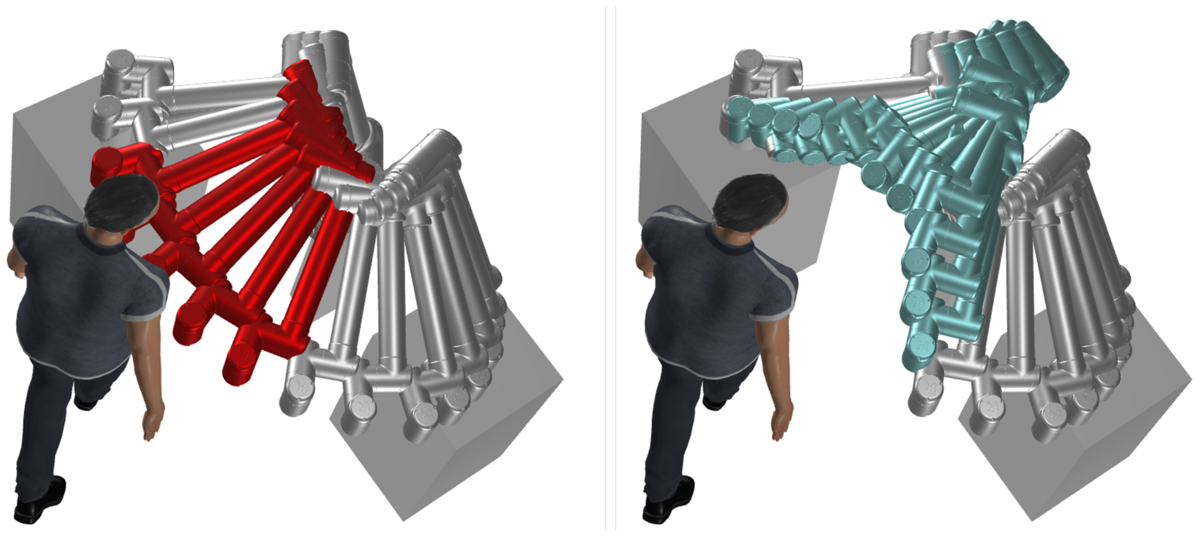
CHALMERS
UNIVERSITY OF TECHNOLOGY



Real-time Tracking of Human Motions and Adaptive Robot Path Planning for Assembly Cooperation

Master's thesis in Systems, Control and Mechatronics

Hanna Berggren
Fabian Melvås

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Master's thesis 2018:11

Real-time Tracking of Human Motions and Adaptive Robot Path Planning for Assembly Cooperation

Hanna Berggren
Fabian Melvås

Department of Electrical Engineering
Systems and Control
Chalmers University of Technology
Gothenburg, Sweden 2018

Real-time Tracking of Human Motions and Adaptive Robot Path Planning for Assembly Cooperation
HANNA BERGGREN
FABIAN MELVÅS

Supervisor: Yi Li, Fraunhofer-Chalmers Centre
Examiner: Knut Åkesson, Electrical Engineering

Master's Thesis 2018:11
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: The robot path was blocked by the human worker so a new path is generated that consider the human worker.

Typeset in LaTeX
Gothenburg, Sweden 2018

Real-time Tracking of Human Motions and Adaptive Robot Path Planning for Assembly Cooperation
HANNA BERGGREN
FABIAN MELVÅS
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Today, industries are working towards Industry 4.0 and smart factories with robots controlled by artificial intelligence. Smart factories are especially applicable to industries where a high product variation is common. Furthermore, some robots in smart factories have to collaborate with human workers. A digital twin of the manufacturing process can then be created and used to check whether the robots can indeed operate safely in the presence of the human workers.
The aim of this thesis to enable human-robot collaboration in smart factories by developing a decision framework on top of some existing software packages. Firstly, we track the movement of the human worker with IMU (Inertial Measurement Unit) sensors attached to the worker. Secondly, we replicate the movement in a virtual environment by updating both pose and position of the worker's digital twin, using the measured data. Thirdly, our decision framework continuously predicts the next movements of both the human worker and the robot. Whenever the human worker violates the minimum distance to the robot (i.e., risk colliding with the robot), an existing software package uses the digital twins of both the worker and the robot to calculate a collision-free path for the robot. Finally, the robot receives the new collision-free path and executes it to avoid collision with the human worker. Consequently, we can guarantee human-robot collaboration without worrying about the robot causing injuries.

Keywords: Human-robot interaction, Human robot collaboration,

Acknowledgements

We would like to thank to our supervisors Yi Li (FCC) and Knut Åksesson (Chalmers) for guidance and help during the project and writing process of the report.
This thesis would not have been possible without the help from Fraunhofer-Chalmers Centre (FCC) and its employees who have helped us with implementations during our thesis work. We would also like to give a special thanks to Johan Carlsson for encouraging us and providing resources to our project. The results would not have been possible without implementations made by Niclas Delfs who created a connection between the Xsens system and IPS, Christian Larsen who created a connection between the UR-sim and IPS. We would also like to thank Jonas Kressin for helping us with IPS when nothing worked and we had no idea why.
During the project, the Xsens-system was borrowed from the department of Product Design Engineering at University of Skövde for which we are very grateful.
Finally, we would like to thank the hub-people: Christian Larsen, Joakim Thorén, and Matteo Canavero for daily help with everything from controlling robots to understanding the concept of compilers.

Hanna Berggren & Fabian Melvås, Gothenburg, Nov 2018

Contents

List of Figures

List of Tables

1 Introduction

Today, we are entering Industry 4.0 [29] in which artificial intelligence controls the robots with very little input from human operators. Industry 4.0 introduces what is commonly referred to as the “smart factory” (i.e., “a manufacturing solution that provides such flexible and adaptive production processes that will solve problems arising on a production facility with dynamic and rapidly changing boundary conditions in a world of increasing complexity” [42]). For decision makings within a smart factory, it is essential that digital twins [40] (i.e., digital replicas of physical assets, processes and systems) are deployed.

Currently, the cages surrounding industrial robots are often used to separate them from human workers. Although some industrial robots are deemed safe enough to be operated outside the cages and in the presence of human workers, they are often programmed to decrease speed as a human worker approaches and stop if the worker is in the danger of colliding.

Furthermore, nonrepetitive tasks in an assembly line are often carried out by hand. In a smart factory, this work can be executed with robots controlled by artificial intelligence either without or in collaboration with human workers. During collaborative work, it is vital to avoid collision between the robot and the worker. This can be achived by mapping the movements of the worker and the robot to their digital twins, who are used to detect and avoid future collisions by changing the robot path.

1.1 Overview of Approach

First, we established a high-level planning system consisting of three main parts: human tracking, control and path planning of the robot, and a decision algorithm that includes a prediction part. A digital real time representation of the working station, including a human worker and a robotic arm, is created in IPS-scene to enable path planning for the robot.

The human is mapped to the digital environment using inertial measurement units (IMUs) from Xsens and the Xsens MVN software. We created two manikins (i.e., digital twins of the human): one MVN manikin and one IPS manikin, with two different skeleton models. These two manikins are connected to each other through a program implemented at FCC. Using recorded motion sequences, different mappings between the skeletons were evaluated. The precision of the tracking performed of the Xsens system was tested to determine whether a complementing tracking system is required.

The robot path planning tools in IPS were evaluated to find appropriate path planning tools that are both fast and accessible via the communication language LUA. Furthermore, the decision program communicates between IPS and URSim, which is a program used to simulate a Universal Robot. We implemented an algorithm that can retrieve the path from IPS, convert it to robot code, and send it to URSim. Since the path planner in IPS does not consider changes in velocity, acceleration, or jerk, the constant velocity approximation was evaluated to determine whether a more advanced motion model is needed. Moreover, the path planner was tested in different scenarios to determine the average time it takes to generate a new path.

The time critical decision system is supported by predicting movements of the robot and the human motions. The prediction of the robot was developed in close collaboration with the path planning since information about the future robot path is required. When choosing the method to visualise the robot, our main priorities were speed and flexibility.

The human prediction requires information regarding the manikin skeleton model. Hence, the model was investigated to find the necessary joints to enable a good estimation of the future human pose, and how that pose can be sent to another IPS scene efficiently. Moreover, a Kalman filter was used to estimate the velocity from the humans pose and position. Then a constant velocity model was used to calculate a prediction of the selected joints. Finally, the predicted pose was used to detect violations of the minimum distance between the human and the robot, by executing an LUA script that moves the manikin to the predicted pose and let a simulated robot step through some seconds of the future robot path.

Whenever the prediction algorithm detects a violation of the minimum distance to the human, a new path is calculated. Moreover, the decision algorithm communicates with the prediction, path planning, and robot simulator URSim.

1.2 Contributions

For human-robot collaboration (e.g., when a robotic arm operates close to humans), the safety of the human workers is paramount. Hence our work aims to develop a system that avoids collisions by controlling the robotic arm and tracking the human movements. We use digital twins of both the robotic arm and the worker to compute the distance between them and calculate new paths for the robotic arm whenever necessary.

Often system tracks the movements of the human worker with IMUs. We used the Awinda IMU-system from Xsens because it is a non-optical tracking system. Hence no line of sight between the Xsens receiver and the sensors placed on the body of the worker is required. The Xsens sensors are capable of capturing full 6 degrees of freedom (DOF) motion of the body segments. However, these sensors suffer from drift. For example, the digital twin of the worker may move slowly in the virtual environment even though the worker was standing still the whole time. In this thesis, a simulated robot is used hence the human can compensate for the drift by moving in the real world.

Furthermore, we implemented an algorithm that decides if and how to calculate a new robot trajectory. The decision is based on not only the current distance between the worker and the robot but also distances between a predicted state of the worker and the robot's future states along its path. Our prediction algorithm approximates the future human pose uses a constant velocity model. Since only the position of the worker is known (i.e., his/her velocity is unknown), the worker's position and arm joint angles are filtered through a Kalman filter to approximate velocity values for the prediction algorithm.

Here are our main contributions:

- We developed an algorithm that uses existing online trajectory planning for robots to quickly generat a new trajectory.
- We built a framework to enable real-time collaboration between human and robot.

1.3 Related Work

In a general review of human-machine cooperation for assembly operations from 2009 [25], the authors concluded that a safe workspace is essential. To ensure this, continued research regarding sensors and a cooperation method where multiple humans are supported by one or more systems are required.

Among the reviewed articles in [25], a common decision strategy is to stop the robot if a minimum distance between the robot and the human is violated, such as the distance measured by a tracking-vision-chip in [13].

Safety strategies for human-robot collaboration can be divided into pre-collision or post-collision [19]. Post-collision safety strategy aims to minimise the injury the robot causes if it collides with a human. However, it is better to avoid the collision. Pre-collision safety strategy aims to detect obstacles using sensors and stop the robot. Additionally, it is mainly pre-collision safety systems that are explored in the context of decision systems and path planning for collaborative robots.

Requirements and architecture for human-robot collaboration in production systems are discussed in [60]. The requirements concern the human interface and aspects about the robot behaviour. The human should be able to communicate with the system using a trivial interface, and the system should have access to the information which could be used for task planning. The proposed closed-loop architecture includes collision detection, task planning, control of the robot, and “Worker Assistance modules”. The proposed system for collision detection includes a 3D digital twin model updated by sensor information. The planning of the tasks and the robot dynamics is done in a 3D environment and converted by IEC-61499-based function blocks to robot code. The aim is to create user-friendly and ergonomic interaction to the “Worker Assistance modules”.

A similar approach that focuses on task planning is used in [41]. The human motions are, while performing tasks, visualized as sweeping volumes with uncertainties which are used when choosing among the robot trajectories. The approximated execution time for each trajectory together with the uncertainty of the duration and the risk of collision, is used in the task planning to find a feasible solution that minimises the cycle time. The model of the human’s tasks is probabilistic and integrated with the motion planning for the robot. The order of the robot tasks and trajectory are calculated through an iterative process which results in a more flexible scheduling.

In [59], an active collision avoidance is described, where the system visualises the collaborative scene as a virtual 3D model, the robot is tracked by information from the robot controller, while two Kinect cameras track the human. The distance between the human and the robot is measured, and then used to choose between four robot control strategies. These strategies are:

1. If a human enters the area near the robot, the human is alerted with a sound and the robot speed is limited.
2. If the human enters within the next safety area, denoted by the hazard zone, the robot is stopped.
3. If the human has entered the hazard zone but keeps approaching the robot, it starts to move away from the human.
4. If the human is within the hazard zone but the robot trajectory can be adjusted to enable the robot to continue its work, the new trajectory is sent to the robot.

By using these strategies, the robot is adapting to the human movements. Once the human has moved out from the hazard zone, the robot continues from where it was interrupted. The developed method for control is integrated with the Wise-ShopFloor framework [58] that is used for real-time monitoring of manufacturing.

A similar method for active collision avoidance is described in [10]. To track the human, an inertial motion capture suit is used together with an Ultra-WideBand localisation system. The tracking data is used to create a digital twin by moving a virtual human, represented as a manikin made out of a hierarchy of bounding volumes. A similar model is used for the robot that is moved by following the joint values from the robot controller. Finally, the robot is stopped if a violation of the minimum distance between the manikin and robot occurs in the virtual environment.

Another approach is to use both vision and physical contact with the robot to make decisions and for the control of the robot. This approach is applied in [6] to a position controlled robot and an admittance controller in an assembly environment. The robot has an initial path and can switch between active and passive behaviour. To activate the switch, the system either uses camera information or detects a force applied to the robot. The camera information can both stop the robot and trigger different operations. The contact forces are used to start or end an assembly and for safety function such as changing the robot trajectory to avoid a collision or stop the robot if the applied force is too high. This framework reduces load for the operator and makes assembly safer.

Robot behaviour can also be changed by using methods from [24], where the robot reacts instantaneously to unforeseen events using an online trajectory-generator. The control system switches between "trajectory-following motions, sensor-guided motions, and sensor-guarded motions within one framework".

1.4 Ethical Aspects

1.4.1 Sensors

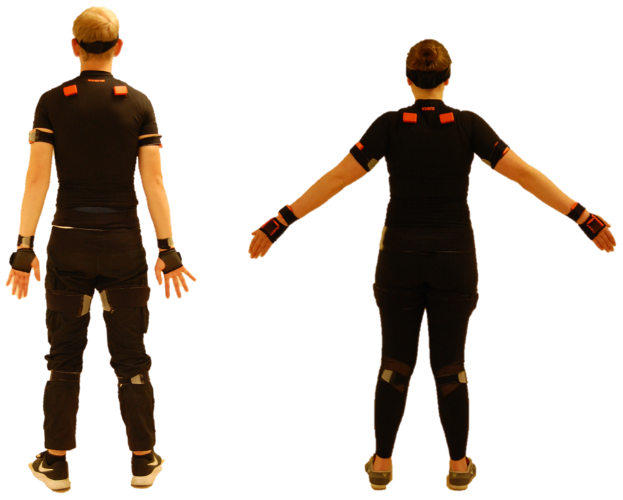


Figure 1.1: A set of motion tracking sensors attached to the bodies of two human workers.

The usage of new technology brings a lot of opportunities, but it also comes with ethical dilemmas. The ethical considerations should be reflected when using robots and sensors.

The sensors can be use in an industrial environment to prevent repetitive strain injuries, for example, the ergonomics in the work environment is essential. By analysing IMU sensor data from daily work with an ergonomic model, one can find improvements that for example reduce the risk of strain injuries. Or it can be detected that the cycle time needs to be increased to enable the worker to perform a task ergonomically.

The sensors could also be used to track motions and monitor how a human worker is moving in the workstation and thereby make sure that all the tasks are performed so that the number of production errors is reduced.

Unfortunately the workers can regard the usage of sensors as intrusive and stressful, since the sensors data also can be used for monitoring and control. Consequently, the purpose of the sensors must be clear for the involved workers. There are also some practical problems with the sensors if used daily, for example, that they must be easy to attach, ware, and calibrate. A solution could be to integrate the sensors to the worker's clothes but they still must be calibrated frequently.

1.4.2 Human and Robot Collaboration

The robots are mainly used to carry out repetitive tasks and work in environments dangerous for humans. In a human-robot collaboration senario, the robots can be used by the human worker as a flexible tool to help with carrying heavy items or execution of repetitive tasks, while the worker focuses on complex tasks the robot cannot solve. In a production line that produces multiple models of a specific product due to customer demands, a collaborative robot is ideal since it can adjust and change paths and tasks depending on the current product.

The robots are not only used in production but also entering our homes since they are starting to adapt to changing environments. Here the robot can be a flexible companion, helping with daily care and support a human as he or she moves in and out of bed, takes a shower, or prepares a meal. By using robots in the care of elderly, the human staff can focus on other things such as specific needs and wishes, for example, help in discussion of treatment or with making planes for visiting relatives and friends.

1.5 Outline

An overview and a general description of the problems solved in this thesis work are explained in detail in Chapter 2.

The theory for human tracking and creation of virtual human representations are explained in Section 3.1. This section also contains a description of IMU sensors and how the humans are represented as manikins when using Xsens MVN Analyze and IPS.

The UR robot used in this thesis, its control system, and theory about generating new paths are explained in Section 3.2.

In time critical real time systems like this, predictions of the human and the robot can be performed to foresee violations of the minimum distance between human and robot. The prediction algorithm is explained in Section 3.3.

Concept and implementation of a collaborative robot system are described in Section 4. More specifically, the framework behind the program architecture and the decision process are described, such as how the decision process controls the calculation of new paths and the robot.

The collaborative robot system is evaluated in Chapter 5. A general discussion is made in Chapter 6 whereas conclusions and future work are found in Chapter 7.

2 Problem Description

The aim of this thesis is to enable human and robot collaboration in an assembly environment, where the robot follows a path between specified positions and adjusts the path whenever blocked by the human worker. To adjust the robot path the human and robot intentions are essential, hence they should be represented in a virtual environment so that their digital twins can be used to recompute the robot path when necessary.

Consequently, a framework that communicates with existing software and systems is required to enable adaptive path planning for the robot. Three main software blocks are used to enable adaptive path planning: the Xsens system, IPS, and URSim. The Xsens system consists of a sensor suite and the software Xsens MVN Analyse that together is used for tracking of human motions. For collision detection between the human and robot and path planning the framework use IPS. This program has previously been used to represent reality in a virtual environment, referred to as a digital twin. Moreover, IPS can be used to visualize industrial cells, generate efficient robot paths in narrow environments, measure distances, detect collisions, and simulate human movements based on a kinematic model [50, 39]. URSim is a program for simulating the real robot behaviour, where instructions are handled in the same way as the real robot and the robot motion can be monitored and streamed

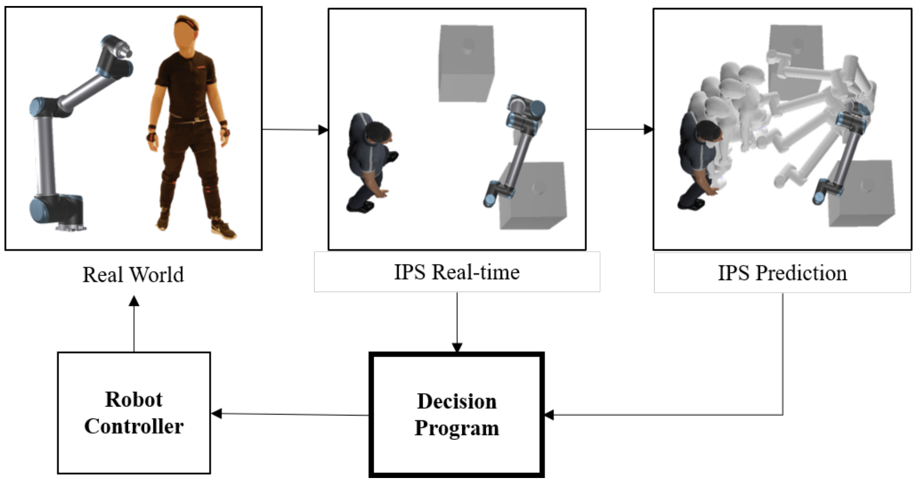


Figure 2.1: Overview of the main framework.

to other software blocks.

The communications between the programs are minimised to speed up the framework since the system has to be fast to handle real time situations. The communications can be described as shown in Fig. 2.1. The pose of the human is tracked and visualized in IPS, together with the robot positioned in the same joint configurations as the simulated robot. Next, an algorithm uses this real time information to generate new paths when a collision is likely to occur. To detect collisions, a prediction algorithm based on the properties of the manikin and the current robot path is used. It also compensates for the delay in the framework.

The decision program retrieves the distance between the robot and the manikin both from the real time visual representation and the predicted position of the human worker. This information is used with the current robot configuration to decide if and how to change the robot path. The decisions have to handle different situations and avoid deadlocks. Furthermore, it is essential to use the path planning algorithm in IPS efficiently since it is time critical to find a new collision-free path. Therefore it is desirable to only re-plan the necessary parts of the path and find a sufficient solution fast.

3

Theory

This chapter covers both the basic theory regarding human tracking and representation as well as the robot representation and control.

