions on all measured scales. Compared to the MoveJ motions the differences are smaller, but when it comes to anthropomorphism there is a significant improvement. The improvement of feeling of safety can not be confirmed with these results, which could be caused by the inconsistency between the two scales. Another reason could be that the participants already feel sufficiently safe with the MoveJ motions, as indicated by the high score for *Relaxed*.

7

Conclusion

This thesis shows inverse optimal control can be used to derive a cost function that enables the generation of motions similar to those of humans. The optimal control method using this cost function is generalizable to new targets. The experiment with human participants shows the positive effects on how the optimal motions are perceived. The approach helps towards increasing human acceptance of robot motions and thereby towards improving human-robot collaboration.

7.1. Future work and applications

Future research should be done on the task dependency of cost functions, as here only point-to-point motions are considered. More complex tasks and situations might require different cost functions. There might also be differences between humans, as not everyone moves in the same way.

A useful extension of this work would be to develop an online implementation for the forward optimal control. A user of the robot would then only have to specify an end-effector target and the internal controller would generate an optimal path towards it. The same cost function as derived in this work can be used for the optimization. This would require a substantial increase in the speed of the optimization, as the current method is too time consuming for an online application.

Another interesting idea is to vary the cost function based on the proximity of the human coworker. For example, when there is no human nearby a cost function minimizing energy or time can be used. Then when nearby, the by humans preferred cost function is applied.

A

Appendix: Questionnaire

Questionnaire

Participant number: ____

Trial: ____

Motion 1

Please rate your impression of the robot:

Moving rigidly 1 2 3 4 5 Moving elegantly

Please rate your emotional state:

Quiescent 1 2 3 4 5 Surprised

Please rate your emotional state:

Anxious 1 2 3 4 5 Relaxed

Please rate your impression of the robot:

Fake 1 2 3 4 5 Natural

Motion 2

Please rate your impression of the robot:

Moving rigidly 1 2 3 4 5 Moving elegantly

Please rate your emotional state:

Quiescent 1 2 3 4 5 Surprised

Please rate your emotional state:

Anxious 1 2 3 4 5 Relaxed

Please rate your impression of the robot:

Fake 1 2 3 4 5 Natural

Motion 3

Please rate your impression of the robot:

Moving rigidly 1 2 3 4 5 Moving elegantly

Please rate your emotional state:

Quiescent 1 2 3 4 5 Surprised

Please rate your emotional state:

Anxious 1 2 3 4 5 Relaxed

Please rate your impression of the robot:

Fake 1 2 3 4 5 Natural

Figure A.1: One of the questionnaires used in the experiment

B

Appendix: Conference Paper

Based on the results of this research a paper was written and submitted to an international conference. It is included in this Appendix.

How Inverse Optimal Control Can Help to Increase Human Acceptance of Robot Motions in Collaborative Manufacturing Tasks

Eert Hoogerwerf[1] Mukunda Bharatheesha[2] Debora Clever[3]

***Abstract*— Collaboration between humans and robots is an important aspect of Industry 4.0. This can be improved by incorporating human-like characteristics into robot motion planning. It is assumed that humans move optimal with respect to a certain objective or cost function. To find this function we use an inverse optimal control approach for finding what linear weighted combination of physically interpretable cost functions best mimics human point-to-point motions. A bilevel optimization is done, where the upper level compares the optimal result of the lower level with the reference motion. Two depth cameras are combined in a setup to record this reference motion.**

The resulting weighted cost functions are then used to generate new motions for a seven degrees of freedom robot arm. The results of the experiment show that humans experience these motions as more anthropomorphic and feel at least equally as safe compared to existing motion planning strategies.

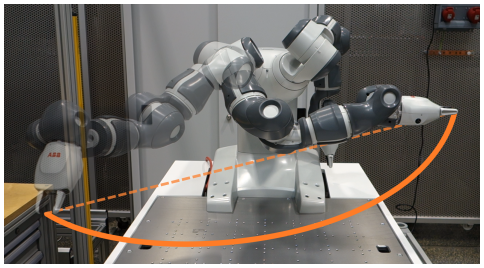


Fig. 1. The YuMi robot performing an optimal motion based on the cost function derived with inverse optimal control using human recorded motion. Point-to-point motions on this robot are currently performed by linearly interpolating a path in joint or Cartesian space.

I. INTRODUCTION

In an environment where humans and robots are working together a mutual understanding of each other is crucial. One way to improve this is by humanizing Human-Robot Interaction (HRI), which means that the robot not only tries to understand the human's actions and intentions, but also that it uses human-like features to communicate its own [1]. This makes sense because we are highly trained in interacting with other humans.

Human and robot arm motion planning has to deal with a high level of redundancy and out of the infinite amount of possibilities it is believed that humans choose a path that is optimal with respect to a certain objective function [2]. This objective function is situation dependent and generally unknown. By doing an inverse optimization using human measurements these objective functions can be determined. The results can help better understand human motion and applying optimal control with the found objective functions can enable human-like motion characteristics for robots.