3.1 Human Tracking and Representation

To enable human and robot collaboration, the system controlling the robot needs to know the position and pose of the human. To that end, the human is represented in the same digital environment as the robot, and the human is tracked using an IMU tracking suit from Xsens. The recorded data is filtered and linked to the manikin in the Xsens software MVN Analyze. Then the positions of certain manikin body parts are mapped to another manikin represented in the IPS module Intelligently Moving Manikin (IMMA). Since the robot is also mapped into IPS, the digital twins of both the human and the robot are represented in the IPS environment.

The human representation in IMMA uses a mathematical model that can synthesize collision-free and ergonomic motions. The base of the model is the human skeleton which consists of bones connected with joints. Due to the differences between individuals regarding mobility in joints and the length of the bones, a manikin is created for each human worker using his/her measurements of different body parts in order to enable a good representation.

3.1.1 Track Human Movements and Human Representations

To track human motions, two different sensor technologies can be used: the image based and the non-image based methods. For image-based methods, markers or colours are commonly used to visually identify the position of the humans and objects tracked by cameras in the environment. Unfortunately, single-camera approaches have limited view angle, depth recognition, and it can occur problems because of occlusion [5]. However, single-camera implementations are still used, for example, methods in [23, 33] recognise high-speed hand gestures with the use of a camera and specially designed processors for visual computing. Another option is stereo camera vision, where two cameras can be combined to create a 3D view. However, multiple cameras increase the complexity and need additional calibration [33]. A stereo camera implementation presented in [26] is used to improve human detection by combining three cameras that support colour recognition in a triangulation. Multiple cameras can also compensate for the occlusion.

Another image based tracking technology is depth sensors. This technology is especially useful when the light is low or unreliable, or when multiple objects have the same colour in the environment. A method within the area of depth sensors is to measure the time for the light to travel and be reflected. It is time efficient but limited by the light power and reflection of surfaces [33]. A review in [51] present methods for hand gesture recognition with depth sensors.

Continuing to non-image based approaches, which include accelerometers and gyroscopes that are difficult to calibrate and setup. A setup based on accelerometers and gyroscopes are therefore not a trivial chose for tracking in manufacturing environments [33]. They are, however, used by [9] to follow human motions and map to a human representation of bounding volumes in a 3D environment. Another wearable non-image method is to use electromyogram sensors. It is for example used in [46] for gesture recognition. They use sensors that are band-based and measure electrical activity produced by skeletal muscles together with micro-electro-mechanical systems (MEMS) to track the human pose.

WiTrack and RF-Capture system [22] track human motions using radio frequency (RF) signals reflected from the human. The RF signals can reach through walls and are therefore not limited to track the human in areas visual for the sensor. Moreover, these tracking systems can identify different limbs and distinguish human workers from each other.

Since the sensors have different limitations, one may benefit from combining them. A work that uses different kinds of sensors is presented in [61], resulting in a “multi-source heterogeneous vision perception framework”. This tracking system consisting of RGB-D cameras, binocular cameras, two monochromatic IR (infrared radiation) cameras, and three infrared LEDs (Light Emitting Diode). The RGB-D cameras produce a 3D point cloud by using both colour detection and depth information, while the binocular cameras are used to track hands and fingers. The framework is computationally expensive, but by using the sensors together they can compensate for the sensors’ limitations and produce a good result according to the authors.

To enable the creation of a virtual representation from sensor measurements the information must be extracted, for example, by different learning algorithms sush as Support Vector Machine (SVM), Artificial Neural Networks (ANN), and Random Decision Forests (RDF). Moreover, these methods can identify patterns in the raw data [33], unlike visual feature methods. A method described in [56] use a Deep Neural Networks (DNNs), where the regression problem is formulated to identify human poses from joint coordinates. Furthermore, the used generic convolutional neural network (CNN) is mainly developed to fit classification problems but has also proven to be suitable for human pose recognition. In [55], the information is extracted by a randomised decision forest classifier (RDF) operating on depth images. This system combines a labelled dataset generation method, a convolutional neural network, and inverse kinematics to determine the human pose.

The representation of humans in the virtual world can be done with multiple methods. Furthermore, human motions are often described by simplified models to decrease the calculation time. The representations can be a point clouds as done in [61]. Another method is to use bounding volumes linked by a kinematic model where each bound volume represents a human body part. Unfortunately, uncertainties are created by using the simplified models and by tracking motions with sensors. One of many methods to represent the uncertainties is to change the size of the bounding volumes proportionally to the velocity of the limb and dependent on the sampling time [9].

3.1.1.1 Inertia Measurement Unit

Cameras have been used in a previous work at FCC to capture the movements of a human. In that work, occlusions were the main problem [5]. Instead, Inertia Measurement Unit's (IMU's) can be used to track linear and angular motions by combining data from the different sensors within them. Each IMU has a triad of gyroscopes, a triad of accelerometers, and a 3D magnetometer. The IMUs used are developed by Xsens.

A basic gyroscope uses a rotating mass (rotor) attached to two gimbals mounted in a frame as shown in Fig. 3.1. The construction makes it possible for the rotor to change orientation in space independent of the frame. When the rotor is spinning, it can rotate the frame while the rotor keeps its orientation due to inertia. Another gyroscope construction often used in electronics consists of a mass attached to the sensor body by springs. By rotating the sensor, the Coriolis force, that affects the mass, creates a compression of the springs in correlates to the angular velocity. The angular velocity can be further integrated to retrieve the current angle of the sensor and enable tracking of orientation changes of the sensor frame. Unfortunately, the integration of measured data is associated with drift problems. To mitigate, the data from the gyroscope is used with the data from both the accelerometer and magnetometer.

Similar to the gyroscope, the accelerometer has also a built-in weight that can move relatively to the body of the sensor as shown in Fig. 3.2. The inertia of the weight creates a distance shift between the weight and the body of the sensor when the sensor is accelerating. This change of distance is used to calculate the acceleration by measuring the spring compression as electrical differences. When the sensor moves with constant velocity or is in freefall, it is only affected by one acceleration (i.e., the gravity $9.82m/s^2$). By integrating the acceleration, the velocity and the position can be calculated. Similar to the gyroscope, the accelerometer has also

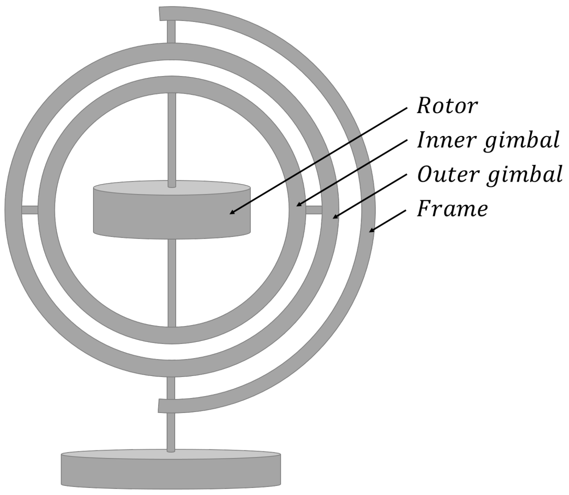


Figure 3.1: A gyroscope consisting of a rotor attached to gimbals in a frame.

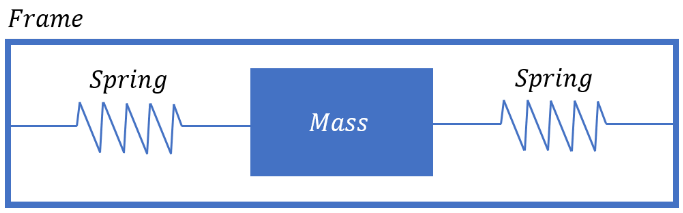


Figure 3.2: An accelerometer with a weight that can move within the sensor.

drifting problems.

There are multiple types of magnetometers. One concept is to apply an electrical current through a conductive material and detect magnetic fields by measure the current change. For applications that use IMUs to find the position and orientation, the magnetometer is used to measure the magnetic field of the earth since that information can be used to orientate around the axis parallel to gravity [45]. Hence, surrounding metallic and electronic objects can disturb the sensor.

The measurements from the different sensors are filtered to compensate for problems related to the different measurement techniques. Research from 2009 regarding orientation estimation with IMU's indicates "Orientation estimation was poor with the quaternion filter, for the Kalman filter results were acceptable, despite a systematic deterioration over time (after 20–30 s)" [11].

Filters such as the Kalman filter use a measurement model that approximates the reliability of the sensor and a mathematical model that describes the motion of the measured object. By processing real time and previously measured data with these models, the state estimation can be improved. Because of the usage of older measurement data the filter has to collect some measurements before it can provide reliable output. Hence, it neglects a short period of magnetic disturbance. Unfortunately, magnetic disturbance remains for a while after the sensor has left the magnetic field if it has been there long enough [11]. The measurements should, therefore, start in a “safe” area and the sensors cannot stay in a heavily distorted earth magnetic field longer than 20–30 seconds without affecting the filter.

Moreover, the magnetometer data can be used by a Gaussian Process Simultaneous Localisation and Mapping framework to improve the position estimation [7]. However, this system cannot yet handle changing magnetic environment such as a production assembly line where multiple electrical objects are present and moving.

3.1.2 IPS Manikin

The usage of the IPS-module IMMA is mainly to create assembly motions for manikins and the assembly parts that are collision-free and ergonomic. For example, the manikin model can be used to measure how ergonomically correct different body positions are and thereby find critical poses to analyse further. It is also used to simulate and analyse the operations in an assembly line.

Each IMMA is created by defining the length of different body segments or by choosing this from a database over measurements from different populations. For example, one can analyse the effect of different anthropometric on ergonomic motions by creating multiple manikins with different anatomy from the database. The manikins biomechanical model is built up in a hierarchical tree structure with 82 segments and 162 joints.

To control the manikin, one can use control points along with the corresponding constraints such as a position and/or an orientation.

Then the manikin is moved to the pose with the lowest comfort loss given the constraints. The optimisation formulation used to calculate the comfort costs considers both joint angles and joint torques. However, the forces such as contact forces and gravity are also taken into consideration, for example, to balance the manikin [3, 18]. Whenever a solution is unfeasible due to multiple constraints, the manikin is moved to the pose that minimises the mean-square error.

In IMMA, a manikin is represented not only as a biomechanical model but also as a collision model and triangular mesh (see Fig 3.3). The collision model is built by rectangular swept spheres (RSS) structured in hierarchies to enable fast distance calculations. Hence, it is used for to compute distances when the scene changes. The triangular mesh representation, with 40000 triangles, is mostly used for visualization, but also for distance calculation during path planning.

3.1.3 Xsens Manikin and Software

For tracking of human motions, the Xsens motion capture system MVN Awinda is used. It consists of 17 wireless motion trackers placed at the pelvis, stern, head,

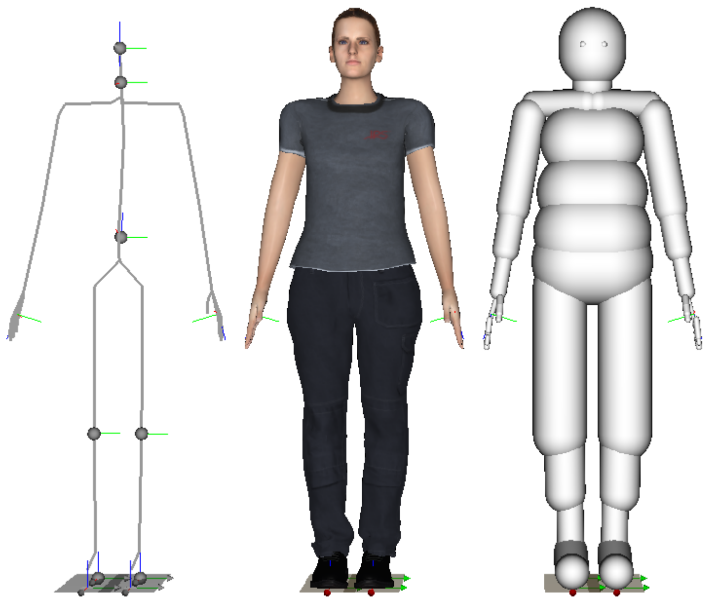


Figure 3.3: Different visualisations of the IPS manikin, from the left: stick figure with control points marked, triangular mesh and collision model.

shoulder blades, hands, feet and upper and lower arms and legs [53] as shown in Fig 3.4. The Xsens MVN software filters the sensors data and uses a biomechanical human motion model to move a manikin according to the input. The biomechanical model consists of 23 segments connected by 22 joints and can help with correction of drifting measurement data since it also correlates the position of the sensors to each other. To create a biomechanical model, the user has to provide measurements of 10 specified body parts. The sensor data are used to identify position and orientation of each joint which together with the length of all segments can be used to create a 3D visualisation of the motion [45].

By recording the resulting manikin movements in Xsens MVN Analysis, the real time human motions can be saved for later usage. The program can also stream either saved or real time movements over a network to another computer as well as export the saved motions. Hence, other software such as IPS can use the data.

3.1.4 Summary

We use IMUs to track a human's linear and angular motions of different body parts. The collected information from the different sensors in the IMU is filtered to mitigate the drift problems associated with each sensor type.

The software Xsens MVN Analyse can be used to track and represent human motions digitally. To enable that, data from a set of IMU sensors fixed to a human are filtered and linked together by a biomechanical model. By using the result in the IPS-module

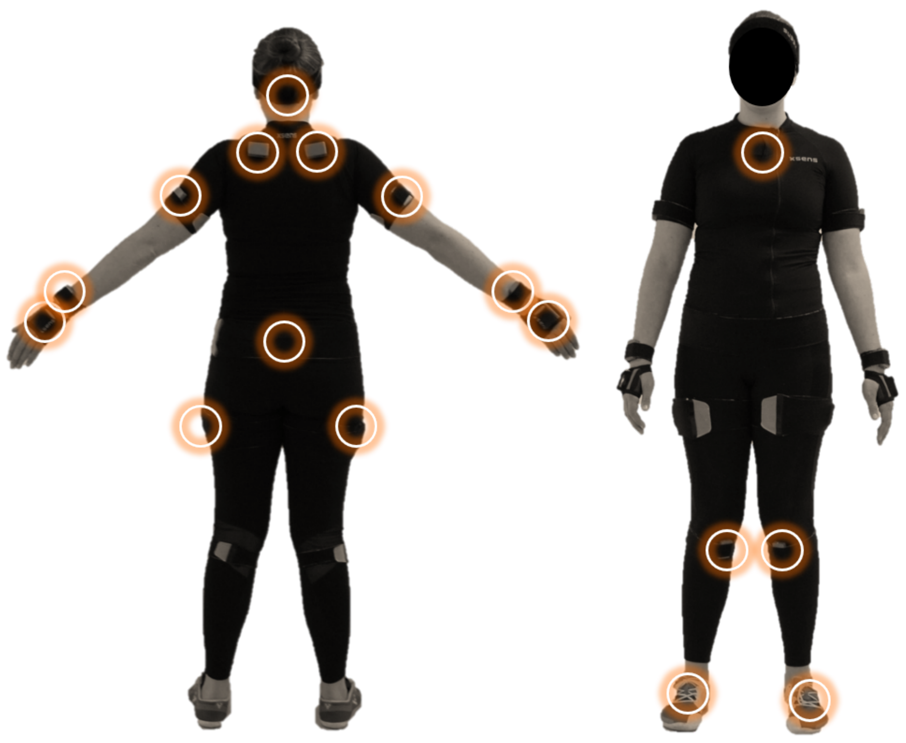


Figure 3.4: A human wearing the Awinda tracking suit with the 17 sensors marked.

IMMA, the ergonomically best manikin pose can be calculated. Furthermore, IPS can not only compute distances between the manikin and the robot but also plan path for the robot.

3.2 Robot Path Planning, Control and Simulation

By using our framework for human tracking, one can represent a human worker in an IPS-scene along with a robot. The IPS software supports kinematics and path planning functions for several robot types. One of them is the collaborative UR10 robot, produced by Universal Robots (UR). In industries, the UR robots are used for different tasks such as packaging, plastic, and polymer production, testing, screw driving, and polishing. During our evaluation tests a simulator called URSim is used instead of a real robot and the instructions are sent by a ZeroMQ connection.

3.2.1 Universal Robotics UR10

To evaluate the developed framework and decision algorithm, it is tested using a Universal Robot UR10 as shown in Fig. 3.5. With a maximum payload of 10-kilograms,

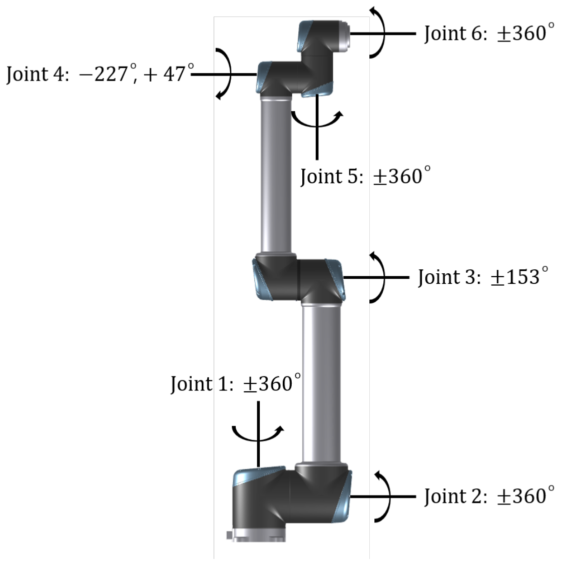


Figure 3.5: The collaborative UR10 robot with its angle limitations for the six joints.

the robot arm can reach positions within a radius of 1.3 meters. Furthermore, the robot arm is TÜV-certified (i.e., it fulfils the safety requirement that enables it to work in an environment near humans). The safety system includes built-in force-sensing that automatically stops the robotic arm when it collides with a human or obstacle, but it cannot detect an obstacle in advance and prevent a collision [43].

The software URSim was developed by UR for simulation of their robots such as the UR10 robot. URsim can provide different information such as the current configuration and velocity during the simulation of the UR robot. Unfortunately, the simulation of the robot dynamics due to inertia is limited. However, the result is sufficient for representing the controller's behaviour, test control concepts, and algorithms. A method to control either the simulated or the real robot is to send robot instructions through a network link. Therefore a switch between the alternatives can be done by changing the IP-address of the network link. All joints of the UR robot can move ±360° [44], but some special joint configurations result in self-collision. The third joint of the UR10's robotic arm has therefore its rotations limited to ±153° in IPS. Similarly, the fourth joint is limited to between −227 and 47° as shown in Fig. 3.5.

The simulated robot is controlled based on a set of "move joint" (movej) instructions sent to the controller using ZeroMQ. An instruction consists of angles for the six joints, a maximum joint speed with a default value of 1.0471975512 rad/s, a maximum acceleration with a default value of 1.3962634016 rad/s^2, and a motion blending radius. The robot controller uses the input to create a trajectory between the joint values in the set [44]. All motions along the path are linear in joint space (i.e., each joint moves linearly between the waypoints defined by joint angles) as shown in Fig. 3.6. The joint speed between two waypoints is therefore determined by the joints that rotate most during its motion. That joint is the limiting one and

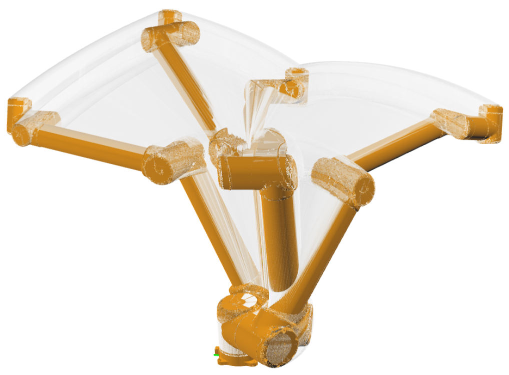


Figure 3.6: The robot path consists of a set of joint configurations. These configurations are denoted as waypoints and are shown as orange robots. The positions of the robot's joints are linearly interpolated between the waypoints.

rotates with the maximum speed defined by the input, whereas the rest of the joints are adjusted to reach the waypoint at the same time as the limiting joint. Furthermore, the blend radius determines the maximum distance from the waypoint the robot is allowed to move as shown in Fig. 3.7. When the robot is approaching a waypoint and start to move towards the next one, a blending is required to ensure that the robot does not reach a singularity in the waypoint. An increased blending radius enables decreased deceleration and acceleration zones as well as increased robot speed throughout the path.

3.2.2 Path Planning

The foundation and basics for most robot path planning methods are summarised in [31], but much has happened in the area since then. A survey by [38] covers and classifies the latest research about robotic motion planning in dynamic environments. The path planning algorithms usually minimise the path length, such optimisation in operational space requires complex and time demanding solutions of inverse kinematic. Therefore most algorithms use joint space instead, especially for real time systems that require fast solutions. Hence sampling-based techniques are useful in real time systems since they are known for their speed and simplicity. The sampling-based algorithms need fast and efficient methods for collision detection, nearest neighbour/graph search and graph representation, like the Probabilistic Roadmap Methods (PRM) and Rapidly-Exploring Random Trees (RRT) [18]. The PRM method first positions the robot in random joint configurations and saves the collision-free ones. Then two samples are gathered in a pair if there exists a collision-free path linear in joint space between them [4]. The RRT method can be bi-directional and can be described as two trees connected through their branches, the tree trunk are the goal and start configurations while the connections are collision-free paths linear in joint space between them. Each algorithm iteration

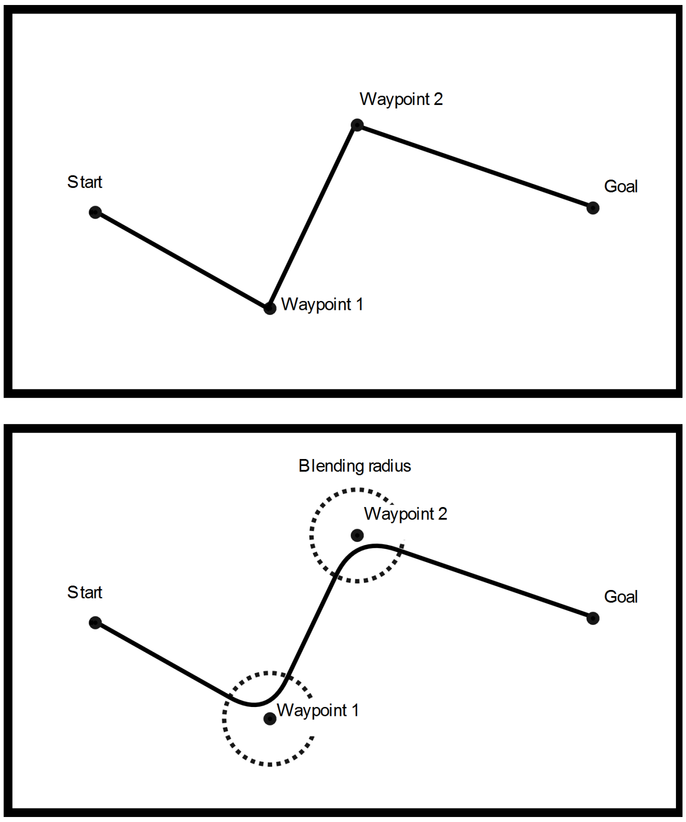


Figure 3.7: Blending radius between two paths segments is added to avoid singularities.

begins with expanding each tree towards a random configuration called attractor to find a collision-free path and continues until the trees branches overlap [30]. Both the PRM and the RRT algorithms are stochastic. This means that different solutions can be found every time and it's hard to predict the behavior of the path planning. Stochastic algorithms are therefore not wanted when working with industrial solutions.

The path planner used in IPS is instead inspired by both of these algorithms to create "a deterministic path planner that adaptively adjusts a grid in the configuration space" [35]. It has planned paths for many different projects, such as weld load balancing in multi-station sheet metal assembly lines [49] and optimisation of robotised sealing stations in paint shops [35].

3.2.3 Path Planning in IPS

A generated path is constructed by waypoints which the robot moves linearly in joint space between as shown in Fig. 3.6. The waypoints can be defined either as a position of the TCP (Tool Centre Point) in the Cartesian coordinate system of the scene or as a specific joint configuration for the robot. The waypoints defined with a TCP position can be reached by multiple joint configurations as shown in Fig. 3.8.

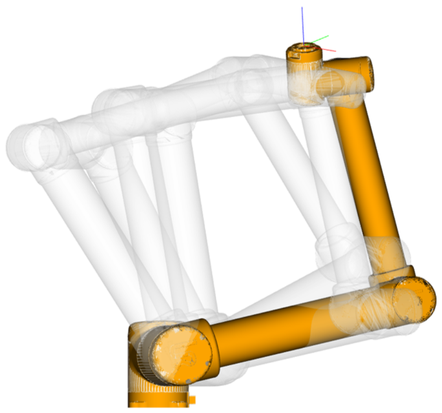


Figure 3.8: All the robot representations, where one configuration is highlighted, share the same TCP position.

Therefore, all the collision-free joint configurations that fulfil the required position for the TCP are calculated using inverse kinematics. Waypoint defined by a joint configuration is accepted as long as it is collision-free.

The next step is to generate an optimal path between the waypoints with the joint configurations as input. The robot motion between two waypoints is linear in joint space, hence the path with shortest angular movement is optimal. Therefore the path planner minimises the joint rotations by changing the order of the robot visiting the waypoints. For waypoints defined by a TCP position, the planner can also choose among those waypoints' multiple joint configurations. When a collision-free solution cannot be found due to obstacles, the path planner can insert one or multiple via points, even though this increase the complexity and calculation time.

The generated path consists of joint configurations in a specific order. From that, the robot controller creates a trajectory with blend zones between the configurations. Therefore, it is likely that the robot has an offset from the planned path at waypoints, especially if the robot is programmed to move fast and not slow down much at all waypoints.

3.2.4 Summary

The URSim simulator includes a controller that can generate a trajectory for the robot based on instructions. These instructions is a set of numbers that describe joint configurations. It is the path planner that finds this configurations and assures that collision free paths can be found in between them. It calculates path-segments between two waypoints and adds new joint configurations in between if a collision-free path segment can not be found. To enable fast movements between these configurations and make the transition smooth, blend zones/radius are created around the waypoints.

3.3 Prediction of Human Movements

To avoid collisions, we predict movements of both the human worker and the robot. After placing the virtual representations of the robot and the worker in their predicted poses, the distance between them is measured. The system takes actions if a collision is imminent (e.g., calculating a new path for the robot).

Since the robot's path is known the system can approximate its future trajectory with a constant velocity model. However, it is difficult to predict the next move of the human worker because his/her goals and intentions are unknown. To solve this problem, the constant velocity model is combined with a Kalman filter where the worker's measured position and his/her velocity computed by the Kalman filter are sent to the constant velocity model as input.

3.3.1 Human Prediction

Since the robot's movements influence the behaviour of the human worker around it, making decisions on how to change the robot path to avoid collisions is complex. In many decision systems, a prediction of the human is performed and used for these decisions. There are several methods for human prediction. One option is to use fixed assumptions and a model that describes how a human generally moves in the particular environment. Another option is learning-based prediction where the system is updated continuously. Alternatively, a network trained in advance can be used to approximate the human's motions [27].

3.3.1.1 Fixed Models

Since fixed prediction models do not learn from experience, they can only be applied to predict a limited number of behaviours. Despite that, several publications use simplified prediction models that, for example, make approximations that the human moves in the same direction with constant speed [27]. These simple models can also be expanded to include other behaviours such as a reduction of speed when walking in a curvature [15] and combining the facing and motion vectors using Qualitative Trajectory Calculus schema [16]. The idea behind the latter is to match the current motion with a similar prerecorded one from a library, containing motions where the human turns in different directions. Furthermore, the system in [16] uses the prediction together with the distance between the robot and the human to change the behaviour of the robot.

The algorithm in [14] performs two predictions during each iteration. After predicting the current motion one step ahead, the human's end position is predicted. Moreover, they use a Partially Observable Markov Decision Process (POMDP) to make decisions regarding how to move the robot based on the predicted intention of the human.

These basic motion models can be extended with obstacle avoidance and thereby correct the human prediction if a collision with obstacles in the surrounding is imminent. Potential fields are, for example, used for this purpose in [20] where the potential field results in a repellent force that acts on the human. Instead of enabling interaction, this method can be used to surround targets with attractive fields. Another option is presented in [20] where positive fields encourage the human prediction to move forward by acting as straight lines in that direction. Similarly, forces can be used to simulate ethological and human factors such as social rules. In [36], social rules are applied to the prediction as attractive and repulsive forces. Approach in [1] considers also multi-obstacles. They present a method that uses Monte Carlo sampling along the predicted path for all the humans and robots in the environment. The trajectory that minimises that the probability of a collision is then chosen to avoid a collision.

Another prediction method that uses stochastic processes and an occupancy grid map for human prediction is presented by [52]. The grid cells represent the area the human works at. The future steps are predicted by calculating the probability for moving from the current cell to the surrounding. This model represents uncertainties since all the surrounding cells are evaluated. Instead of a grid, the predicted human can also be represented with Gaussian distributions represented as elliptic areas [28]. The methods in [28] and [57] both include long and short time planning. Unfortunately, the “freezing robot problem” can occur if the uncertainty areas grow too much and block the robot. However, by using a model that captures “the non-Markov nature of agent trajectories” [57], the problem can be solved.

3.3.1.2 Learning Algorithms

Learning algorithms are usually trained on prerecorded data such as a set of recorded human motions. The evaluation of these prediction algorithms is often fast. A widely used concept is Neural Networks (NN) which mimics the human brain [37]. An NN is, for example, used for both one-step and long-term prediction in [14], where the NN is used together with an observable Markov decision process (OMDP) to determine the goal position of the human.

The framework Spatial Behaviour Cognition Model (SBCM) [8] is an algorithm based on social behaviour. The idea is to use a library of human motions with an algorithm that recognises behaviours for prediction in an arbitrary environment.

A similar framework presented in [48] observes human motions and considers the variation between individual operators. For example, if an operator has performed something once, it is likely the operator performs that task again and in the same order. The robot can then approach the human based on the observed behaviour and the previous knowledge. There are multiple systems developed for robots to approach a human. The algorithm in [17] detects if the human intends to interact and places the robot at an appropriate location. The method in [20] learns the framework from previous interaction experiences and changes its behaviour over time to better adjust to that human.

The framework presented in [2] can also learn patterns from human motions. It derives a hidden Markov model and uses probability when choosing a path. Based on the probability the path planning algorithm adjusts the robot behaviour. A similar framework [48] observes human motions and considers the variation between individual operators. For example, if an operator has performed something once, it is likely the operator performs that task again and in the same order. The robot can then approach the human based on the observed behaviour and the previous knowledge. There are multiple systems developed for robots to approach a human, the algorithm in [17] detects if the human intends to interact and places the robot accordingly. Similar to [20], this framework uses potential fields, but it is combined with a system that learns from previous interaction experiences and thereby adjusts to the human over time. The framework presented in [21] includes also human intentions and predicts a human arm with a model using underlying intended movement.

Instead of predicting the final goal of the human worker, the method in [54] uses waypoints the worker probably will move in-between. The waypoints are predefined based on functions and tasks in the environment or automatically selected from learned behaviour. From this, the method generates a likely path between the waypoints. The prediction is thereby based on a probabilistic model of human motions using a probability grid structured from observed human behaviour.

Within the field of prediction, Markov Decision Processes (MDP) are frequently used. The feature-based model developed in [62] uses an MDP together with maximum entropy inverse optimal control. The cost function is, therefore, able to change over time and can adapt to changes in the environment. Similarly, an MPD is used together with recognising sequence patterns in [32]. Thereby, the method can predict complex future movements by recognising small actions and some specific objects. They also use a Probabilistic Suffix Tree (PST) to model the Markov dependencies between the tasks. Another Markov based model, the Variable order Markov Model (VMM) [12] uses a hierarchical spatiotemporal pattern. The human tasks and the pattern are learned using a Hierarchical Self-Organising Map (HSOM).

The approach presented in [34] uses two assumptions; "the trajectory the human performs is optimal with respect to an unknown cost function" and "that human adaptation to their partner's motion can be captured well through iterative replanning with the above cost function". They adjust the cost function by using inverse optimal control and a set of performed human motions. However, the humans' goal position must be known to enable calculation of the motion. Moreover, tests are performed on a kinematic model of the human with 23 degrees of freedom to evaluate the framework.

3.3.2 Kalman Filter Theory

The Kalman filter in [47] describes the measurements and system dynamics with linear mathematical models and Gaussian nose approximations. It is, for example, used for state estimation by combining measurement and expected behaviour. Then the noise levels that correspond to the motion model and measurements accuracy are used to tune the filter. The model for the system dynamics is:

$$\mathbf{x}_k = \mathbf{A}_{k-1}\mathbf{x}_{k-1} + \mathbf{q}_{k-1}, \tag{3.1}$$

where $\mathbf{x}_k \in \mathbb{R}^n$ is the state vector for current time k, $\mathbf{A}_{k-1}$ is the matrix with linear equations for transition from the previous iteration $k-1$, and $\mathbf{q}_{k-1} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q}_{k-1})$ is the process noise at iteration $k-1$. The notation $\mathbf{D} \sim \mathrm{N}(\mu, \sigma^2)$ implies that $\mathbf{D}$ is a Gaussian noise with mean μ and variance σ^2. The measurement model is:

$$\mathbf{y}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{r}_k, \tag{3.2}$$

where $\mathbf{y}_k \in \mathbb{R}^m$ is the measurements, $\mathbf{r}_k \sim \mathrm{N}(\mathbf{0}, \mathbf{R}_k)$ is the measurement noise, and $\mathbf{H}_k$ is the measurement model matrix. In probabilistic terms, the result of the Bayesian filter equations for the linear models in Eq. 3.1 and 3.2 can be described using Gaussian distributions

$$\begin{aligned} p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) &= \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^-, \mathbf{P}_k^-), &(3.3)\\ p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) &= \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k, \mathbf{P}_k), &(3.4)\\ p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}) &= \mathrm{N}(\mathbf{y}_k \mid \mathbf{H}_k\mathbf{m}_k^-, \mathbf{S}_k), &(3.5) \end{aligned}$$

where $p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1})$ is the probability of $\mathbf{x}_k$ given measurements from the start until the time instance $k-1$ (i.e., $\mathbf{y}_{1:k-1}$). Kalman filter uses a prediction step and a update step. In the prediction the system model is evaluated,

$$\begin{aligned} \mathbf{m}_k^- &= \mathbf{A}_{k-1}\mathbf{m}_{k-1}, &(3.6)\\ \mathbf{P}_k^- &= \mathbf{A}_{k-1}\mathbf{P}_{k-1}\mathbf{A}_{k-1}^T + \mathbf{Q}_{k-1}. &(3.7) \end{aligned}$$

Then the update is calculated, where the predicted mean of the state vector $\mathbf{m}_k^-$ and corresponding covariance $\mathbf{P}_k^-$ are combined with the measurements. Hence the update step is

$$\begin{aligned} \mathbf{v}_k &= \mathbf{y}_k - \mathbf{H}_k\mathbf{m}_k^-, &(3.8)\\ \mathbf{S}_k &= \mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k, &(3.9)\\ \mathbf{K}_k &= \mathbf{P}_k^-\mathbf{H}_k^T\mathbf{S}_k^{-1}, &(3.10)\\ \mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k\mathbf{v}_k, &(3.11)\\ \mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T. &(3.12) \end{aligned}$$

Resulting in the estimated state vector $\mathbf{x}_k \sim \mathrm{N}(\mathbf{m}_k, \mathbf{P}_k)$. The filter is initialised with $\mathbf{m}_0$ and $\mathbf{P}_0$.

3.3.3 Summery

Since the position of the human worker is known in our system, we can approximate the velocity of the human worker using a Kalman filter and use it to predict the future movements of the human worker with a linear velocity model.

4 Concept and Implementation of a Collaborative Robot System

Our collaborative robot system consists of four parts: tracking and representation of a human in a digital environment, path planning and control of a robot, prediction of movements for the human and the robot, and the main control program. To achieve this, four IPS-scenes listed below are used. The system is shown in Fig 4.1 together with the flow of data.

- The real time IPS-scene
- The Prediction IPS-scene
- The Path-Planning IPS-scene
- Manikin update IPS-scene

The human motions are tracked using an Xsens IMU sensor-suit connected to the Xsens MVN Analysis software that processes the data and creates a human representation, an Xsens-manikin. In the real time IPS-scene, an IPS-manikin follows the position and orientation of the Xsens-manikin's different body parts.

Similar to the human, the robot is represented in the real time IPS-scene to enable path planning and distance measurements between them. To this end, the robot simulation in URSim (hosted on a separate Linux computer) is connected to the real time IPS-scene, where the IPS robot configuration is constantly updated based on the simulated URSim robot.

The simulated robot moves along a given preprogrammed path between tasks. The path is followed as long as a minimum separation distance between the human and the robot is maintained. However, if the separation distance is violated (e.g., when the human collides with the robot) a new robot path is needed. Consequently, a new collision-free path is calculated in the Path-Planning IPS-scene with the use of the predicted human pose from the Prediction IPS-scene to replace the configurations that violate the separation distance. This path consists of a set of robot configurations and the duration of every inter-path (i.e., the time for the robot to perform the path between the waypoints). Finally, the path is sent to the simulated robot

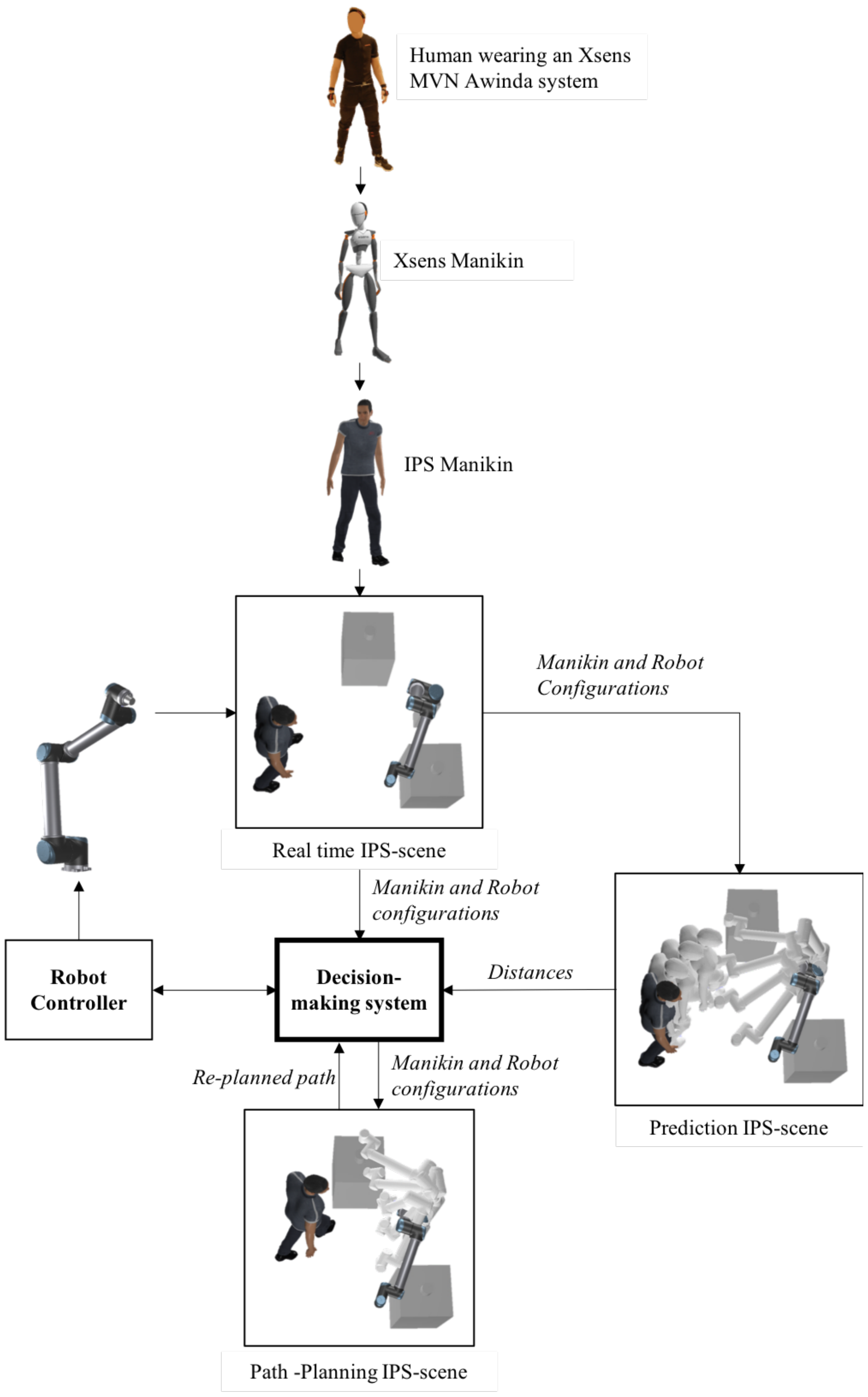


Figure 4.1: Our collaborative robot system and the communication network between the IPS-scenes and the decision making system.

to be executed.

Next, the human motions are predicted with a motion model to anticipate any violation of the minimum separation distance. Firstly, the joint angle and the position from the manikin in the real time IPS-scene are filtered with a Kalman filter. The filter returns approximations of the velocities of the manikin and its limbs. Secondly, the velocities, position, and angles are used as input to a constant velocity model, which compute a predicted pose and position of the manikin. Thirdly, the predicted manikin pose is represented in the Prediction IPS-scene and used to compute future clearance to the robot. During the distance calculation, the manikin is placed in its predicted pose while the robot moves in steps to the joint configurations that represent its future path. For each step, the distance between the predicted manikin and robot is measured and sent to the decision system. Finally, the distances are used to identify if any violation of the minimum distance is likely by comparing to the minimum threshold. If violation is likely, the controller calculates a new path and decides whether the robot should continue along the current path, decelerate, or stop during the calculation. This decision is influenced by, for example, the estimated time until the violation occurs.