A. Related work

A bilevel optimization approach was introduced in [3] for finding the optimization criteria humans use for a locomotion tasks. The lower level uses optimal control based on weighted cost functions. These weights are supplied by the upper level and optimized based on the difference with recorded walking trajectories. The combination of five criteria, involving minimization of time, acceleration and orientation towards the target, gave the best results. In [4] the approach was extended for the generation of gait trajectories in complex environments using an objective function identified from human captured walking motions.

A similar bilevel approach was used for arm motions in [5] for a two dimensional pointing task. The cost functions that had the biggest contributions where those that minimized energy consumption and angle acceleration. The same approach was later used for a more realistic three dimensional task by [6], where it was found that a geodesic and an energy criteria best mimicked human motion.

Arm reaching motions in collaboration tasks were researched by [7]. Two participants performing a number of pick-and-place tasks while standing next to each other were recorded. For the inverse optimal control they use cost functions based on distances to the other participant and functions achieving smoothness. The trajectories planned with the recovered weights are found to better resemble human motion than with manually tuned weights.

In [8] the lower level is replaced with its necessary conditions of optimality, the Karush–Kuhn–Tucker (KKT) conditions, which are then used as constraints for the upper level. This transforms the bilevel optimization into a one-level optimization. It was applied for reaching motions and a combination of torque change and joint jerk minimization matched the human trajectories best. The found cost functions were used to plan optimal motions for a humanoid robot, which was able to match human recorded motions closely.

Task dependance of cost function was examined for a

[1]Eert Hoogerwerf is with the Faculty of Mechanical, Maritime and Materials Engineering, Delft University of Technology, 2628 CD Delft, The Netherlands t.e.hoogerwerf@student.tudelft.nl

[2]Mukunda Bharatheesha is with the Faculty of Mechanical, Maritime and Materials Engineering, Delft University of Technology, 2628 CD Delft, The Netherlands m.bharatheesha@tudelft.nl

[3]Debora Clever is with ABB Corporate Research Center, 68526 Ladenburg, Germany debora.clever@de.abb.com

box moving task using time dependent weights [9] with the Inverse-KKT approach. This is also a one-level optimization, but instead of the approach of [8] the KKT-conditions are used in the cost function.

Compared to KKT-based apporaches, the bilevel approach is more flexible as it does not require the optimization gradient in analytical form. But it is computationally more demanding as for every upper level evaluation the lower level optimization has to be completed [10].

From this short overview it is clear that inverse optimal control has been used to determine human motion objectives and the application robot motion planning has been investigated as well. But the application to collaborative robots and the resulting influence on Human-Robot Interaction is still open for research.

B. Contribution

The contribution of this paper is an inverse optimal control application to a seven degrees of freedom case, validated with an experiment that shows the positive effects on how humans experience the resulting optimal motions. The paper is organized as follows: In section II the theory behind inverse optimal control using the bilevel approach is summarized, followed by our implementation in section III. The results are given in section IV in the form of a composite cost function, which is validated with an experiment to determine how humans experience the resulting motions.

II. INVERSE OPTIMAL CONTROL

With optimal control a certain objective function is minimized to derive a local optimal solution. For inverse optimal control this objective function is to be determined using a known optimal solution, in the case of a robot arm this can be recorded motion of a human. This learned objective function can then be used to achieve human-like characteristics for reaching new targets. This generalization to new situations is an important feature of inverse optimal control.

Because real-time control is not an objective here and to have more flexibility in the choice of cost functions, the bilevel approach is chosen. The upper level optimizes parameters α_i based on the difference between the solution of the lower level and the recorded reference motion. The lower level solves a forward optimal control problem using the parameters supplied by the upper level.

A. Linear combination

One way of learning the objective function U can be by assuming that it is a linear combination of cost functions U_i weighted by parameters α_i that do not change over time. The total cost is then

$$U = \int_0^T \Big[\sum_{i=1}^{n} \alpha_i U_i(\boldsymbol{z}(t), \boldsymbol{u}(t))\Big] dt, \tag{1}$$

with $\boldsymbol{z}$ representing the states and $\boldsymbol{u}$ the controls. The problem of finding the objective function is now reduced to finding the weights α_i. A weight can go to zero if its corresponding cost function does not contribute to the reference motion.

B. Upper level

The inverse optimal control problem can be split into two levels. The upper level optimization is used to determine the weight factors $\boldsymbol{\alpha}$ using a least-squared minimization

$$\min_{\alpha} \sum_{j=1}^{m} ||\upsilon^*(t_j; \boldsymbol{\alpha}) - \upsilon_M(t_j)||^2. \tag{2}$$

With every iteration of the upper level, the lower level is called to derive the optimal trajectory $\boldsymbol{z}(t)$ and controls $\boldsymbol{u}(t)$. From this solution a comparison criteria $\upsilon^*(t)$ is derived, while υ_M is based on the measured motion. Here the motions are compared based on the position of the end-effector, the arm angle and the final time: $\upsilon(f(\boldsymbol{q}), \theta, T)$, see section III-C.

Because the lower level evaluations are expensive and the results can be noisy, a derivative free optimization method is used for the upper level. In this case the BOBYQA algorithm developed by Powell [11]. The system is implemented in Matlab using the NLopt library [12] for the upper level optimization.