4.1 Framework and Communication

The algorithms for prediction and decision were written in C++ and embedded in the decision system which communicates with three different IPS-scenes and the UR-robot simulation. The communications to the IPS-scenes are handled through the LUA programming language. LUA-scripts are executed from the decision system as either existing scripts or by creating text-strings written in LUA-language in real time. Moreover, the communication is bidirectional, and information from the IPS-scenes is gathered and sent back to the decision system using the JavaScript Object Notation (JSON) format.

4.2 Human Tracking and Representation

The general concept of tracking the human is shown in Fig. 4.2. The system used for tracking the positions and orientations of the human limbs is an Xsens system consisting of a set of IMUs attached to the human. After processing the data from the sensors in the MVN-Analyze software, the positions of the body segments are sent from MVN to the real time IPS-scene and visualised. In this IPS-scene, the manikin is positioned according to the positions of the body segments from MVN by using the control points shown in Fig 4.3.

To create a virtual representation of the human, we measure some specified body parts and give the data to Xsens MVN-Analyze and IPS. From this information,

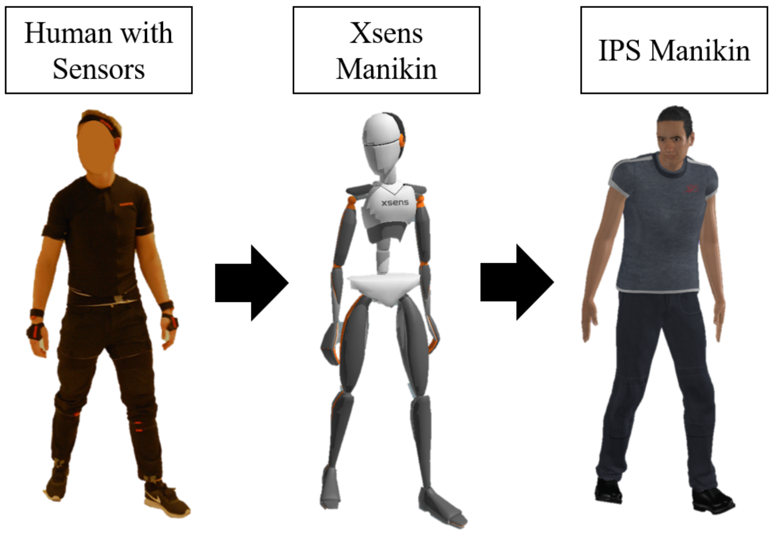


Figure 4.2: The human's movements are tracked with sensors. After processing the sensor data in MVN-Analyze, the filtered data are sent to an IPS-scene.

both programs can generate a manikin with anatomy similar to the real human. Furthermore, the manikin in MVN-Analyze is connected to the real human, with the use of 17 sensors attached to the human's limbs. The data from the sensors are filtered before they are used by the manikin model. Since the data have to be filtered, the sensors require a configuration sequence where the human makes slow motions in order to tune the time-changing parameters of the filter. After the calibration, the positions of the segments along with their contemporary time are streamed through a network and are received by IPS. The information is used to calculate the pose of the IPS manikin and thereby connect the IPS manikin to the MVN manikin.

The pose of the IPS manikin is calculated using an algorithm that finds the most ergonomic pose given the control point constraints corresponding to positions and orientations of manikin's joints as Cartesian coordinates in the room. The body limbs used when linking the manikins together are pelvis (denoted L5 or L3L4), neck, head, both hands, and both feet. These control points on the IPS manikin are visualised in Fig. 4.3. Since the manikin skeletons differ, some positions may need to be displaced/shifted before the data is used to position the IPS manikin. If the mapping between the positions is not correct, it will affect the calculation time of the IPS manikin since it would be harder to find a suitable ergonomic position.

Furthermore, to verify the mapping of the two manikins, a comparison test is performed. In the recorded sequence used for the test, a man takes a step forward, raises his hands, takes a step backward, and repeats the procedure once more. The segments that are used for the comparison are listed in Table 4.1, which include all the segments used to map the two manikin models plus two additional segments (i.e., toes and the upper arms/elbows) that are used to compare the models. The results are shown in Fig. 4.4 and 4.5 where the data sent from Xsens are plotted together

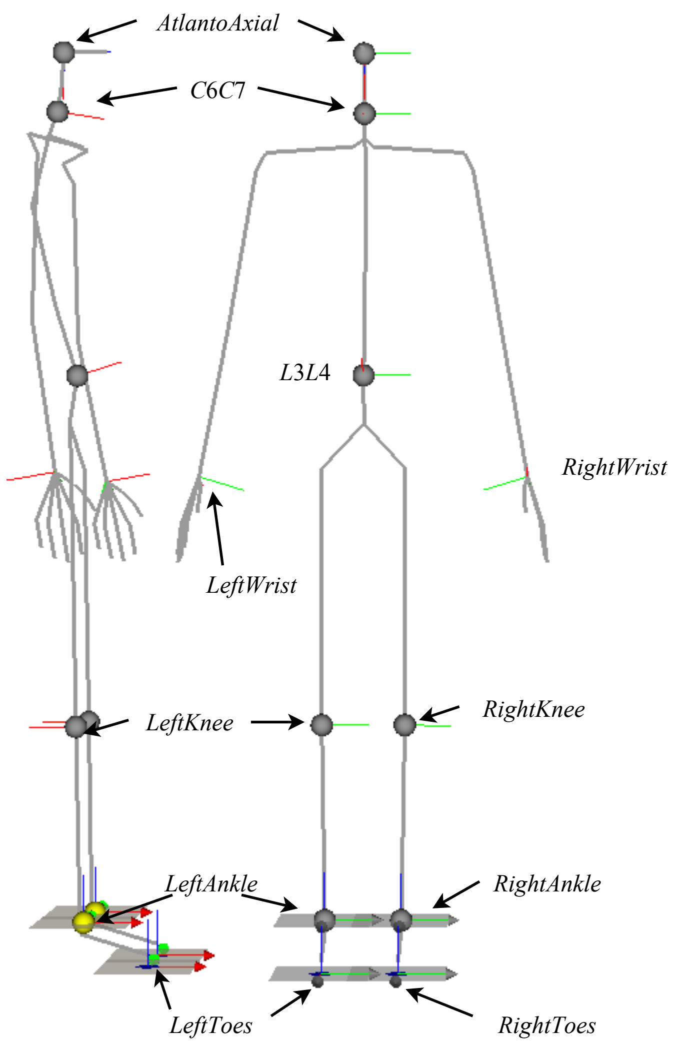


Figure 4.3: Control points used for moving manikins in IPS.

Xsens		IPS	
Number	**Name**	**Number**	**Name**
2	L5	2	L3L4
6	Neck	7	C6C7
7	Head	8	AtlantoAxial
11	RightHand	51	RightWrist
15	LeftHand	31	LeftWrist
18	RightFoot	19	RightAnkle
22	LeftFoot	14	LeftAnkle
19	RightToes	20	RightToes
23	LeftToes	15	LeftToes
9	RightupperArm	25	RightElbow
13	LeftupperArm	31	LeftElbow

Table 4.1: Segments used to link the Xsens manikin to the IPS manikin.

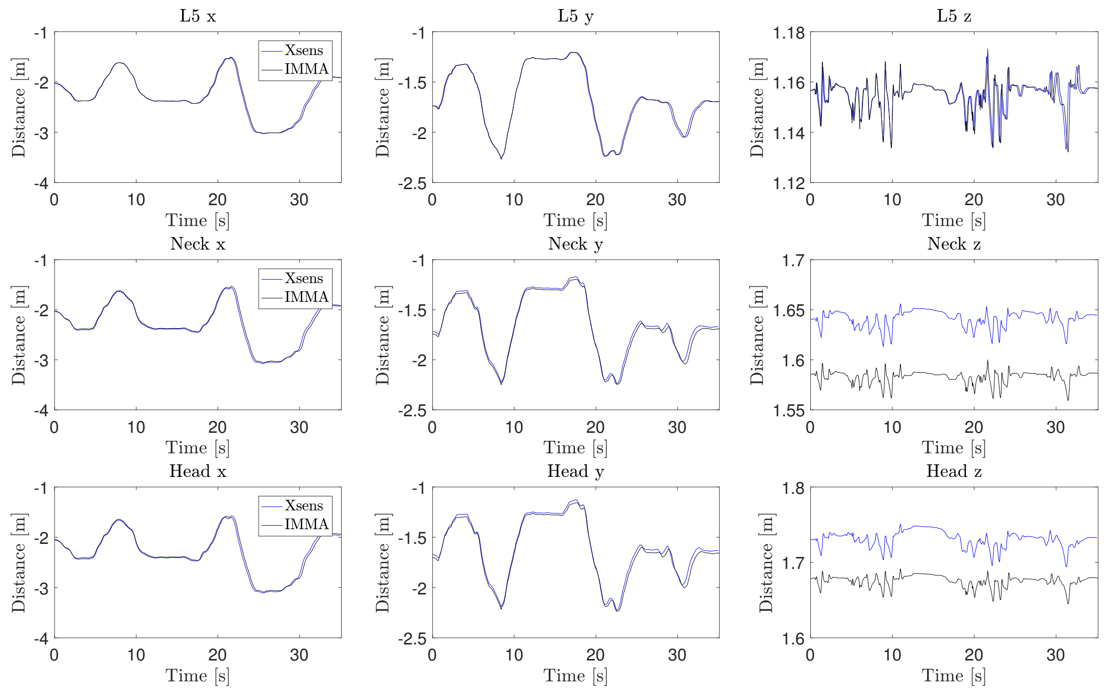


Figure 4.4: A test comparing the link between segments in the manikin's upper body.

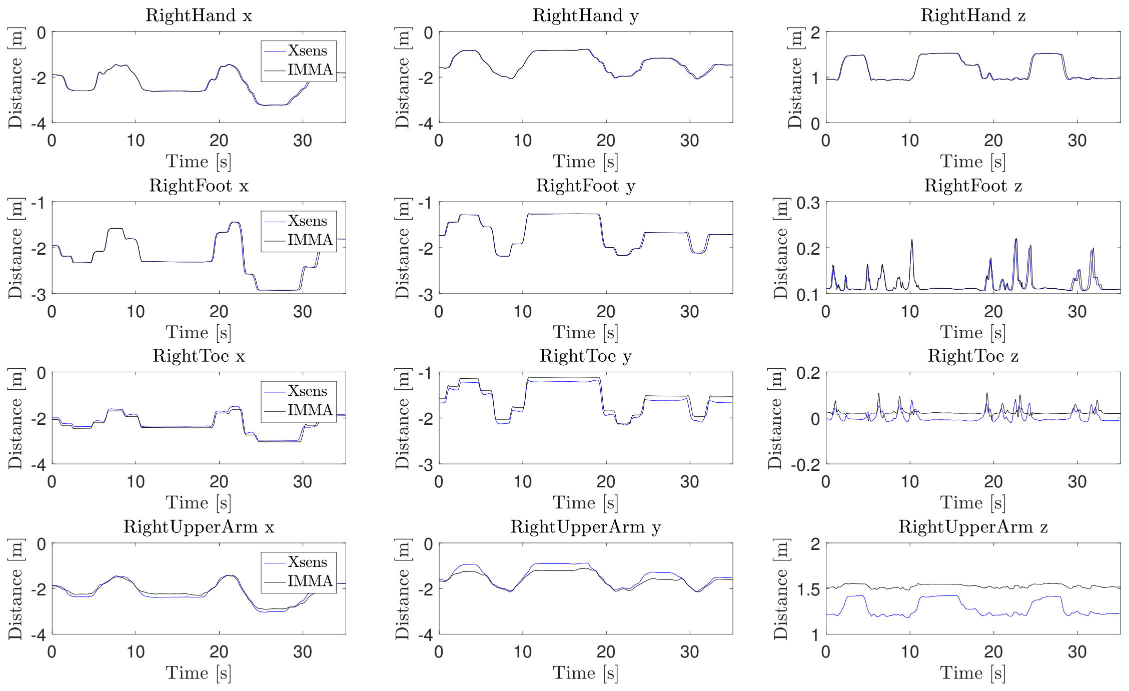


Figure 4.5: A test comparing the link between segments in the manikin's right side.

with the positions of the control points in IPS after the pose is calculated by IMMA. The lines in the figure correspond to the measurements and should be overlapping if no offset distance is required due to the differences in the used manikin models. Although, the offsets between some of the segments can be clearly seen in Fig. 4.4 where the neck and the head differ in the z-direction. Different poses between the models are also observed in Fig. 4.5 where the two segments not used for mapping (i.e., the toes and upper arms) differ between the models. In these measurements, some main trends can be seen such that the IPS manikin tends to find slightly different locations for the upper arms. The differences are due to differences in the length of the manikin segments and because the IPS manikin may produce better ergonomic poses than the human actually performed. The difference in length of the segments can though be further tuned since a different pose of the manikin may affect the path planning and thereby the robot's ability to maintain a minimum distance to the human.

4.2.1 Sensor Drift

The sensor units used are IMUs which consist of an accelerometer, a gyroscope and a magnetometer. These are used together to compensate for problems associated with the different measurement techniques. An accelerometer often gives a clear measurement of the direction of gravity. Since the position is retrieved through integration, it might drift and is therefore not fully reliable. The same integration problem occurs for a gyroscope. A magnetometer gives the direction in the plane perpendicular to gravity and thereby provides the orientation of the human in the room. This sensor is not reliable if the magnetic field in the room is changing. Hence, both location and orientation of the human in the room can drift. This is typically observed after fast movements or near a disturbing magnet field.

To measure the drift of the sensors, a test was performed where the position data in x- and y-directions for Link L5, from the MVN Analyse software, are observed. The observed position at link L5 is one of the positions used to map the manikins. Since it is located close to the lower spine of the manikin, it gives an average position of the human. The result of the test is shown in Fig. 4.6, where the measurements from the sensors are given as positions in the room viewed from above. The corresponding velocities are shown in Fig 4.7. During the test, a human walked in the shape of an hourglass with the outer dimensions 2 x 4 meters. The human started in a position with coordinate (0,0). Then the human walked to coordinates (4,-2), (4,0), (0,-2) and returned to (0,0). The path is about 13 meters and the resulting drift is approximately 0.78 meters, which corresponds to 6% of the path length. No major magnetic disturbance sources such as computers or machines were present during the recording. Hence, the results indicate a drifting problem.

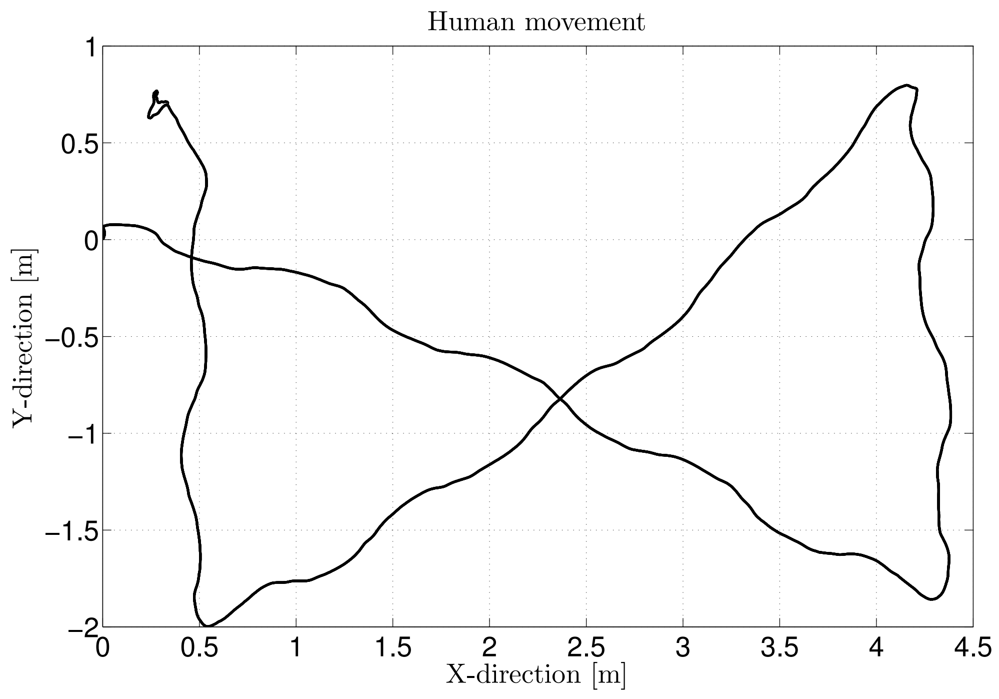


Figure 4.6: Human starts in position (0,0), moves to (4,-2), (4,0), (0,-2), and back to (0,0).

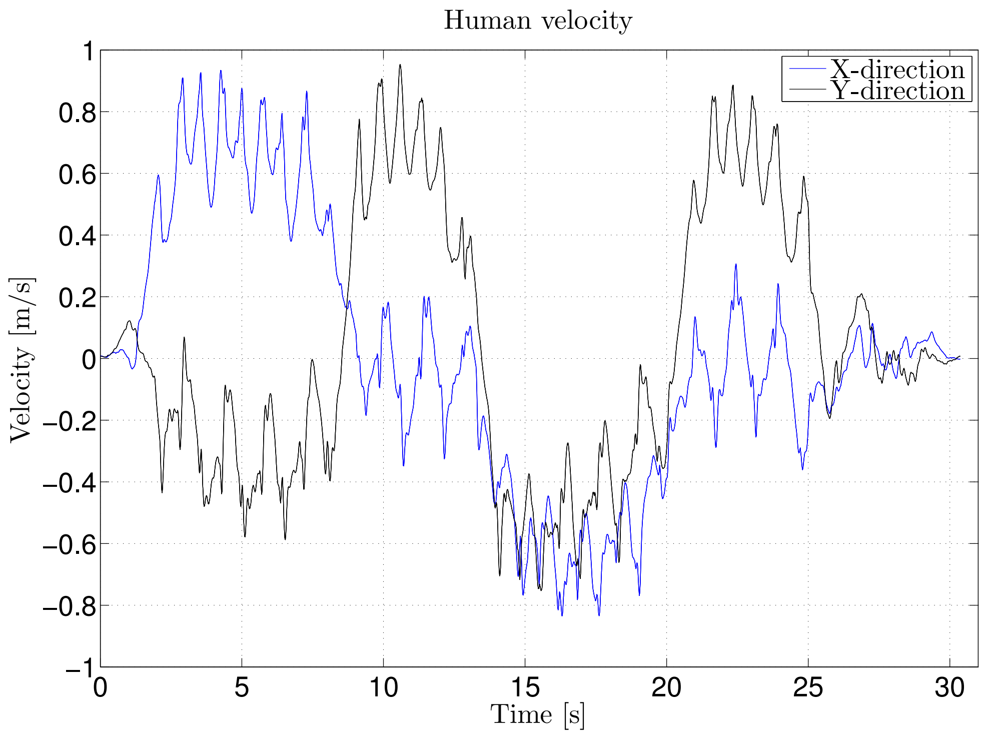


Figure 4.7: The velocity in x- and y-directions of the movement shown in Fig. 4.6.

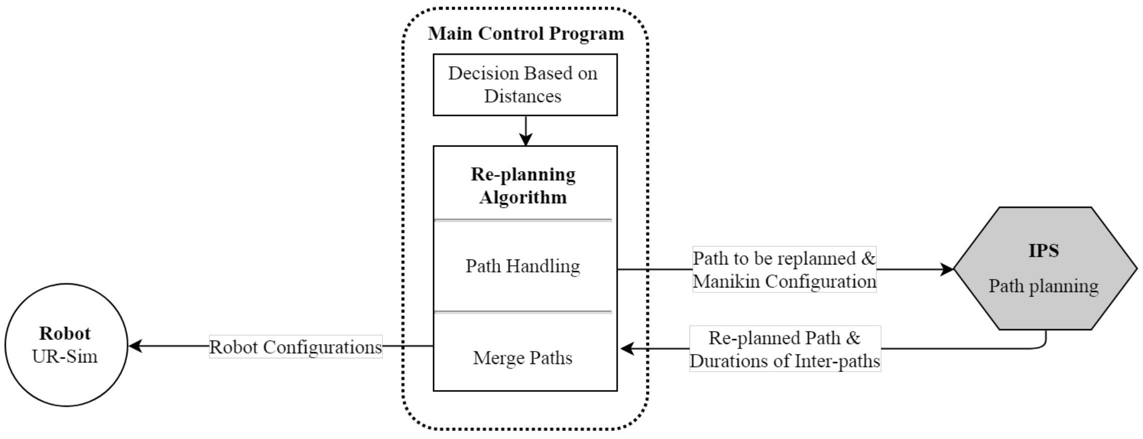

Main Control Program
Decision Based on Distances
Re-planning Algorithm
Path Handling
Merge Paths
Path to be replanned & Manikin Configuration
IPS
Path planning
Re-planned Path & Durations of Inter-paths
Robot Configurations
Robot
UR-Sim

Figure 4.8: Overview of the part of the framework used for path planning.

4.2.2 Number of Sensors

The seven control points used can depict the essential behaviour of the human. However, fewer control points are desirable since it will decrease the time it takes to calculate the ergonomic manikin pose in IPS. Fewer control points could be used if only parts of the human need to be tracked. For example, if the robot moves in a space where only parts above the waist of the human could be reached and the pose of the legs does not need to be tracked. Another approach is to use fewer control points when the human is positioned far away from the robot where a collision is unlikely.

However, Xsens require all the 17 sensors in order to calculate the positions of the segments. Xsens uses the manikin model as a reference to compensate for sensor data drifts. If only the filtered sensor data (not matched to their manikin skeleton) is used, it is likely that body parts of the manikin would drift away from each other.

4.3 Robot Path Planning

The concept of how the path planning for the robot is performed is shown in Fig 4.8. The robot is simulated in the software URSim and is controlled by a set of robot instructions sent to the simulator. Each instruction consists of joint values for the six joints and the maximum allowed joint speeds and accelerations.

Initially, the robot follows a path that is recalculated according to the following description if the minimum distance is violated. Firstly, data regarding the pose and position of the human, together with the part of the path that violates the clearance, is collected. Secondly, a new path that maintains the clearance is calculated and sent to the decision system. Thirdly, the new path is merged with the previous one and finally sent to URsim. In addition, the new path is also used in future predictions.

4.3.1 Path Generation

The initial robot path is calculated from a set of TCP positions by the IPS path planner. The resulting path is both feasible and linear in joint space. Moreover, the path consist of a set of waypoints where each waypoint corresponds to a specific set of joint angles (i.e., a joint configuration). Thereby the segment between two consecutive waypoints can be recalculated, while still guarantees to fit the existing path.

The path planning is done in the Path-Planning IPS-scene where digital copies of the manikin and the robot are visualized, moved, and controlled. The decisions regarding how and when a new path should be calculated are made in the decision system. The control program collects information regarding start position, goal position, current manikin pose, and the minimum distance allowed between the robot and the human. Both the start- and goal-positions for the robot need to be defined as waypoints when used for path planning. Whereas the goal position is always given as a number corresponding to an existing waypoint, the start position is either given as a number or defined as a joint configuration. If given as a joint configuration, the robot in the Path-Planning IPS-scene is placed in that configuration, and a corresponding waypoint is created and used. When the start and goal are determined, the manikin is placed at the position predicted by the prediction algorithm. Then a “shell” that consists of a mesh with about 40 000 triangles is created around the manikin to be used by the shortest distance algorithm during path planning to maintain the clearance.

Once the manikin is in position, the path planner tries to find a feasible path that maintains the minimum distance between the robot and the manikin. If such a path does not exist, the planner attempts to solve the problem by adding waypoints between the existing once. For the case where this does not work the planner returns the best path with respect to joint space, despite that it violates the minimum distance. The resulting path will probably move the robot to a position where the decision system stops the robot due to too small distance in the real time IPS-scene.

Next, the new path is sent from the path planner to the decision system as a set of waypoints and inter-path times (i.e., the approximated time it takes for the robot to move between the waypoints). The new waypoints are merged with the current path and then sent to the robot controller and the prediction algorithm. After that, the calculated path in the Path-Planning IPS-scene is deleted.

4.3.2 Robot Velocity

The velocities and accelerations during the motion of the robot are, for example, used by the path planner in IPS and in the decision system by the prediction step. They are approximated by a model based on constant velocity and an infinite maximum acceleration. While the robot moves slowly this motion model follows the real

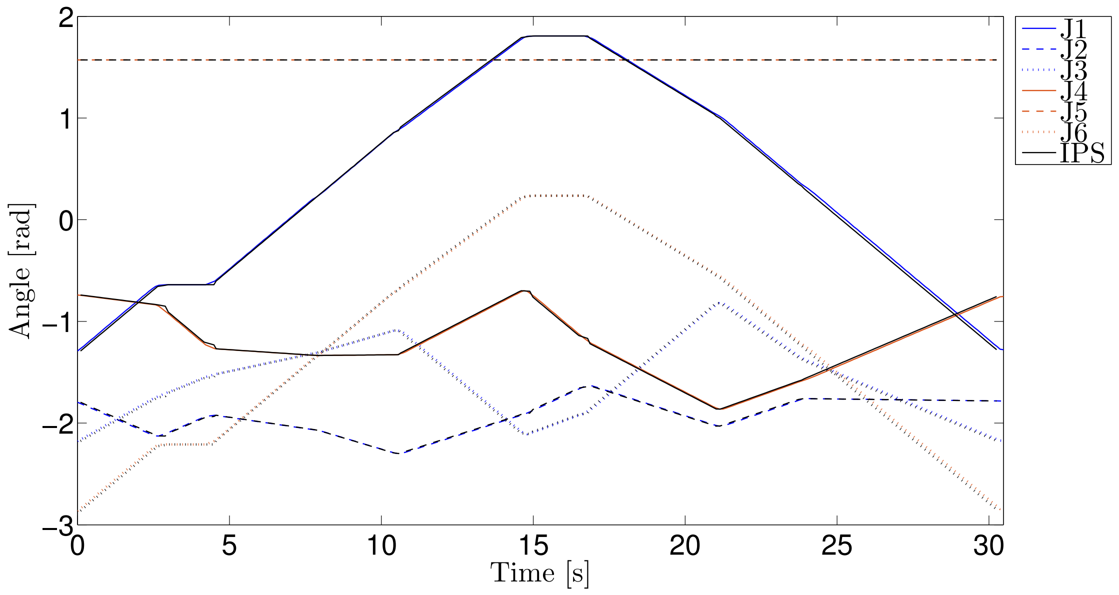


Figure 4.9: A comparison between the predicted configuration of the robot, shown in black lines, and the measured angles of the robot simulated by URSim, shown in coloured lines. The prediction from the path planner in IPS matches the actual behaviour if the maximum joint velocity is set to 0.25 rad/sec.

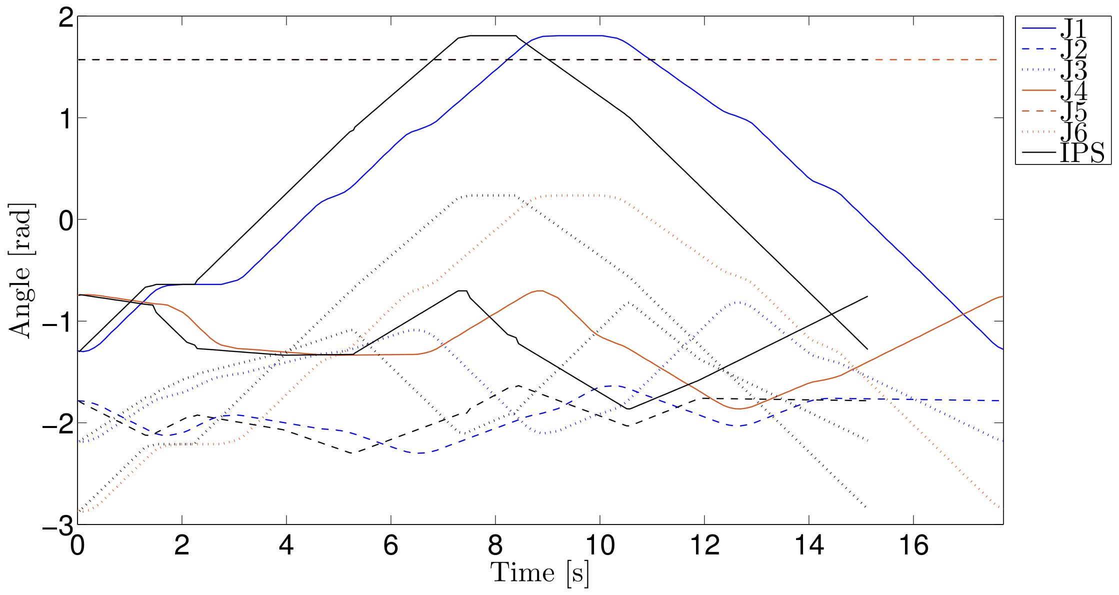


Figure 4.10: A comparison between the predicted configuration of the robot, shown in black lines, and the measured angles of the robot simulated by URSim, shown in coloured lines. The prediction from the path planner in IPS does not match the actual behaviour when the maximum joint velocity is set to 0.5 rad/sec.

Waypoint	**X coordinate [mm]**	**Y coordinate [mm]**	**Z coordinate [mm]**
1	-1000	600	600
2	-500	800	800
3	0	800	1000
4	500	800	800
5	1000	600	600

Table 4.2: Coordinates of the robot TCPs used as waypoints on the initial path.

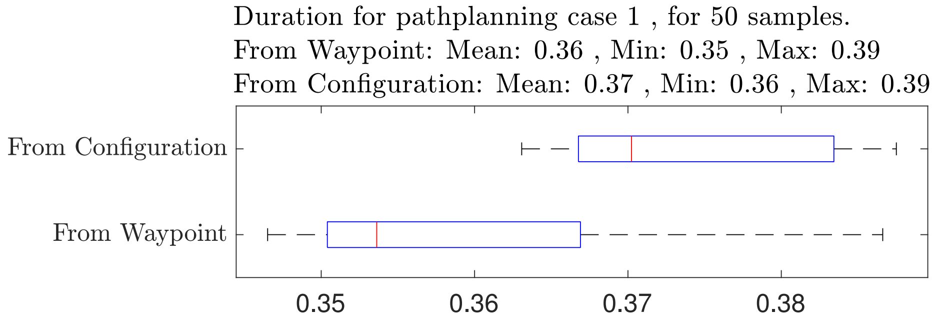
velocity but for faster movements the result is poor. This can be seen in Fig. 4.9 and 4.10, where the poor result can be attributed to torque limits of the robot joints and deceleration before a turn. To make the acceleration more accurate, the path can be changed to include large blend zones. This affects the speed since it decreases the acceleration zones and the robot can move faster when passing a waypoint as shown in Fig 3.7.

The motion model is tested with two different limitations for the maximum joint velocity. The results are shown in Fig. 4.9 and 4.10. In the test, a set of waypoints and a maximum velocity is used during the path planning in IPS to calculate the duration of the path. The URSim gets the same set of waypoints and maximum velocity to simulate the robot motion. For all the time instances, the joint configurations of the simulated robot are plotted together with the approximated joint configuration calculated by IPS. Furthermore, the results show that high velocities require acceleration zones and the constant velocity prediction produces poor results when using a velocity of 0.5 rad/s. Consequently, the lower maximum velocity (i.e., 0.25 rad/s) is used in the framework.

4.3.3 Evaluation of the Path Planner

To decide how to act when a violation of the minimum distance is detected, the expected time until it happens is compared with the duration of planning a new path. However, the planning duration changes for each case and the decision system needs it before the calculation of the new path has started. Therefore a set of test cases are evaluated to estimate a mean duration to be used for the comparison.

To make the time estimation representative test cases must be chosen carefully. The duration that is estimated starts when the configurations that violate the minimum distance are known. Since it also involves some planning steps, they are copied to another algorithm that carries them out several times. The first step for the algorithm is to generate the LUA code that carries out the path planning and send it to the Path-Planning IPS-scene. After that, the manikin is moved to the predicted pose that violates the minimum distance, and is used to calculate a new path. The last step is to send the result back to the decision system and interpret the the resukt given in JSON format. The test cases are made representative by varying the parameters that change when the system is used such as if the planning is done from



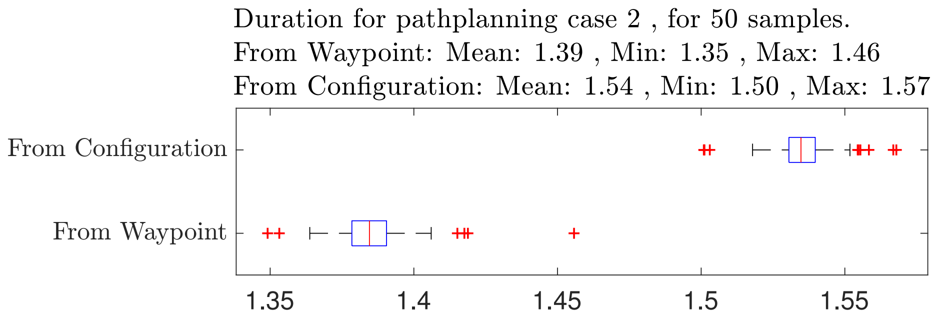
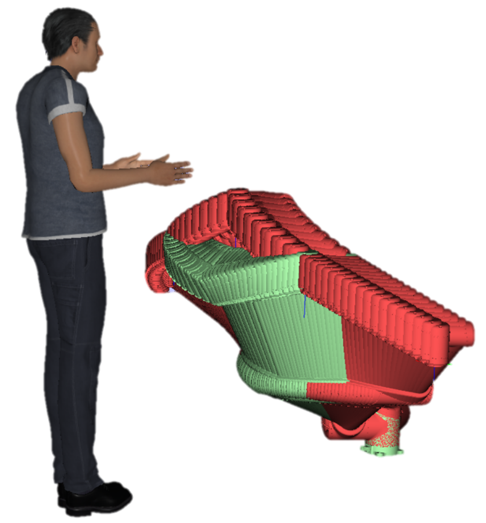


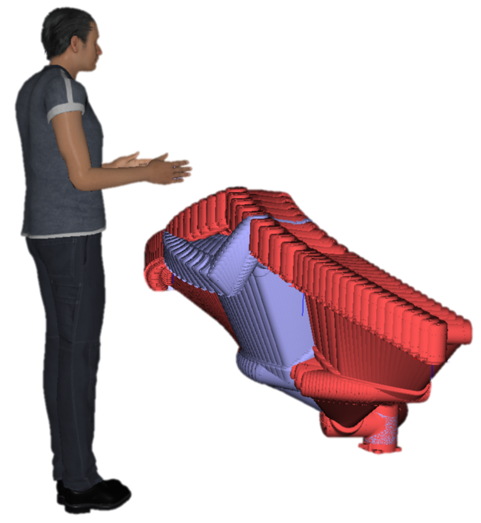
Figure 4.11: The time distribution of the function that handles path planning as a box plot for each case, where each red cross corresponds to an outlier, each black crosshatched line shows the span from the minimum to the maximum, each blue box spans from the first to the third quartile, and each red line marks a median. Each case was repeated 50 times.

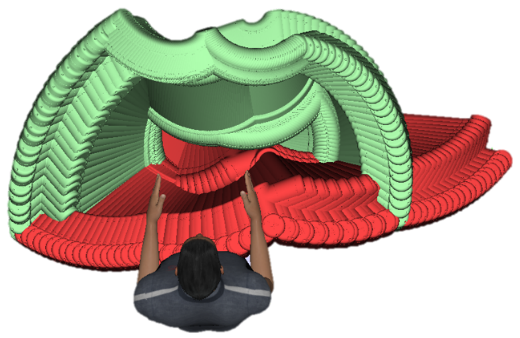
an existing waypoint or the configuration the robot is at, and the distance from the manikin to the robot. For each variation, the test is carried out 50 times and the result is shown in Fig. 4.11. Additionally, the initial waypoints used for the test path are given as TCP positions in Table 4.2.

The variation led to four different test cases that are combinations of the following: the manikin stands either in the robot path or just within violation distance; the path is calculated from an existing waypoint or from a joint configuration with shorter distance to the manikin. These four cases are shown in Fig. 4.12, where the robot moves from the right from the manikin's perspective. In the figure, the initial path is shown in red, the path calculated from the waypoint is shown in green, whereas the path calculated from the joint configuration is shown in lilac.

Between the 50 samples of each case, a small variation in duration is identified. Furthermore, the test shows that the calculation time of a new path increases if the distance between the robot and manikin decreases. This is probably why the test result shows that planning from a waypoint is faster than from a configuration. The figure shows also that the path planning requires more time when the robot passage is narrow.


(a) Case 1: The new path in green was planned from waypoint to waypoint.


(b) Case 1: The new path in lilac was planned from a configuration to a waypoint.


(c) Case 2: The new path in green was planned from a waypoint to a waypoint.

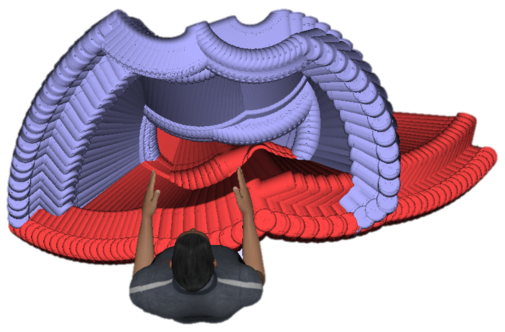

(d) Case 2: The new path in lilac was planned from a configuration to a waypoint.

Figure 4.12: The cases used for time evaluation tests of the path planning function. The initial robot path is marked as red, starting from the right in the manikin perspective. The replanned path that considers the manikin pose is shown in green or lilac.

4.4 Prediction of Human Action and Robot Motion

We use a prediction algorithm to detect likely future violations of the minimum safety distance. The algorithm places the manikin in a pose: the combination of position and orientation, that represents a prediction 1 second into the future. After that, the robot path is represented by moving the robot step by step $T_{seconds}$ seconds along the robot path and the shortest distance between the robot and the manikin is calculated. Since it is crucial to avoid collision, the distance information is used when computing new robot paths. To make the transition to the new path smoother, $T_{seconds}$ can be increased to allow for a longer prediction time and thereby detect the likely collision well in advance.

4.4.1 Human Action Prediction

The skeleton for the IPS manikin consists of 162 joints that can be controlled individually to create a certain manikin pose. However, not all joints can be used during the predictions due to the limitation in computer capacity. Therefore, a constant state model (CSM) is used for prediction of the joint angles in the spine and legs. The model assumes that the state is constant during prediction. Furthermore, the fingers are not tracked during the motion capture and hence excluded from the prediction.

For safe human-robot collaboration, the arms' movements are especially important, because they are usually close to the robot. Therefore, the joints Translation, Right- and Left-GH, -ShoulderRotation and -Elbow are used in the prediction as shown in Fig. 4.13. The translation joint corresponds to a position given in the Cartesian coordinates in the world frame. It is placed close to the human spine since that is a good centre point. This joint is, therefore, less likely to move a long distance between two samples compared to the rest of the human body, and hence can be regarded as reliable in a prediction. Together with the other joints in the manikin arms, they define the poses of the human arms.

The predictions of the joints are performed using CVM which assumes the joint velocity to be constant during the predicted time dt_{pre}. The model is

$$x_{pre} = x + dt_{pre}\dot{x}, \tag{4.1}$$

where x is the position/angle of the joint, $\dot{x}$ the joint velocity/angular velocity and x_{pre} the predicted position/angle. To complement the constant velocity model, the joint limits for the arms in IPS biomechanical model are implemented as limits for the prediction. The values are shown in Table 4.3. If a joint angle is out of range, the limit is used instead.

Since the joint velocities/angular velocities are not accessible from the IPS manikin,

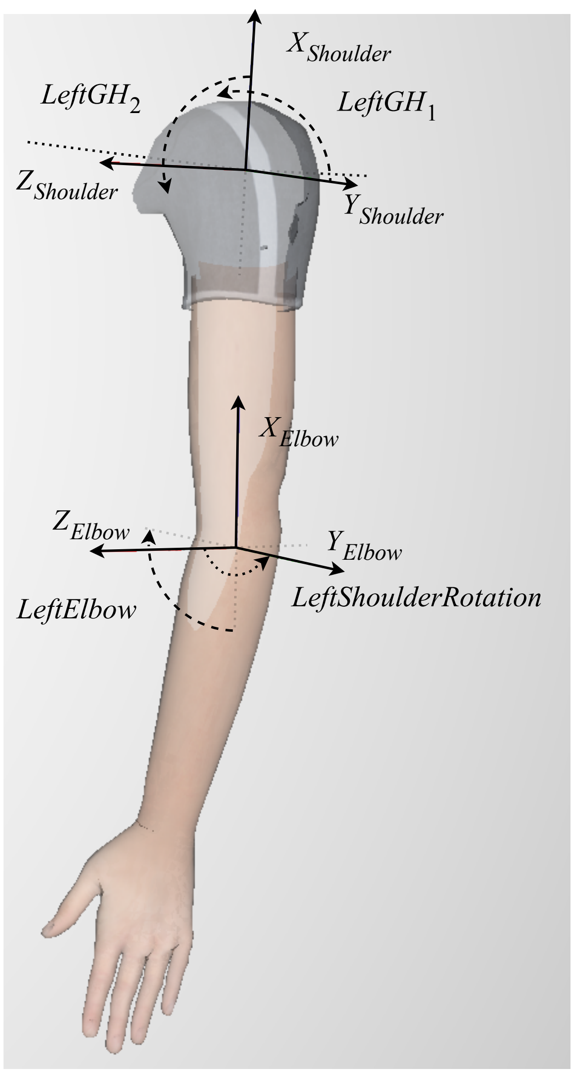


Figure 4.13: The joints in the manikin's left arm used by the constant velocity model for prediction.