C. Lower level

The lower level is an optimal control problem formulated as

$$\min_{z,u,T} \int_0^T \Big[\sum_{i=1}^{n} \alpha_i U_i(\boldsymbol{z}(t), \boldsymbol{u}(t))\Big] dt \tag{3}$$

$$\text{subject to:} \quad \textit{System equation:}\ \dot{\boldsymbol{z}} = \Omega(t, \boldsymbol{z}(t), \boldsymbol{u}(t)) \tag{4}$$

$$\textit{Forward kinematics BC:}\ f(\boldsymbol{q}(0)) = \boldsymbol{x}_0; \tag{5}$$

$$f(\boldsymbol{q}(T)) = \boldsymbol{x}_e \tag{6}$$

$$\textit{Joint angular speeds BC:}\ \dot{\boldsymbol{q}}(0) = 0; \tag{7}$$

$$\dot{\boldsymbol{q}}(T) = 0 \tag{8}$$

The joint angles and joint angular speeds are the states of the system

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix}. \tag{9}$$

This is solved with a direct boundary value problem approach using multiple shooting as developed by Bock and Plitt [13]. In this method the states and controls are first discetized and then optimized. The controls are discretized by making them represent the nodes in piecewise constant, linear or cubic interpolating functions. We choose shape-preserving piecewice cubic interpolation because this leads to a smooth function without the risk of overshoot. The time between the nodes (τ_i) is made variable as well, where a linear constraint ensures that the sum of all τ_i's is equal.

For multiple shooting the timespan is divided into shooting intervals. The states represent the initial values and the final value of the shooting intervals. On every shooting interval the system equation is solved with these initial values. With acceleration chosen as control variable this system equation is

$$\dot{\boldsymbol{z}} = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{u} \end{bmatrix}, \tag{10}$$

with $\boldsymbol{u} = \ddot{\boldsymbol{q}}$. The boundary constraints are enforced using the first and last discrete states. Continuity constraints ensure that

the difference between the end of an interval and the start of the next interval disappears. This leads to a set of initial value problems that can be solved as a nonlinear programming problem using the SQP algorithm. The `fmincon` function from Matlab is used for this, with the `ode45` function for numerical integration of the system equation.

The composite cost function of (1) can be combined with the system equation and be solved simultaneously by the `ode`-solver

$$\begin{bmatrix} \dot{\boldsymbol{z}} \\ \dot{U} \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{u} \\ \sum_{i=1}^{n} \alpha_i U_i(\boldsymbol{z}, \boldsymbol{u}) \end{bmatrix}. \tag{11}$$

This way the accuracy of the solution of the cost is also incorporated in the error control of the `ode`-solver.

The SQP algorithm requires gradients of the objective and constraint functions to determine its search direction and to determine the first-order optimality of the current step. The forward kinematics boundary constraints (5) and (6) are based on the positional and rotational differences of the end-effector with its target. The analytical Jacobian of the arm can be used to efficiently and exactly calculate the gradient of this function towards $\boldsymbol{q}$

$$\dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \qquad \boldsymbol{J}(\boldsymbol{q}) = \left(\frac{\partial x_i}{\partial q_j}\right)_{i,j}. \tag{12}$$

The gradient of the objective function and the gradient of the continuity constraints are calculated numerically using a finite difference method.

III. HUMAN AND ROBOT MOTION

In this section we elaborate our choice of cost functions, describe the models and present our motion recording system.

A. Cost functions

The cost functions are selected based on previous research. Criteria minimizing hand and angular velocity, acceleration and jerk have been used to achieve smoothness of the motion in [14]. In [8] minimization of hand jerk, joint jerk and torque change is considered.

Minimization of energy was shown to be an important criteria for humans [5]. It can be calculated as

$$W = \int_0^T \dot{\boldsymbol{q}}^\mathsf{T} \boldsymbol{\tau} dt, \tag{13}$$

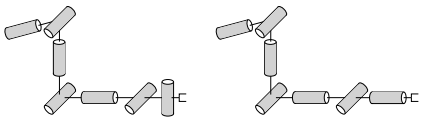
where $\boldsymbol{\tau}$ is the torque in the joints. However, this assumes that moving in the opposite direction of the torque generates energy, while in reality for humans it costs energy. So instead the absolute work of torques is used, where we make the assumption that moving with and against the torque consumes the same amount of energy.

In [6] the geodesic criteria is used, it minimizes joint velocities squared times the inertia matrix. It prefers the shortest path in joint space and maximizes smoothness of the motion. Together with the energy criteria it was shown to lead to realistic human motions.

The cost functions used for the optimization are displayed in Table I. $U_{\ddot{x}}$ was chosen because the squared function tends to increase the movement time. Other squared functions, like $\ddot{\boldsymbol{q}}^\mathsf{T}\ddot{\boldsymbol{q}}$ and $\dddot{\boldsymbol{q}}^\mathsf{T}\dddot{\boldsymbol{q}}$, had their weights go to zero indicating they did not contribute to the upper level minimization.