Joint name	Maximum limit	Minimum limit
$Left/RightGH_1$	100 °	-70 °
$Left/RightGH_2$	170 °	-61 °
Left/RightShoulderRotation	35 °	-85 °
Left/RightElbow	160 °	-1 °

Table 4.3: Limits for the joints in the IPS biomechanical model used in prediction.

State	**Unit**	**Motion noise q**	**Measured**	**Measurement noise r**
$Translation_x$	m	0	Yes	1
$TranslationVelocity_x$	m/s	0.70	No	0
$Translation_y$	m	0	Yes	1
$TranslationVelocity_y$	m/s	0.70	No	0
$Translation_z$	m	0	Yes	1
$TranslationVelocity_z$	m/s	0.20	No	0
$LeftGH_1$	rad	0	Yes	1
$LeftGHVelocity_1$	rad/s	0.048	No	0
$LeftGH_2$	rad	0	Yes	1
$LeftGHVelocity_2$	rad/s	0.15	No	0
$LeftSholderRotation$	rad	0	Yes	1
$LeftSholderRotationVelocity$	rad/s	0.50	No	0
$LeftEilbow$	rad	0	Yes	1
$LeftEilbowVelocity$	rad/s	0.050	No	0
$RightGH_1$	rad	0	Yes	1
$RightGHVelocity_1$	rad/s	0.048	No	0
$RightGH_2$	rad	0	Yes	1
$RightGHVelocity_2$	rad/s	0.15	No	0
$RightSholderRotation$	rad	0	Yes	1
$RightSholderRotationVelocity$	rad/s	0.50	No	0
$RightEilbow$	rad	0	Yes	1
$RightEilbowVelocity$	rad/s	0.050	No	0

Table 4.4: The states used in the Kalman filter along with the values used for motion and measurement noises.

they must be estimated to enable prediction using CVM. A Kalman filter (see Kalman filter theory in Section 3.3.2) is used for the estimation where the joint motion is described with a constant velocity model and an added Gaussian noise. In the filter, 22 states are needed (11 to describe the value of the joints and the other 11 to describe the corresponding velocity). Furthermore, three of the 11 states describe the position of the manikin in the world frame and four states describe the joint angles in each arm. The states are shown in Table 4.4 together with the resulting measurement and motion noises. Moreover, the joint values from the manikin in the real time IPS-scene are used as measurements. Therefore the measurement model is the measured value including a Gaussian noise.

The prediction algorithm for joints is shown in Alg. 1. The measured joint data, Y, is collected from the real time IPS-scene and sorted into j_{CSM} - joint values that are predicted with the constant state model, and j_{CVM} - joint values. The velocities for the j_{CVM} joints are estimated using a Kalman filter, described in Chap. 3.3.2, and predicted using a linear constant velocity model. A joint is limited if the value exceeds one of its limits. Then all the predicted joints are sent to the prediction IPS-scene to update the manikin pose.

Algorithm 1 Human prediction

```
1: Procedure human prediction ($\boldsymbol{j}_{CSM}$, $\boldsymbol{j}_{CVM}$)
2: $\boldsymbol{x}_{KF}$ =KalmanFilter($\boldsymbol{j}_{CVM}$)    ▷ Filter the measurements
3:     $\boldsymbol{x}_k^- = \boldsymbol{A}_{k-1}\boldsymbol{x}_{k-1}$    ▷ Kalman filter prediction part
4:     $\boldsymbol{P}_k^- = \boldsymbol{A}_{k-1}\boldsymbol{P}_{k-1}\boldsymbol{A}_{k-1}^T + \boldsymbol{q}_{k-1}$
5:     $\boldsymbol{v}_k = \boldsymbol{j}_{CVM,k} - \boldsymbol{H}_k\boldsymbol{x}_k^-$    ▷ Kalman filter update part
6:     $\boldsymbol{S}_k = \boldsymbol{H}_k\boldsymbol{P}_k^-\boldsymbol{H}_k^T + \boldsymbol{R}_k$
7:     $\boldsymbol{K}_k = \boldsymbol{P}_k^-\boldsymbol{H}_k^T\boldsymbol{S}_k^{-1}$
8:     $\boldsymbol{x}_k = \boldsymbol{x}_k^- + \boldsymbol{K}_k\boldsymbol{v}_k$    ▷ $\boldsymbol{x}_k$ is a vector with the states from Table 4.4 at iteration $k$
9:     $\boldsymbol{P}_k = \boldsymbol{P}_k^- - \boldsymbol{K}_k\boldsymbol{S}_k\boldsymbol{K}_k^T$
10:    $\boldsymbol{x}_{KF} = \boldsymbol{x}_k$
11: $\boldsymbol{x}_{pred} = \boldsymbol{A}_{pred}\boldsymbol{x}_{KF}$    ▷ Calculate predicted joints
12: for i = 1:m do    ▷ Check if the max of min limits for each joint is exceeded
13:     if $\boldsymbol{x}_{pred}(i) > \boldsymbol{Limit}_{Max}(i)$ then
14:         $\boldsymbol{x}_{pred}(i) = \boldsymbol{Limit}_{Max}(i)$
15:     else
16:         if $\boldsymbol{x}_{pred}(i) < \boldsymbol{Limit}_{Min}(i)$ then
17:             $\boldsymbol{x}_{pred}(i) = \boldsymbol{Limit}_{Min}(i)$
18:         end if
19:     end if
20: end for
21: $\boldsymbol{j} \leftarrow \boldsymbol{j}_{CSM} \cup \boldsymbol{x}_{pred}$
```

4.4.1.1 Evaluation of Human Prediction

For tuning and evaluation of the Kalman filter used in the human prediction, joint data from a motion sequence with the IPS manikin was used. The Kalman filter and prediction were implemented in MATLAB where the noise tuning could be performed using visualisation tools. The filter generates an estimation of the velocities and thereby enables a better prediction of the future joint values. Thereby the predicted joint value x_{pre} is compared with the joint value measured from the IPS manikin Y, and the joint value estimated in the filter x_{KF}. The comparison is done by calculating the differences $x_{KF} - x_{pre}$ and $Y - x_{pre}$ for each joint and for all data points in the motion sequence. The result is compared with a normal distribution generated from the result. Furthermore, the differences for the Translation joint in direction x, y, and z are shown in Fig. 4.14 as histograms, where the position is predicted 1 second into the future. Histograms of $x_{KF} - x_{pre}$ and $Y - x_{pre}$ are shown in Fig. 4.14(a) and Fig. 4.14(b) respectively. The red line is the generated normal distribution for each case. Moreover, the tuning of the filter results in a prediction with a low difference and standard deviation. The resulting noise levels after tuning is presented in Table 4.4.

The behaviour of the result in direction x, y and z over time are shown in Fig. 4.15,

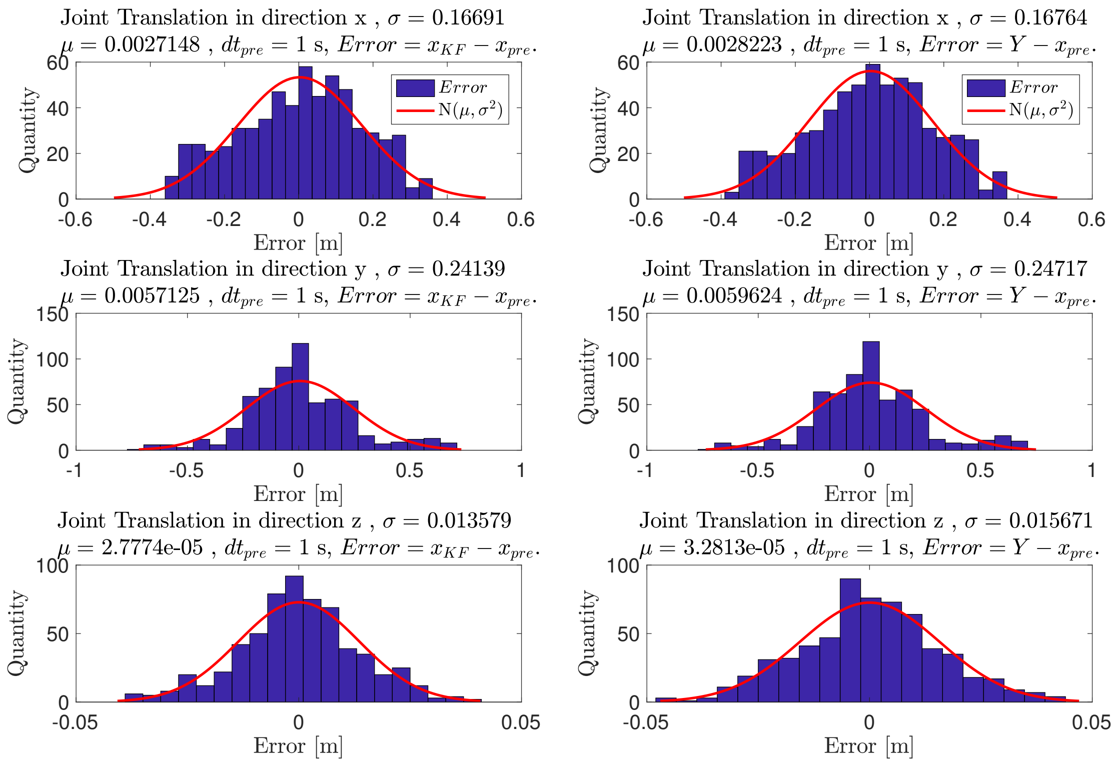


Figure 4.14: Differences between the predicted human positions and the Kalman filtered positions are shown in the subfigures to the left, whereas differences between the predicted human positions and measured positions are shown in the subfigures on the right. The red line in each subfigure is a normal distribution approximated from the errors. The prediction is 1 seconds ahead.

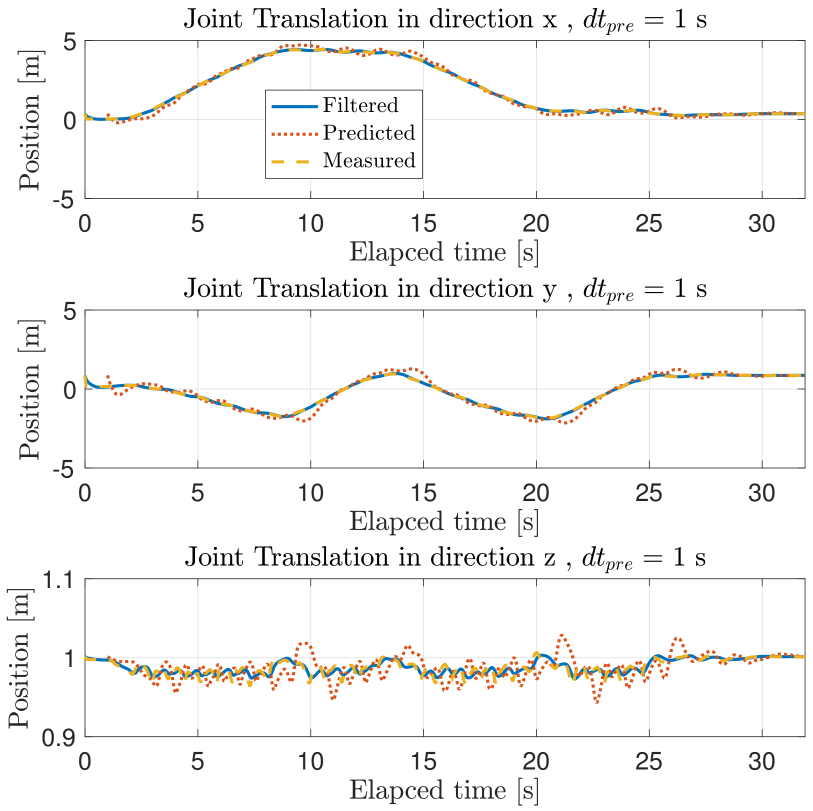


Figure 4.15: Comparison between measured, filtered and predicted positions of the human 1 seconds ahead. The positions in the world frame correspond to the "Translation" joint.

where the Kalman filter results x_{KF} are denoted "filtered", the prediction result x_{pre} are denoted "predicted", and the joint data from the IPS manikin Y are denoted "measured". In all directions, the prediction results in an overshoot if an abrupt change of the direction appears. A reason for this behaviour could be the constant velocity model in the filter.

In Table 4.5, the generated standard deviation is presented for different prediction times. The results show that increased prediction time results in decreased accuracy. Furthermore, it is assumed that the normal distributions are a good approximation for the differences. Standard deviation from the mean value, $\pm\sigma$ (which includes 68.27 % of the differences) shows that it reaches a value close to or above the limit (0.2 meters) used to detect collisions between the human and the robot in the x- and y-directions. Since the representation of the future is used to decide when a new path is needed, the human representation must be reliable and thereby predictions of the human movement longer than 1 second cannot be used. Furthermore, to change the manikin pose, a lot of computing capacity is required, especially when compared to the rest of the prediction algorithm. Therefore only the pose from prediction 1 second into the future is used to represent the manikin during the prediction. The results from predictions 2, 3, and 5 seconds for the arm pose are shown in the

Joint	**σ [m] at $\mathbf{dt_{pre} = 1}$ s**	**σ [m] at $\mathbf{dt_{pre} = 2}$ s**	**σ [m] at $\mathbf{dt_{pre} = 3}$ s**	**σ[m] at $\mathbf{dt_{pre} = 5}$ s**
$Translation_x$				
$Y - x_{pre}$	0.17	0.43	0.78	1.68
$x_{KF} - x_{pre}$	0.17	0.42	0.77	1.68
$Translation_y$				
$Y - x_{pre}$	0.25	0.63	1.17	2.51
$x_{KF} - x_{pre}$	0.24	0.62	1.16	2.51
$Translation_z$				
$Y - x_{pre}$	0.01	0.02	0.03	0.05
$x_{KF} - x_{pre}$	0.01	0.02	0.03	0.05

Table 4.5: The standard deviations of the differences between prediction x_{pre} and the joint values from the real time IPS-scene Y or the Kalman filter result x_{KF}, for different prediction times dt_{pre}.

appendix.

The noise tuning for the prediction of the arms was performed similar as the tuning of the Translation joint. The result against time, for 1 seconds prediction, is presented in Fig. 4.16, where it can be seen that the problem with overshoot is also apparent for these joints. The differences are shown in Fig. 4.17. Results for prediction times greater than 1 second is presented in the appendix.

4.4.2 Robot Prediction

After predicting the human movement and updating the pose of the manikin in the Prediction IPS-scene, it is time to calculate the distance between the manikin and the robot when in advance simulating that the robot is tracing its path. The current robot path is known in terms of waypoints, as explained in Chapter 3.2. Furthermore, since the waypoints of the path are known, it can be simulated in the IPS-scene dedicated for prediction. Here the robot is placed in joint configurations corresponding to waypoints and positions between the waypoints. The shortest distance from the robot to the manikin is then computed and used in the decision process together with the corresponding configuration.

4.4.2.1 Simulation of the Robot Path

While the robot is tracing the path (a sequence of waypoints), the distance between the robot and the manikin is computed for each of the robot configurations. To move the robot between the waypoints, each robot configuration needs to be calculated and sent to the IPS-scene. These configurations are calculated by an algorithm that generates the appropriate number of steps to represent the robot path for a

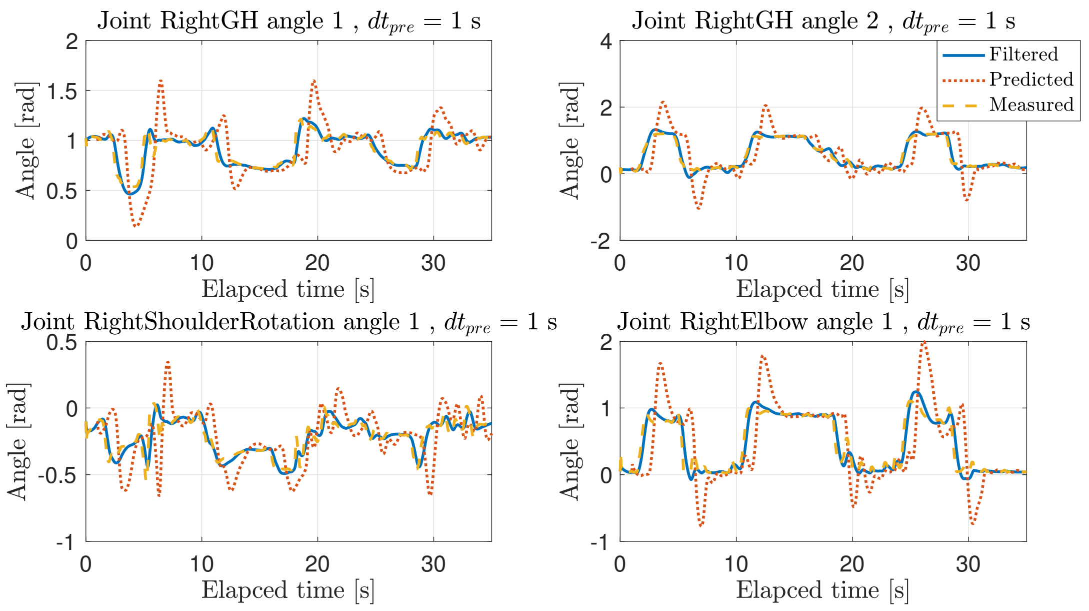


Figure 4.16: Prediction, measurements, and filtered data of the joints in the arm during a whole sequence. Prediction time dt_{pre} is set to $1s$.

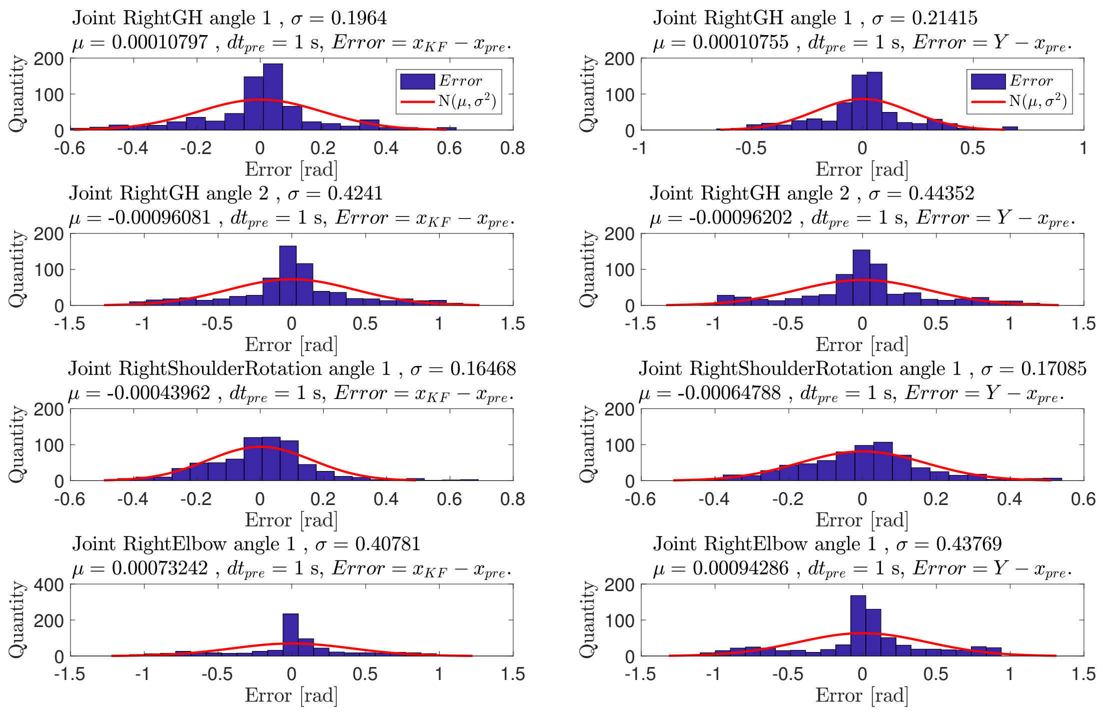
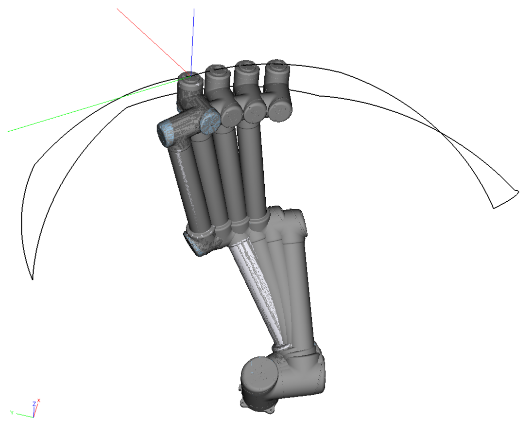
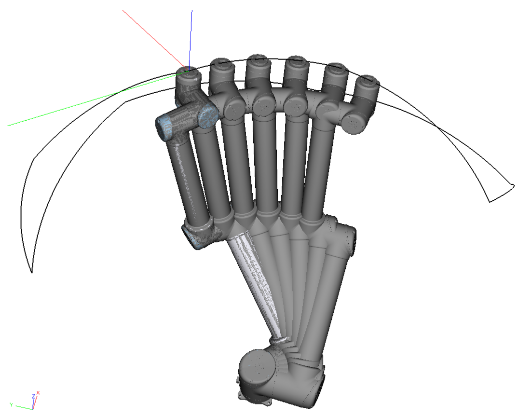
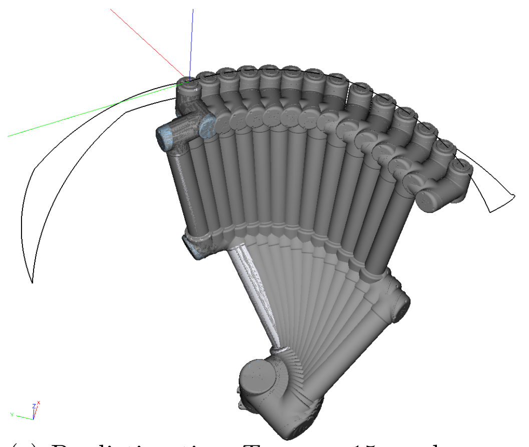


Figure 4.17: Differences between the predicted human arm angles and the Kalman filtered arm angles are shown in the subfigures to the left, whereas differences between the predicted human arm angles and measured arm angles are shown in the subfigures on the right. The red line in each subfigure is a normal distribution approximated from the errors. The prediction is 1 seconds ahead.

predicted time into the future. The number of steps used should be few, because the prediction IPS-scene has to be rendered for each robot configuration, which is time consuming. Some examples of predictions shown in Fig. 4.18 demonstrate how varying the distance between configurations and the predicted time affects the prediction.


(a) Prediction time T_{Robot} = 5s and r_{lim} = 20 cm.


(b) Prediction time T_{Robot} = 10s and r_{lim} = 20 cm.


(c) Prediction time T_{Robot} = 15s and r_{lim} = 10 cm.

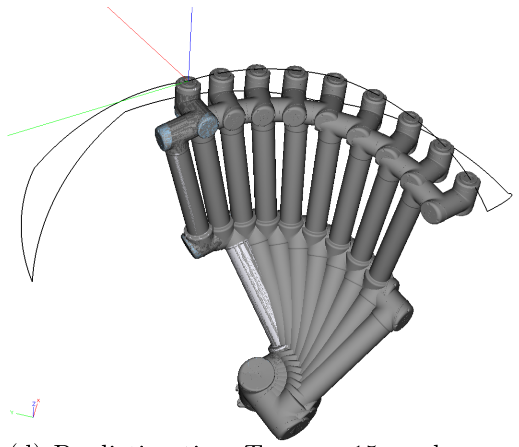

(d) Prediction time T_{Robot} = 15s and r_{lim} = 20 cm.

Figure 4.18: Predicted robot path for one iteration of the prediction, where r_{lim} is a limiting distance between two consecutive robot configurations, whereas T_{Robot} is the time the robot path represents.

4.4.2.2 Calculating the Robot Configurations

The robot configurations are calculated based on assumptions, inputs, and variables. These assumptions are stated below:

- The robot path is described as a sequence of waypoints.
- Between two consecutive waypoints, the robot moves linearly with respect to joint space and with constant velocity.

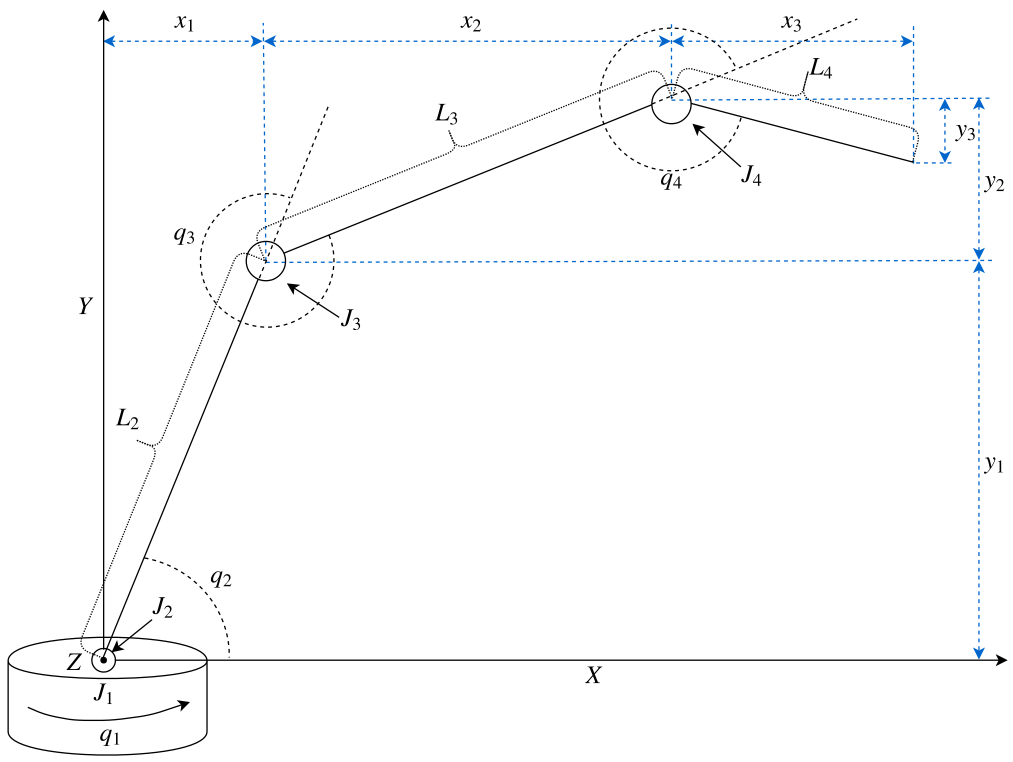


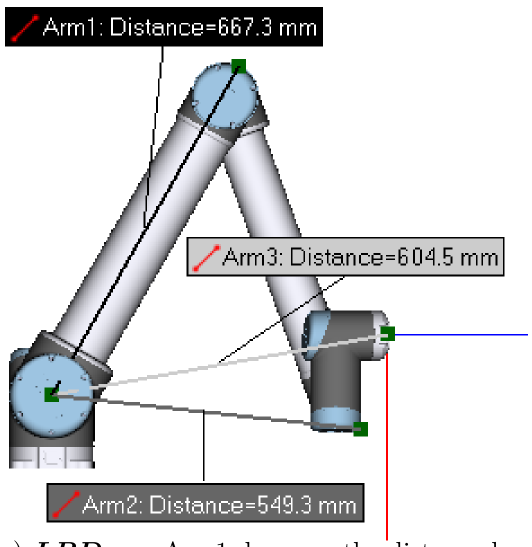
Figure 4.19: A simplified robot arm with four rotational joints.

- The joint speed is adjusted to make all joints reach the joint values in the next waypoint at the same time.
- The joint with the greatest difference between two waypoints (i.e., the joint that needs to rotate furthest) uses the maximum joint speed and is denoted the lead joint.
- The time between two consecutive waypoints is the time the lead joint needs to move between the waypoints using maximum allowed velocity. It is calculated and saved with the sequence of waypoints.

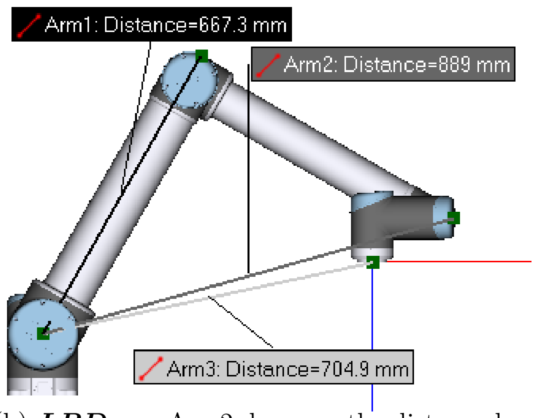
The robot prediction is a simulation of the robot tracing its path by moving to a sequens of calculated robot configurations based on two inputs: r_{lim} and T_{Robot}. The distance r_{lim} is the limiting distance between two consecutive configurations in the prediction, and T_{Robot} is the time that defines how much of the robot's future movement is simulated. Their initial values are $r_{lim} = 15$ cm and $T_{Robot} = 5$ seconds. How these two inputs impact the result is shown in Fig. 4.18.

A robot configuration consists of six joint values and is denoted by $\boldsymbol{q}$ whereas the current configuration of the robot is denoted by $\boldsymbol{q}_{current}$. Furthermore, a configuration step $\boldsymbol{q}_{step}$ is the difference between two consecutive configurations in the prediction (i.e., $\boldsymbol{q}_{step} = \boldsymbol{q}_j - \boldsymbol{q}_i$ where $\boldsymbol{q}_i$ is prior to $\boldsymbol{q}_j$).

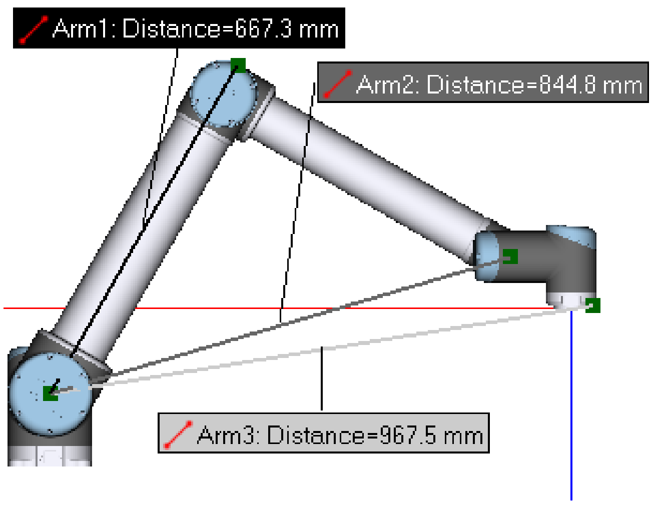
When the robot moves one joint from its q_i position to q_j the estimated robot move-



(a) $\boldsymbol{LPD}_{J_2}$ = Arm1, because the distance between J_2 and J_3 is the greatest.



(b) $\boldsymbol{LPD}_{J_2}$ = Arm2, because the distance between J_2 and J_4 is the greatest.



(c) $\boldsymbol{LPD}_{J_2}$ = Arm3, because the distance between J_2 and the TCP is the greatest.

Figure 4.20: A UR robot in three different configurations and the corresponding *LPD* for joint 2 (i.e., $\boldsymbol{LDP}_{J_2}$).

ment in the world frame is maximum r_{lim}. We calculate the Longest Perpendicular Distance (LPD) for the joints. The LPD_{Jk} is the perpendicular distance from the rotational axis of joint k to the part of the robot that is furthest away from that joint, and hence will move furthest if the joint is rotating. The part can either be another robot joint or the TCP depending on the current robot configuration. The LPD is recalculated for every configuration step, $\boldsymbol{q}_{step}$, for joint 1, 2, and 3 as shown in Fig.4.19. Joint 1 (J_1) rotates around the Y-axis, whereas the joint angels of the joints 2, 3, and 4 (J_2, J_3, J_4) are q_2, q_3, and q_4, respectively, where q_i is the angle of joint J_i and L_i is the length of link i. An example of different LPD for joint 2 is shown in Fig.4.20.

The LPD for the different joints for the robot shown in Fig. 4.19 are calculated as follows.

$$\boldsymbol{LPD}_{J_1} = \max\begin{pmatrix} |x_1|, \\ |x_1 + x_2|, \\ |x_1 + x_2 + x_3| \end{pmatrix}, \tag{4.2}$$

$$\boldsymbol{LPD}_{J_2} = \max\begin{pmatrix} \sqrt{(x_1 + x_2 + x_3)^2 + (y_1 + y_2 + y_3)^2}, \\ \sqrt{(x_1 + x_2)^2 + (y_1 + y_2)^2}, \\ L_2 \end{pmatrix}, \tag{4.3}$$

$$\boldsymbol{LPD}_{J_3} = \max\begin{pmatrix} \sqrt{(x_2 + x_3)^2 + (y_2 + y_3)^2}, \\ L_3 \end{pmatrix}, \tag{4.4}$$

$$\boldsymbol{LPD}_{J_4} = L_4. \tag{4.5}$$

Where the distances x and y are calculated as

$$x_1 = L_2 \cos(q_2), \qquad y_1 = L_2 \sin(q_2), \tag{4.6}$$

$$x_2 = L_3 \cos(q_2 + q_3), \qquad y_2 = L_3 \sin(q_2 + q_3), \tag{4.7}$$

$$x_3 = L_4 \cos(q_2 + q_3 + q_4), \qquad y_3 = L_4 \sin(q_2 + q_3 + q_4). \tag{4.8}$$

The notations are shown in Fig. 4.19 where frame spanned by axes X and Y follows the rotation of J_1. The $\boldsymbol{LPD}_{J_1}$ only considers the length of the robot in X direction whereas $\boldsymbol{LPD}_{J_2}$ is the maximum distance from J_2 to either J_3, J_4, or the TCP in the $X - Y$ plane. $\boldsymbol{LPD}_{J_3}$ varies with the length of L_3 and L_4 and the angle of q_3 and q_4, while $\boldsymbol{LPD}_{J_4}$ is the length of L_4.

The robot arm shown in 4.19 is a simplified version of the robot used during our experiments shown in Fig. 3.5. One simplification is to represent joint 5 and 6 with a extra long link 4. The calculation of $\boldsymbol{LPD}_{J_{1-4}}$ are carried out as described above but $L_4 = L_4 + L_5$ instead. The LPD for joint 5 is link 5, and for joint 6 LPD is the tool width (i.e., $\boldsymbol{LDP}_{J_6} = 0$ for our experiments). The reason to use constant values for $\boldsymbol{LDP}_{J_{4-6}}$ is is to reduce the number of calculations during the prediction and since the link length of joint 4, 5, and 6 are rather short.

The configuration step $\boldsymbol{q}_{step}$ is calculated based on three inputs, $\boldsymbol{q}_{current}$, $\boldsymbol{q}_{next}$, and r_{lim}. $\boldsymbol{q}_{next}$ can either be a configuration at a waypoint or the robot configuration at time T_{Robot}. The path between $\boldsymbol{q}_{current}$ and $\boldsymbol{q}_{next}$ has to be linear in joint space.

First, the LPD is calculated for the joints. Then, the effect each joint's rotation has on the robot's movement in the world frame is approximated. The greatest distance is approximated as the arc each joint create ($\boldsymbol{Arc}$) if only that joint is rotated. The distance $\boldsymbol{Arc}$ only considers the movement when rotating one joint and does not include the effect of multiple simultaneous joint rotations. It is calculated as

$$\boldsymbol{Arc}_{J_i} = \boldsymbol{LPD}_{J_i} |\boldsymbol{q}_{next}(i) - \boldsymbol{q}_{current}(i)| \tag{4.9}$$

where i corresponds to the joint number, for example, $|\boldsymbol{q}_{next}(1) - \boldsymbol{q}_{current}(1)|$ corresponds to the joint rotation in radians from $\boldsymbol{q}_{current}(1)$ to $\boldsymbol{q}_{next}(1)$ for the joint 1. These arcs, calculated for each joint, are divided by r_{lim} to calculate the number of prediction steps needed for the robot to reach $\boldsymbol{q}_{next}$ without violating the distance constraint r_{lim}. The joint requiring the most number of steps is limiting and the other joints are adjusted accordingly.

$$n_{steps} = \max\left(\frac{\boldsymbol{Arc}_{J_1}}{r_{lim}}, \frac{\boldsymbol{Arc}_{J_2}}{r_{lim}}, \frac{\boldsymbol{Arc}_{J_3}}{r_{lim}}, \frac{\boldsymbol{Arc}_{J_4}}{r_{lim}}, \frac{\boldsymbol{Arc}_{J_5}}{r_{lim}}, \frac{\boldsymbol{Arc}_{J_6}}{r_{lim}}\right). \tag{4.10}$$

The rotation for all joints can be divided in to n_{steps} number of steps since it is assumed that the motion is linear in joint space and all joints have a constant velocity. The last calculation to retrieve the configuration step vector $\boldsymbol{q}_{step}$ is therefore,

$$\boldsymbol{q}_{step} = \frac{\boldsymbol{q}_{next} - \boldsymbol{q}_{current}}{n_{steps}}. \tag{4.11}$$

The steps to calculate the predicted robot configurations are as follows. Firstly, it is observed if there are any changes in the linear movement during the next T_{Robot} seconds. Since the robot only changes speed or direction when passing a waypoint, this is done by checking if a waypoint will be passed. Secondly, a list of configurations is created. The list starts with the current configuration $\boldsymbol{q}_{current}$, and ends with the configuration T_{Robot} seconds into the future ($\boldsymbol{q}_{last}$). Thirdly, more configurations are added to the list to represent the robot path between $\boldsymbol{q}_{current}$ and $\boldsymbol{q}_{last}$. To this end, $\boldsymbol{q}_{step}$ is recalculated and added to $\boldsymbol{q}_{current}$ resulting in the first configuration in the prediction $\boldsymbol{q}_1$, which is saved in the list. Next, a new $\boldsymbol{q}_{step}$ is calculated and added to the previous prediction $\boldsymbol{q}_1$. This is repeated until the final configuration $\boldsymbol{q}_{last}$ is reached and the algorithm is done. If the predicted robot passes a waypoint, the configuration of the waypoint is used instead of the calculated configuration. The algorithm is described in Alg. 2.

Algorithm 2 Robot prediction

```
Procedure robot prediction
    Input: $\boldsymbol{q}_{current}$, $T_{Robot}$, $r_{lim}$
    Output: $\boldsymbol{Q}_{prediction}$    ▷ A list of multiple predicted robot configuration vectors.

$\boldsymbol{q}_{current} \leftarrow$ Current robot configuration
$T_{Next} \leftarrow$Time to next waypoint
$r_{lim} \leftarrow$Limiting distance
$\boldsymbol{q}_{last} \leftarrow$ Configuration at time $T_{Robot}$
$\boldsymbol{Q}_{prediction} \leftarrow$ Empty list

if $T_{Next} < T_{Robot}$ then    ▷ Control if any waypoints.
    $passing_{waypoint} \leftarrow True$
else
    $passing_{waypoint} \leftarrow False$
end if

$\boldsymbol{Q}_{prediction}.append(\boldsymbol{q}_{current})$

while $\boldsymbol{Q}_{prediction}.back() < \boldsymbol{q}_{last}$ do
        ▷ Calculate $\boldsymbol{q}_{step}$ with start from $\boldsymbol{Q}_{prediction}.back()$:
    Calculate $LPD$ for all joints.
    Calculate $Arc$ created from rotations of the joints.
    Choose the step length based on longest $Arc$ and $r_{lim}$.
    Calculate $\boldsymbol{q}_{step}$.
    $\boldsymbol{q}_{waypoint} \leftarrow$ configuration at waypoint

    if $passing_{waypoint}$ and ( $\boldsymbol{Q}_{prediction}.back() + \boldsymbol{q}_{step}) > \boldsymbol{q}_{waypoint}$ then
        $\boldsymbol{Q}_{prediction}.append(\boldsymbol{q}_{Waypoint})$
    else
        $\boldsymbol{Q}_{prediction}.append(\boldsymbol{Q}_{prediction}.back() + \boldsymbol{q}_{step})$
    end if
end while

$\boldsymbol{Q}_{prediction}.back() = \boldsymbol{q}_{last}$
Return $\boldsymbol{Q}_{prediction}$
```

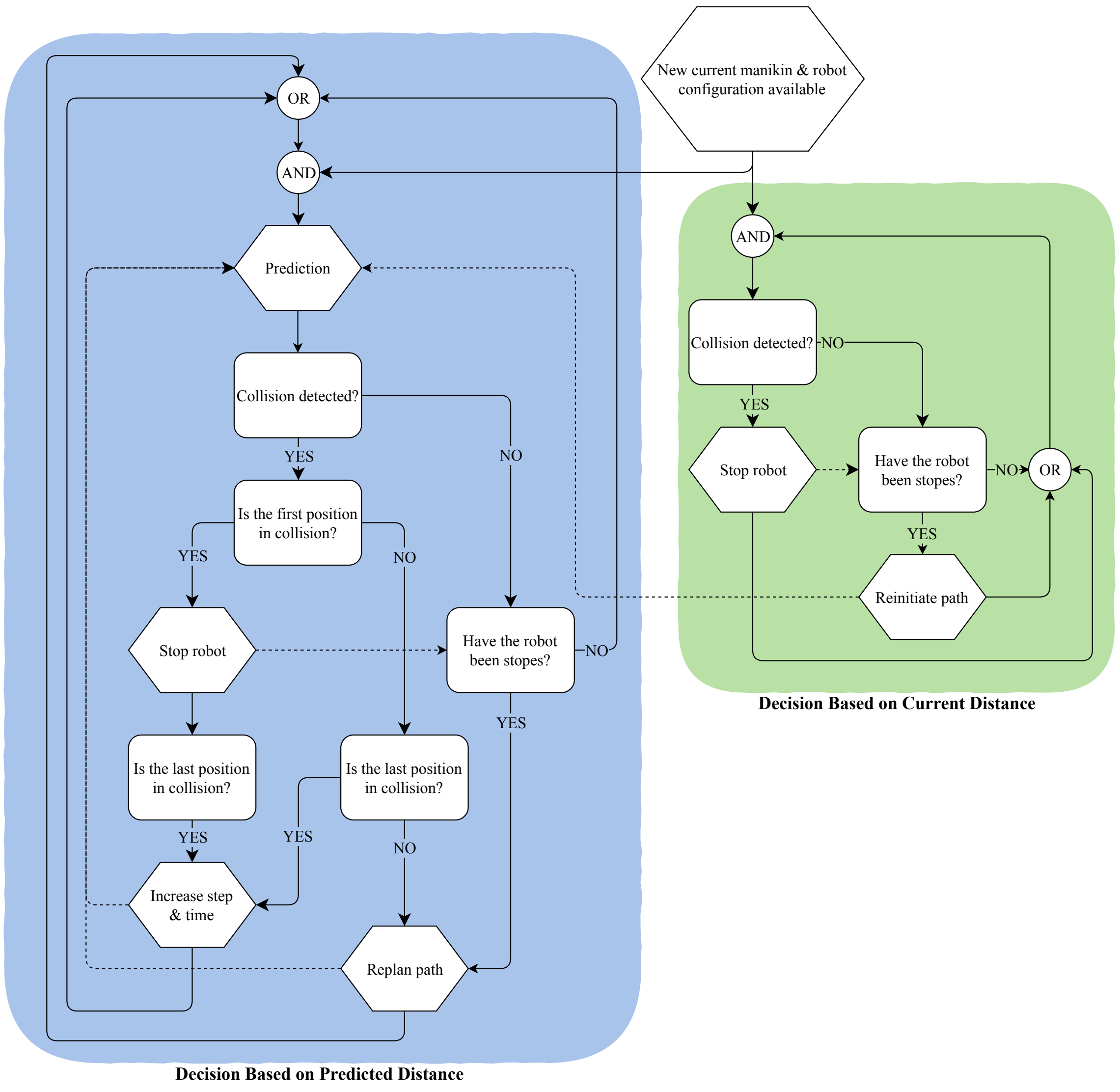

New current manikin & robot configuration available
OR
AND
Prediction
Collision detected?
YES
NO
Is the first position in collision?
YES
NO
Stop robot
Have the robot been stopes?
NO
YES
Is the last position in collision?
YES
Is the last position in collision?
YES
NO
Increase step & time
Replan path
Decision Based on Predicted Distance
AND
Collision detected?
NO
YES
Stop robot
Have the robot been stopes?
NO
OR
YES
Reinitiate path
Decision Based on Current Distance

Figure 4.21: The main controller's decision process. The solid lines corresponds to direction of process and the dotted lines corresponds to information. The reinitiate of a path, sends the previous followed path to the robot.