TABLE I
COST FUNCTIONS USED IN OPTIMIZATION

Time	U_T	$T = \int_0^T 1 dt$
Hand acceleration	$U_{\ddot{x}}$	$\int_0^T \ddot{\boldsymbol{x}}^\mathsf{T}\ddot{\boldsymbol{x}} dt$
Energy	U_{energy}	$\int_0^T \sum \lvert\dot{q}_i \tau_i\rvert dt$
Geodesic	$U_{geodesic}$	$\int_0^T \dot{\boldsymbol{q}}^\mathsf{T}\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}} dt$



(a) Human arm (b) Robot arm

Fig. 2. The human and robot arm can be modeled similarly except for the orientation of the last joint. The differences in offsets between the joints are not shown here.

B. Models

The robot used in this research is the YuMi from ABB. It has two arms, both with seven degrees of freedom, and is intended for collaboration with humans. A model of the YuMi was made with the Robotic Toolbox [15].

Fig. 2 shows the kinematic structure of the human and robot models. Except for the orientation of the last joint, the main difference lies in the offsets between the joints. The human shoulder can be approximated as a ball-and-socket joint [16] and is modeled with three revolute joints with zero offset. Similarly, both the elbow and the wrists joints consist of two revolute joints with zero offsets.

The energy and geodesic cost functions are based on the dynamics parameters of the model, so these are included as well. For the inverse optimal control both the human and the robot model can be used for the lower level optimization. With the human model the cost functions have a physically meaningful interpretation. We found, however, that the implementation on the robot would lead to different motions because of the kinematic and dynamic differences. Previous researches solved this problem by carefully scaling the weights depending on the particular robot as in [4]. On the other hand, when doing the optimization with the robot model, the found cost functions directly lead to human-like motions when applying them on the corresponding real robot and scaling of the weights in not necessary.

C. Motion recording

To capture reference data for the upper level of the inverse optimal control, we used a dual depth camera setup for recording human motion. A Microsoft Kinect v2 was chosen because of its ability to markerless track humans body positions. Internally it uses a classifier to segment different parts of the body based on depth image data [17]. The SDK

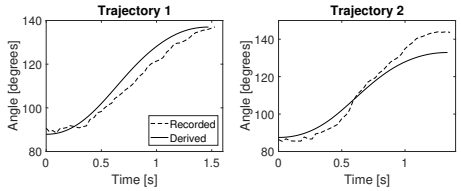


Fig. 3. Arm angle as recorded and derived with the bilevel optimization for two trajectories.

gives access to 25 body positions in three dimensional space, but it was found that the positions and orientation of the arm and hand could not be tracked sufficiently enough. To overcome this a Leap Motion was added to the system, also a depth camera but with a closer range and a much higher accuracy for the lower arm and the hand.

The program for the motion recording was programmed in C++ on Windows. The two cameras are calibrated using a checkerboard pattern using functions provided by the OpenCV library. The body positions of the two cameras are transformed into the same coordinate frame, were the weighting is based on the confidence value provided by the Leap Motion

$$\boldsymbol{p}_{final} = (1 - \text{confidence}) \cdot \boldsymbol{p}_K + \text{confidence} \cdot \boldsymbol{p}_L. \quad (14)$$

This ensures that the hand is tracked even outside of the range of the Leap Motion without making sudden jumps.

The position and orientation of the hand with respect to the body frame are recorded, together with the arm angle and the duration of the motion. These are used to form the reference data v_M for the upper level optimization. The arm angle parametrizes the motion in the nullspace of the arm. It is defined as the angle between a vertical reference plane and a plane though the shoulder, elbow and wrist. Together these seven values uniquely describe the arm configuration [18] and make comparison of motions between two manipulators with small kinematic differences possible.

IV. RESULTS AND VALIDATION

This chapter shows the results of the bilevel optimization, expressed as the optimized weights itself and the contribution of each weighted cost to the total cost. To determine if our method for generating trajectories does indeed leads to improved collaboration we have performed an experiment where we compare it with two existing motion planning strategies. Human participants are asked to rate the shown motions on a number of scales indicating anthropomorphism and feeling of safety.

A. Results of the bilevel optimization

The bilevel optimization was run on four recorded trajectories, two from left to right and two from right to left. The robot model was used for the lower level optimization. The algorithm was able to derive a motion that closely matched the reference, as is shown in Fig. 3 for the arm angle.

TABLE II
OPTIMIZED WEIGHTS FOR THE FOUR TRAJECTORIES

	α_T	α_{energy}	$\alpha_{geodesic}$	$\alpha_{\dddot{q}}$	$\alpha_{\ddot{x}}$	RMSE
I	1	0.0110	1.0304	0	0.7001	7.08
II	1	0.0200	0.9691	0	0.5207	4.62
III	1	0	1.0957	0	0.7222	10.16
IV	1	0.0196	1.2866	0	0.7035	7.52

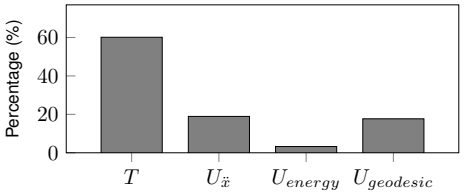


Fig. 4. Contribution of each cost function to the total cost.

Table II shows the result of the optimization of the weights for the four trajectories. Averaging the found weights would not be feasible because they are relative to each other. So instead the set of weights that performs best on all four trajectories was chosen as final result:

$$U = T + 0.020\ U_{energy} + 0.97\ U_{geodesic} + 0.52\ U_{\ddot{x}} \quad (15)$$

Fig. 4 shows the contribution of each weighted costs to the total cost. It is calculated as

$$c_i = \frac{\alpha_i U_i}{U(\boldsymbol{\alpha})} \qquad \text{with} \quad U(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i U_i. \quad (16)$$

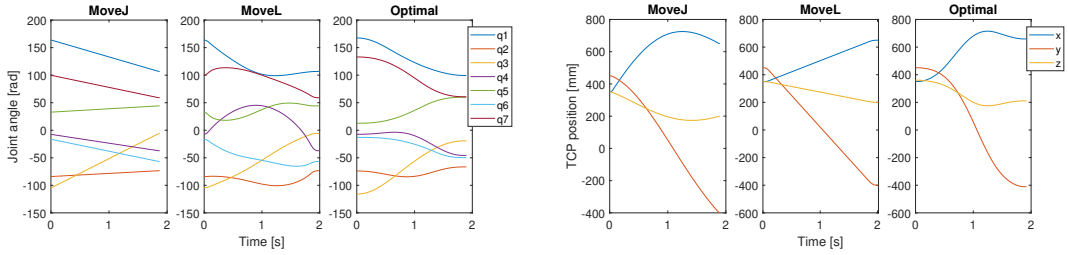
The main difference with the result from [6] is the large influence of T and $U_{\ddot{x}}$. This can be explained by their use of a fixed end time. The geometric and temporal properties of human arm motion are decoupled, according to [19]. T and $U_{\ddot{x}}$ counteract each other, whereas geodesic and energy have little influence on the duration of the motion.

B. Validation of results on the robot

The experiment was preformed using the YuMi robot, which at the moment has two implementations for planning point-to-point motions: MoveL, with paths linear interpolated in Cartesian space and MoveJ, linear interpolated in joint space. These two are compared with our own implementation using the same forward optimal control scheme as was used in the lower level of section II-C, but now with the identified cost function of (15). This motion will be referred to as 'Optimal'.

Fig. 5a shows the joint angle trajectories of the three motion planning implementations. As expected, MoveJ has linear interpolated paths in joint space and MoveL has curved paths. Interestingly Optimal is somewhat in between. In the Cartesian trajectory of the end-effector MoveL has linear interpolated paths, see Fig. 5b. MoveJ and Optimal both show curved paths, with Optimal following a less rounded curve.

The joint angle trajectories are generated offline and exported to motion commands in a RAPID module, which can run on the YuMi robot. The correct joint angular speeds



(a) Joint angles (b) End-effector positions

Fig. 5. Comparison of the three motion implementations used in the experiment.

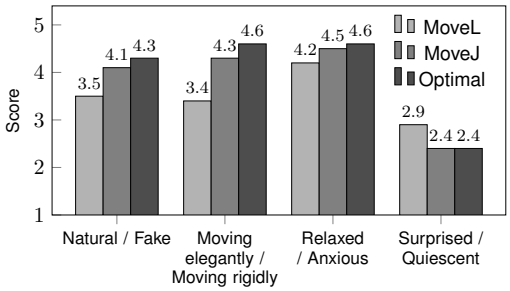


Fig. 6. Average scores given for the three motion planning strategies.

are achieved by specifying the duration of each individual command.

1) Experiment: A questionaire based on [20] is used to determine the reactions of humans on the developed method. It is proposed to use a semantic differential scale to give rating to different criteria regarding anthropomorphism, animacy, likeability, perceived intelligence and perceived safety. We chose to use *Fake - Natural* and *Moving rigidly - Moving elegantly* to determine the anthropomorphism of the motions and *Anxious - Relaxed* and *Quiescent - Surprised* to see how comfortable or safe the participant feels.

A group of 10 participants was tested, with an average age of 26 years. The participants were asked to rate their familiarity with robots on a scale from 1 (low) to 5 (high), on average this was 2.

2) Experimental procedure: The participant was first given a description of the experiment and was asked to sign a consent form. The participant was sitting in front of the robot.

First the three different motions were shown one after the other. Then the motions were shown again, but now with time in between to fill in the four scales. This procedure was repeated four times, with different start and end targets for the robot arm, so in total 12 motions were shown. The order of the motions was mixed for every new target, and the order of the scales was mixed between participants.

TABLE III
ANSWERS TO THE SCALES EXPRESSED IN PERCENTAGES

	1	2	3	4	5	
Fake	12.5	10.0	22.5	30.0	25.0	Natural
Moving rigidly	7.5	20.0	20.0	30.0	22.5	Moving elegantly
Anxious	0	10.0	12.5	30.0	47.5	Relaxed
Quiescent	22.5	15.0	22.5	30.0	10.0	Surprised

(a) MoveL

	1	2	3	4	5	
Fake	0	7.5	10.0	45.0	37.5	Natural
Moving rigidly	0	5.0	15.0	27.5	52.5	Moving elegantly
Anxious	0	2.5	2.5	37.5	57.5	Relaxed
Quiescent	35.0	25.0	12.5	22.5	5.0	Surprised

(b) MoveJ

	1	2	3	4	5	
Fake	0	2.5	2.5	55.0	40.0	Natural
Moving rigidly	0	0	5.0	27.5	67.5	Moving elegantly
Anxious	0	0	5.0	32.5	62.5	Relaxed
Quiescent	32.5	27.5	7.5	30.0	2.5	Surprised

(c) Optimal

V. DISCUSSION

From the average scores displayed in Fig. 6 it is clear that the optimal motions are perceived as more natural and more elegant than the other two implementations. The participants feel less relaxed and more surprised for the MoveL motions, but between the MoveJ and Optimal motions they feel similar. Table III shows the percentage of every choice on the scales. The Optimal motion scores higher for *natural*, *moving elegantly* and *relaxed*.

To measure the consistency of the results we calculate Cronbach's alpha. For the responses to the MoveJ motions it is 0.64 if we include all four scales. If the last scale, *Quiescent - Surprised*, is omitted the alpha-value increases to 0.85, indicating that for internal consistency it is better to exclude this scale. This could be because this scale seems flipped with respect to the other scales, leading to a certain

TABLE IV
CORRELATION BETWEEN RESPONSES

	Fake / Natural	Rigidly / Elegantly	Anxious / Relaxed	Quiescent / Surprised
Fake / Natural	1.0			
Rigidly / Elegantly	0.7421	1.0		
Anxious / Relaxed	0.4488	0.4323	1.0	
Quiescent / Surprised	0.0497	-0.0047	-0.2215	1.0

level of confusion for the participants.

Table IV shows the correlation between the responses to the four scales for all motions. Natural and elegance have a strong positive correlation and a motion perceived as natural and elegant leads to a more relaxed feeling for the participants.

Using an alpha level of 0.05, a dependent-samples t test was conducted. For the *Natural - Fake* scale between MoveL and Optimal there is a significant difference: $t(39) = 4.18, p < 0.05, d = 0.66$. Between MoveJ and Optimal the difference for this scale is not significant. But for *Moving rigidly - Moving elegantly* between MoveJ and Optimal the result is significant $t(39) = 2.48, p < 0.05, d = 0.39$.

A. Anthropomorphism

The first two scales are measuring anthropomorphism, so if we combine them for MoveJ (M: 4.20, SD: 0.89) and Optimal (M: 4.48, SD: 0.64) we can again do the t test: $t(79) = 2.58, p < 0.05, d = 0.29$, which indicates that Optimal is perceived as significantly more anthropomorphic.

B. Perceived safety

When it comes to the last two scales the results can be combined to give a measure for perceived safety. The t test shows the difference is significant between the MoveL and Optimal motions: $t(79) = 3.19, p < 0.05, d = 0.36$. The participants do not experience a significant difference in safety between MoveJ and Optimal, but both are an improvement over MoveL.

It is important to state that perceived safety is only a subjective feeling, while the actual safety of the robot needs to be ensured with other measures.

VI. CONCLUSIONS

We applied inverse optimal control to point-to-point motions of a seven degrees of freedom arm. The bilevel optimization identified a cost function that matched the human recorded motion. The results of this study confirm our hypotheses that the identified cost function can be used to generate motions similar to the human and that it is generalizable to new targets. The experiment with human participants shows the positive effects on how the optimal motions are perceived. Our approach helps towards increasing human acceptance of robot motions and thereby towards the improvement of Human-Robot Interaction.

In the future the approach of this work could be used for an online implementation where the forward optimal control plans its paths in a way that is comfortable for the human. The weightings can also be varied based on the requirements at the moment, for example a higher focus on energy and time minimization in the absence of a human coworker. Another interesting idea is to make the weights dependent on the specific human interacting with the robot, as preferences between humans can greatly differ.

More research is needed on the tasks dependency of the cost functions, as here only point-to-point motions are considered. More complex tasks and situations might require different cost functions.

REFERENCES

[1] A. Sciutti, M. Mara, V. Tagliasco, and G. Sandini, "Humanizing human-robot interaction: On the importance of mutual understanding," *IEEE Technology and Society Magazine*, vol. 37, no. 1, pp. 22–29, 2018.

[2] R. M. Alexander, *Optima for animals*. Princeton University Press, 1996.

[3] K. Mombaur, A. Truong, and J.-P. Laumond, "From human to humanoid locomotion—an inverse optimal control approach," *Autonomous robots*, vol. 28, no. 3, pp. 369–383, 2010.

[4] D. Clever, Y. Hu, and K. Mombaur, "Humanoid gait generation in complex environments based on template models and optimality principles learned from human beings," *The International Journal of Robotics Research*, 2018.

[5] B. Berret, E. Chiovetto, F. Nori, and T. Pozzo, "Evidence for composite cost functions in arm movement planning: An inverse optimal control approach," *PLoS computational biology*, vol. 7, no. 10, e1002183, 2011.

[6] N. Sylla, V. Bonnet, G. Venture, N. Armande, and P. Fraisse, "Human arm optimal motion analysis in industrial screwing task," in *Biomedical Robotics and Biomechatronics (2014 5th IEEE RAS & EMBS International Conference on*, IEEE, 2014, pp. 964–969.

[7] J. Mainprice, R. Hayne, and D. Berenson, "Predicting human reaching motion in collaborative tasks using inverse optimal control and iterative re-planning," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE, 2015, pp. 885–892.

[8] S. Albrecht, K. Ramirez-Amaro, F. Ruiz-Ugalde, D. Weikersdorfer, M. Leibold, M. Ulbrich, and M. Beetz, "Imitating human reaching motions using physically inspired optimization principles," in *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, IEEE, 2011, pp. 602–607.

[9] P. Englert, N. A. Vien, and M. Toussaint, "Inverse KKT: Learning cost functions of manipulation tasks from demonstrations," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1474–1488, 2017.

[10] J. F.-S. Lin, V. Bonnet, A. M. Panchea, N. Ramdani, G. Venture, and D. Kulić, "Human motion segmentation using cost weights recovered from inverse optimal control," in *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*, IEEE, 2016, pp. 1107–1113.

[11] M. J. Powell, "The BOBYQA algorithm for bound constrained optimization without derivatives," *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, pp. 26–46, 2009.

[12] S. G. Johnson. (2008). The NLopt nonlinear-optimization package, [Online]. Available: `http://ab-initio.mit.edu/nlopt`.

[13] H. G. Bock and K.-J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.

[14] T. Flash and N. Hogan, "The coordination of arm movements: An experimentally confirmed mathematical model," *Journal of neuroscience*, vol. 5, no. 7, pp. 1688–1703, 1985.

[15] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms In MATLAB® Second, Completely Revised*. Springer, 2017, vol. 118.

[16] L. J. Soslowsky, E. L. Flatow, L. U. Bigliani, and V. C. Mow, "Articular geometry of the glenohumeral joint.," *Clinical orthopaedics and related research*, no. 285, pp. 181–190, 1992.

[17] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, Ieee, 2011, pp. 1297–1304.

[18] K. Kreutz-Delgado, M. Long, and H. Seraji, "Kinematic analysis of 7 DOF anthropomorphic arms," in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, IEEE, 1990, pp. 824–830.

[19] A. Biess, D. G. Liebermann, and T. Flash, "A computational model for redundant human three-dimensional pointing movements: Integration of independent spatial and temporal motor plans simplifies movement dynamics," *Journal of Neuroscience*, vol. 27, no. 48, pp. 13 045–13 064, 2007.

[20] C. Bartneck, D. Kulić, E. Croft, and S. Zoghbi, "Measurement instruments for the anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety of robots," *International journal of social robotics*, vol. 1, no. 1, pp. 71–81, 2009.

Bibliography

[1] ABB. YuMi® - IRB 14000, 2018. URL `https://new.abb.com/products/robotics/industrial-robots/yumi`.

[2] Sebastian Albrecht, Karinne Ramirez-Amaro, Federico Ruiz-Ugalde, David Weikersdorfer, Marion Leibold, Michael Ulbrich, and Michael Beetz. Imitating human reaching motions using physically inspired optimization principles. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 602–607. IEEE, 2011.

[3] Yogesh J Bagul. A smooth transcendental approximation to |x|. *International J. of Math. Sci. & Engg. Appls.(IJMSEA)*, 11(II):213–217, 2017.

[4] Christoph Bartneck, Dana Kulić, Elizabeth Croft, and Susana Zoghbi. Measurement instruments for the anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety of robots. *International journal of social robotics*, 1(1):71–81, 2009.

[5] 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):e1002183, 2011.

[6] John T Betts. Survey of numerical methods for trajectory optimization. *Journal of guidance, control, and dynamics*, 21(2):193–207, 1998.

[7] Armin Biess, Dario G Liebermann, and Tamar Flash. A computational model for redundant human three-dimensional pointing movements: integration of independent spatial and temporal motor plans simplifies movement dynamics. *Journal of Neuroscience*, 27(48):13045–13064, 2007.

[8] Hans Georg Bock and Karl-Josef Plitt. A multiple shooting algorithm for direct solution of optimal control problems. *IFAC Proceedings Volumes*, 17(2):1603–1608, 1984.

[9] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[10] Debora Clever, Yue Hu, and Katja Mombaur. Humanoid gait generation in complex environments based on template models and optimality principles learned from human beings. *The International Journal of Robotics Research*, 2018.

[11] Peter Corke. *Robotics, Vision and Control: Fundamental Algorithms In MATLAB® Second, Completely Revised*, volume 118. Springer, 2017.

[12] Paolo De Leva. Adjustments to Zatsiorsky-Seluyanov's segment inertia parameters. *Journal of biomechanics*, 29(9):1223–1230, 1996.

[13] Jörn Diedrichsen. Optimal task-dependent changes of bimanual feedback control and adaptation. *Current Biology*, 17(19):1675–1679, 2007.

[14] Moritz Diehl, Hans Georg Bock, Holger Diedam, and P-B Wieber. Fast direct multiple shooting algorithms for optimal robot control. In *Fast motions in biomechanics and robotics*, pages 65–93. Springer, 2006.

[15] Peter Englert, Ngo Anh Vien, and Marc Toussaint. Inverse KKT: Learning cost functions of manipulation tasks from demonstrations. *The International Journal of Robotics Research*, 36(13-14):1474–1488, 2017.

[16] Tamar Flash and Neville Hogan. The coordination of arm movements: an experimentally confirmed mathematical model. *Journal of neuroscience*, 5(7):1688–1703, 1985.

[17] Kinect for Windows Team. Microsoft to consolidate the Kinect for Windows experience around a single sensor, 2015. URL https://blogs.msdn.microsoft.com/kinectforwindows/2015/04/02/microsoft-to-consolidate-the-kinect-for-windows-experience-around-a-single-sensor/.

[18] John M Hollerbach. Optimum kinematic design for a seven degree of freedom manipulator. In *Robotics research: The second international symposium*, pages 215–222. Cambridge, MIT Press, 1985.

[19] Leap Motion Inc. Leap Motion, 2018. URL https://www.leapmotion.com/.

[20] K Kreutz-Delgado, M Long, and H Seraji. Kinematic analysis of 7 DOF anthropomorphic arms. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 824–830. IEEE, 1990.

[21] Dana Kulić and Elizabeth Croft. Physiological and subjective responses to articulated robot motion. *Robotica*, 25(1):13–27, 2007.

[22] Alain Liegeois. Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE transactions on systems, man, and cybernetics*, 7(12):868–871, 1977.

[23] Jonathan Feng-Shun Lin, Vincent Bonnet, Adina M Panchea, Nacim Ramdani, Gentiane Venture, and Dana Kulić. Human motion segmentation using cost weights recovered from inverse optimal control. In *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*, pages 1107–1113. IEEE, 2016.

[24] Anthony A Maciejewski and Charles A Klein. Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. *The international journal of robotics research*, 4(3):109–117, 1985.

[25] Jim Mainprice, Rafi Hayne, and Dmitry Berenson. Predicting human reaching motion in collaborative tasks using inverse optimal control and iterative re-planning. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 885–892. IEEE, 2015.

[26] David Q Mayne, James B Rawlings, Christopher V Rao, and Pierre OM Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[27] Katja Mombaur, Anh Truong, and Jean-Paul Laumond. From human to humanoid locomotion—an inverse optimal control approach. *Autonomous robots*, 28(3):369–383, 2010.

[28] H Nagasaki. Asymmetric velocity and acceleration profiles of human arm movements. *Experimental brain research*, 74(2):319–326, 1989.

[29] Yoshihiko Nakamura and Hideo Hanafusa. Optimal redundancy control of robot manipulators. *The International Journal of Robotics Research*, 6(1):32–42, 1987.

[30] Roger Penrose. A generalized inverse for matrices. In *Mathematical proceedings of the Cambridge philosophical society*, volume 51, pages 406–413. Cambridge University Press, 1955.

[31] Davide Piovesan, Alberto Pierobon, Paul DiZio, and James R Lackner. Experimental measure of arm stiffness during single reaching movements with a time-frequency analysis. *Journal of neurophysiology*, 110(10):2484–2496, 2013.

[32] Michael JD Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in optimization and numerical analysis*, pages 51–67. Springer, 1994.

[33] Michael JD Powell. The BOBYQA algorithm for bound constrained optimization without derivatives. *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, pages 26–46, 2009.

[34] F Schlagenhauf, Pratyusha Sahoo, and W Singhose. A comparison of dual-Kinect and Vicon tracking of human motion for use in robotic motion programming. *Robotics & Automation Engineering*, 2017.

[35] Alessandra Sciutti, Martina Mara, Vincenzo Tagliasco, and Giulio Sandini. Humanizing human-robot interaction: On the importance of mutual understanding. *IEEE Technology and Society Magazine*, 37(1): 22–29, 2018.

[36] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1297–1304. Ieee, 2011.

[37] Louis J Soslowsky, Evan L Flatow, Louis U Bigliani, and Van C Mow. Articular geometry of the glenohumeral joint. *Clinical orthopaedics and related research*, (285):181–190, 1992.

[38] Nahema Sylla, Vincent Bonnet, Gentiane Venture, Nasser Armande, and Philippe Fraisse. Human arm optimal motion analysis in industrial screwing task. In *Biomedical Robotics and Biomechatronics (2014 5th IEEE RAS & EMBS International Conference on*, pages 964–969. IEEE, 2014.

[39] Michael W Walker and David E Orin. Efficient dynamic computer simulation of robotic mechanisms. *Journal of Dynamic Systems, Measurement, and Control*, 104(3):205–211, 1982.

[40] Y. Wang and P. Artemiadis. Closed-form inverse kinematic solution for anthropomorphic motion in redundant robot arms. *Adv Robot Autom*, 2(110):2, 2013.

[41] Xu Xu and Raymond W McGorry. The validity of the first and second generation Microsoft Kinect™ for identifying joint center locations during static postures. *Applied ergonomics*, 49:47–54, 2015.

[42] Tsuneo Yoshikawa. Analysis and control of robot manipulators with redundancy. In *Robotics research: the first international symposium*, pages 735–747. MIT press Cambridge, MA, USA, 1984.

[43] Andrea Zanchettin. *Human-centric behaviour of redundant manipulators under kinematic control*. PhD thesis, Italy, 2012.