4.5 Main Controller

We used the controller program to predict future collisions. If a collision is imminent, the program investigates the collision to see if it can be isolated in a specific part of the path. Next, the re-planning algorithm is executed. The newly calculated path segment is then merged with the existing path and sent to the robot controller.

4.5.1 Decision Based on Distances

The built-in safety system stops the robot if too much torque is applied in the joints. We developed a controller to complement the safety system. This controller uses information from the real time IPS-scene. As shown in Fig. 4.21, the controller is divided into two control loops. The first loop measures the distance in the real time IPS-scene and stops the robot if the distance between the robot and the human is less than the threshold. The second loop contains a prediction step that calculates the distances between a set of predicted robot configurations and a prediction of the human's future position and pose and makes decisions accordingly. Consequently, the decision program has no information about the human's tasks nor their order. The program works best when the human is not expected to be close to the robot except on rare occasions.

Furthermore, after a set of instructions is sent to the robot, the instruction cannot be changed. To control the robot, we have identified the following options:

- Stop the robot by clearing the current instructions.
- Decelerate the robot by sending a position and an acceleration value corresponding to a deceleration. This instruction results in a linear deceleration in joint space based on the current motion.
- Change path by replacing the current instruction set with a new one.

4.5.1.1 Decision Based on Current Distance

The real time control loop measures the distance between the robot and the manikin in the real time IPS-scene by executing a LUA-script that collects the distance data and sends it back to the control algorithm as a JSON representation. A distance less than the threshold might lead to a collision. Therefore, we stop the robot by emptying the instructions in the robot controller. A deadlock situation can occur if the robot is stopped even though a new path is not available or the system has not started to calculate it yet. To avoid the deadlock, the robot will continue on its previous path if this happens.

4.5.1.2 Decision Based on Predicted Distances

The prediction algorithm is the first part of the decision algorithm. It returns a structure with information, such as the predicted manikin pose and the distances between the manikin and a set of future robot configurations, where the manikin pose is the result from a prediction 1 second into the future. On the other hand, the robot is represented with multiple configurations from its current configuration to a configuration T_{Robot} seconds ahead. The number of configurations depends on

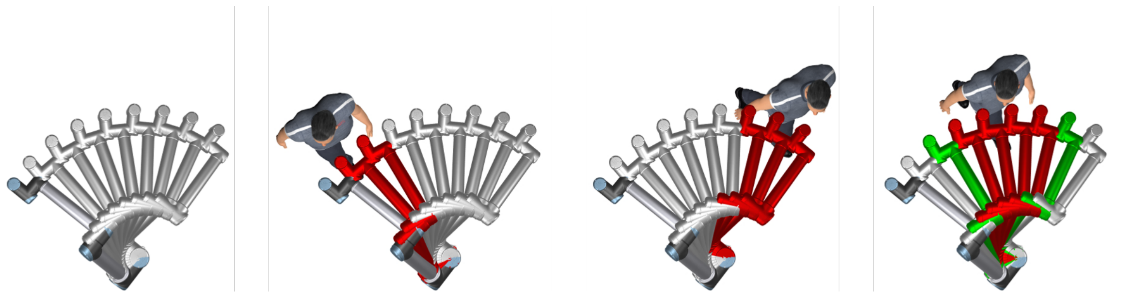


Figure 4.22: Different results from the prediction algorithm viewed from above. The robot starts its motion from the left. A red robot indicates that the configuration violates the minimum distance constraint. The green robots represents collision-free configurations.

the prediction time and the minimum allowed distance between two of the predicted robot representations. An increased distance results in fewer configurations to represent a part of the robot path, but it also decreases the accuracy of the representation.

Different scenarios can occur in the prediction step and Fig. 4.22 shows the four cases covered by the decision process. In the figure, the predicted robot configurations that are closer to the manikin then the threshold are marked as red. The three collision scenarios are a collision in the beginning, at the end, and as an isolated part of the path. For example, in the rightmost subfigure in Fig. 4.22, the collision is isolated and the collision-free configurations before and after are marked in green. The information in the different scenarios is used to decide when and how to plan a new path. To enable the calculation of a new path, we require that the collision must be isolated.

In the first case as shown in first subfigure in Fig. 4.22, all distances are greater than the minimum allowed distance. In other words, no collision is foreseen and hence actions is not required. This loop continues as long as no collision is detected.

In the second case as shown in second subfigure in Fig. 4.22, the first predicted step is in a collision. The robot is therefore stopped immediately by clearing all instructions in the robot controller. This is typically triggered if the human moves fast towards the robot and the prediction did not foreseen a collision in the previous iteration. The prediction loop continues to run with use of the last generated path, even though the robot is stopped and stays stationary as long as the collision in the first predicted step remain. However, if the calculation of a new path has not started when the distance has increased and is tolerated again, the controller is instructed to let the robot continue where it was interrupted in order to prevent a deadlock.

In the third case as shown in third subfigure in Fig. 4.22, the last prediction step is in collision. A collision-free configuration must be found before and after the collision to enable the calculation of a new path. Consequently, if none of the last predicted steps has a distance above the tolerance, the prediction has to be redone.

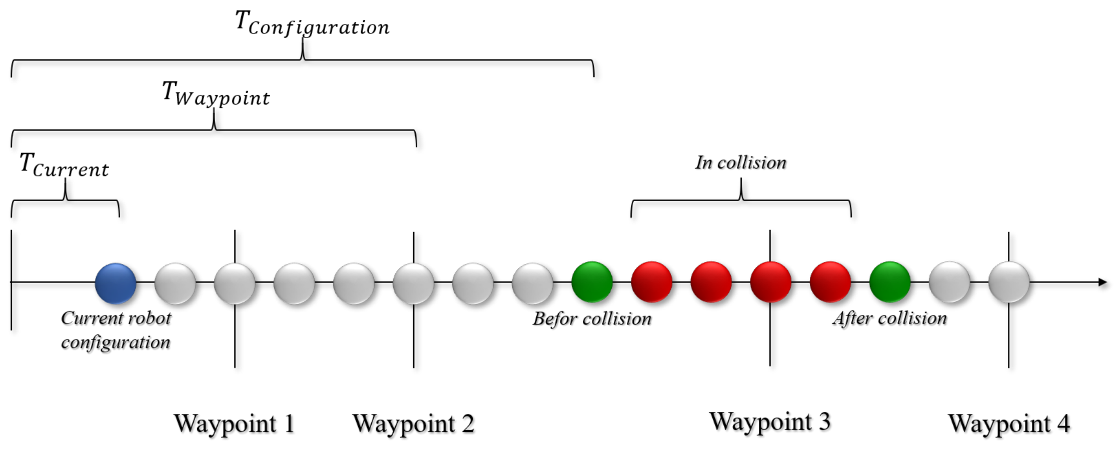


Figure 4.23: A prediction example of the robot path. The dots represent the predicted robot configurations with respect to time. The number of robot configurations between waypoints varies since the distance between waypoints is not fixed.

The next prediction is therefore modified to include more time. Sometimes the step length is increased to find a collision-free configuration.

In the fourth case as shown in fourth subfigure in Fig. 4.22, a section of the path is in a collision, but both the first predicted step and the last ones are collision-free. The result indicates an isolated collision and hence a new path can be calculated.

4.5.2 Re-planning Algorithm

The robot moves linearly in joint space between waypoints that correspond to robot configurations. When calculating a new path, it is preferable to plan from the closest waypoint before the collision than from a robot configuration closer to the manikin since the prediction is uncertain. The predicted time until the collision is used to decide between the two options.

The time used for decisions is always relative to when the robot started on the path because the behaviour of the controller is unknown and the robot might need a longer time than expected. Moreover, the joint speed is assumed to be constant between two consecutive waypoints and hence the time betweenthe waypoints is calculated as $Time = JointDistanse \times Speed$.

Furthermore, here are four different times used by the re-planning algorithm. They are based on the predicted duration which is calculated from the constant velocity assumption:

$T_{Current}$	time from robot start until now.
$T_{Configuration}$	time from robot start until the robot reaches the configuration before the collision.
$T_{Waypoint}$	time from robot start until the robot reaches the waypoint before the collision.
$T_{PathPlanning}$	the time it takes to perform a path planning.

An example that illustrates a predicted path and the usage of the different times is shown in Fig. 4.23. The re-planning algorithm is described in Algorithm 3.

Algorithm 3 Re-planning algorithm

```
 1: $Configuartion_{Current} \leftarrow$ The robot's current configuration
 2: $T_{Current} \leftarrow$ The robot's current time
 3: $Configuartion_{BC} \leftarrow$ The last collision-free configuration before the collision
 4: $T_{Configuartion} \leftarrow$ Time to last collision-free configuration before the collision
 5: $Waypoint_{BC} \leftarrow$ The last waypoint before the collision
 6: $T_{Waypoint} \leftarrow$ Time to last waypoint before the collision
 7: $T_{PathPlanning} \leftarrow$ Time to calculate new path
 8:
 9: if $T_{Configuration} \leq T_{Current}$ then
10:     Stop robot
11:     Re-plan from $Configuartion_{Current}$.
12: else
13:     if $(T_{Configuration} - T_{Current}) \leq T_{PathPlanning}$ then
14:         Decelerate the robot
15:         Re-plan from $Configuartion_{BC}$.
16:     else
17:         if $(T_{WayPoint} - T_{Current}) \leq T_{PathPlanning}$ then
18:             Re-plan from $Waypoint_{BC}$.
19:         else
20:             Re-plan from $Configuartion_{BC}$.
21:         end if
22:     end if
23: end if
```

The re-planning algorithm is activated to chose the start configuration for the new path and calculate it if a collision is detected and isolated. When choosing the start configuration, there are three alternatives to be considered:

- The current robot configuration denoted as $Configuartion_{Current}$.
- The last collision-free configuration before the collision denoted as $Configuartion_{BC}$.
- The last waypoint before the collision denoted as $Waypoint_{BC}$.

Since the robot has moved during the prediction, this algorithm observes the current robot configuration ($Configuartion_{Current}$) and the corresponding time ($T_{Current}$).

If the robot has passed both $Waypoint_{BC}$ and $Configuartion_{BC}$, the robot is immediately stopped and a new path calculated from the $Configuartion_{Current}$, because the collision is very close and the robot must be stopped to avoid it. However, the new path is planned from one of the other two alternatives if there is time left until the robot reaches at least one of them. Then the algorithm calculates the times until the robot reaches the two alternatives and compares it with the average time used during the planning of a new path ($T_{PathPlanning}$). With that information, the algorithm chooses between three solutions. The first solution is used if the time until $Configuartion_{BC}$ is reached is shorter than $T_{PathPlanning}$. Then a new path is planned from $Configuartion_{BC}$. Simultaneously, a deceleration is calculated to stop the robot at the $Configuartion_{BC}$ when $T_{PathPlanning}$ seconds has passed. The robot deceleration is both linear in joint space and adjusted to the fastest moving joint. Furthermore, if $T_{PathPlanning}$ is longer than the time to reach $Waypoint_{BC}$ and shorter than the time to reach $Configuration_{BC}$, we choose the second solution, which calculates the new path from $Configuration_{BC}$ without decelerating the robot. Finally, if $T_{PathPlanning}$ is longest, we choose the third solution, which calculates the new path from the $Waypoint_{BC}$ without decelerating the robot.

4.5.2.1 Path Handling

A new path is either calculated from a waypoint or a given configuration as further described in Chapter 4.3. The run-time of the path planning algorithm depends on the complexity of the scene. It tasks longer time if the human is close or if the passage is narrow, but the time is approximated to 0.5 seconds as shown in Section 4.3.3.

Furthermore, we use different manikin representations during the decision and path-planning process. In the decision process, the real time IPS-scene is used and the distance between the robot and the manikin is measured against a collider representation of the manikin. While in the path planning process, the manikin is represented by a visual mesh. Therefore, the decision algorithm is programmed with a conservative cutoff distance, 0.5 meters, which is longer than the minimum distance allowed by the algorithm.

4.5.2.2 Merge Paths

When a robot path is sent to the robot controller, the path is represented by a list of joint configurations. If a new path replaces part of the existing path, they are merged. Hence, the old instructions are saved during the path planning process. Moreover, the robot is often instructed to keep moving during the path planning. The first instruction of the newly merged path is therefore often the next waypoint or a new waypoint on that path. Since the robot moves linearly between configurations, smooth transitions and predictable robot behaviour can be ensured as long as the first control instruction contains the configuration the robot is heading for at that

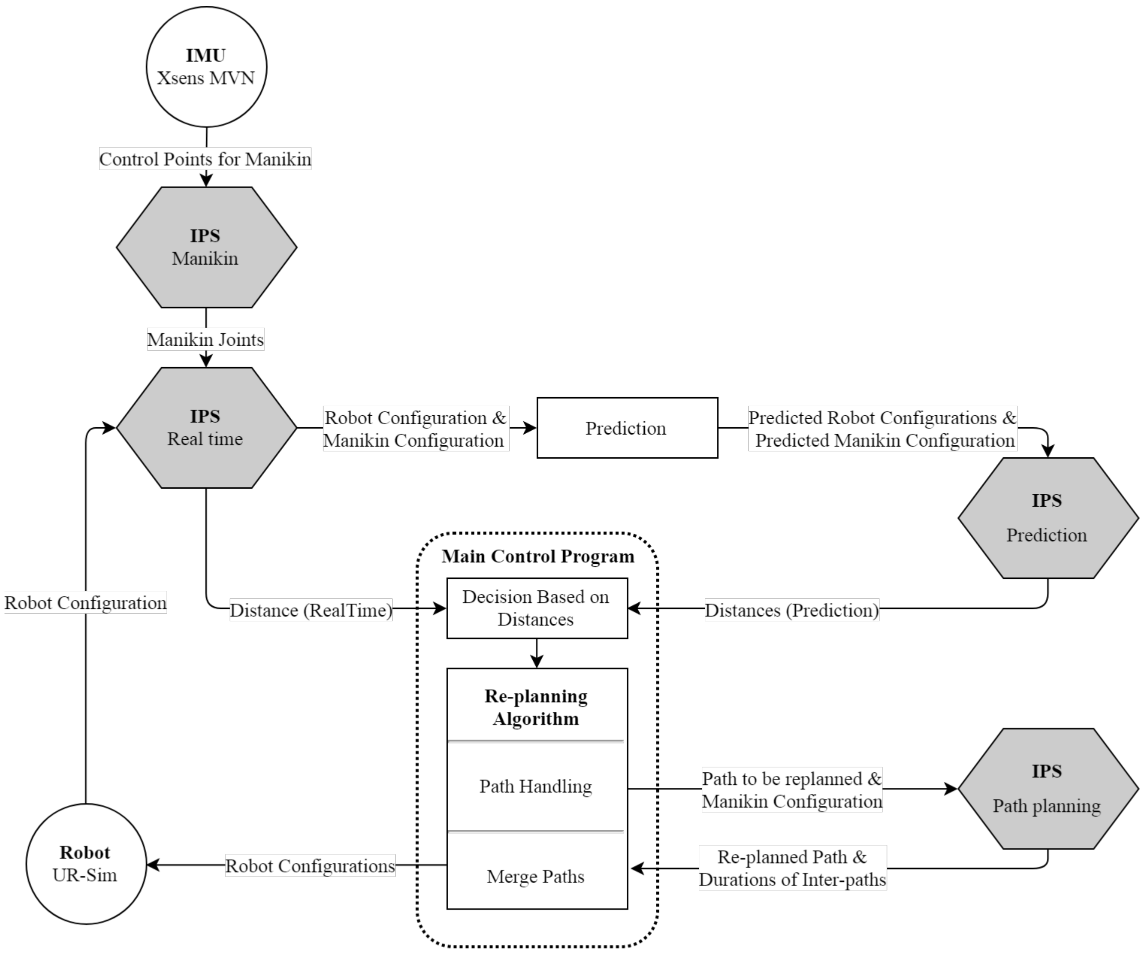

IMU
Xsens MVN
Control Points for Manikin
IPS
Manikin
Manikin Joints
IPS
Real time
Robot Configuration & Manikin Configuration
Prediction
Predicted Robot Configurations & Predicted Manikin Configuration
IPS
Prediction
Main Control Program
Robot Configuration
Distance (RealTime)
Decision Based on Distances
Distances (Prediction)
Re-planning Algorithm
Path Handling
Merge Paths
Path to be replanned & Manikin Configuration
IPS
Path planning
Re-planned Path & Durations of Inter-paths
Robot
UR-Sim
Robot Configurations

Figure 4.24: The communication network between the IPS-scenes and the decision algorithm.

moment.

4.6 Summary

We summarise the concept in Fig. 4.24. The human is tracked and visualised with the robot in an IPS-scene. From this scene, positions are used in a prediction algorithm to calculate distances between the robot and the manikin and predict future collisions. Decisions regarding re-planning is taken based on the distances collected from both the real time scene and the prediction scene.

5 Use Case and Evaluation

To evaluate the developed framework and decision process, a worker interrupts a robot, that moves along a nominal path, and thereby triggers the different functionalities of the system. An overview of the test scene in IPS is shown in Fig. 5.1 where both the manikin, which is linked to the human, and the robot, which is linked to the URsim, are visulized. The nominal path is generated from a list of nine waypoints which the robot moves in between in a circular loop (i.e., it start and stop at the same waypoint) as shown in Fig. 5.2 and Fig. 5.3. In addition to the visual path, a graph of the robot's joint configurations during the nominal path are shown in Fig. 5.4.

The tests were performed with a real human and a simulated robot. During the tests, the human can see the virtual representation of the environment on a screen in front of him as shown in Fig. 5.5. At the same time, the robot follows its nominal path while the human moves in the environment to trigger the system to take different decisions such as replanning the robot path.

The testes were performed under the following circumstances:

- The real human was a 1.8 meters tall male.
- The tests were performed in a non-magnetic environment, aside from the screens showing the virtual environment and robot simulation.
- The robot joint velocities were limited to 0.25 rad/s, which is further described in Section 4.3.2.
- The safety distance between the human and the robot was 0.4 meters. If the human is within that distance the robot will stop.
- The minimum distance between robot and the human in the prediction scene was 0.2 meters.
- It took approximately 0.5 seconds to generate a new path. Further discussion can be found in Section 4.3.3.
- The Kalman filter was tuned as described in Section 4.4.1.
- The initial prediction time for the robot was 5 seconds and for the human 1 second.

The results of the tests are shown in graphs similar to Fig. 5.4 where the different joint configurations of the robot are shown. In addition, the distances between

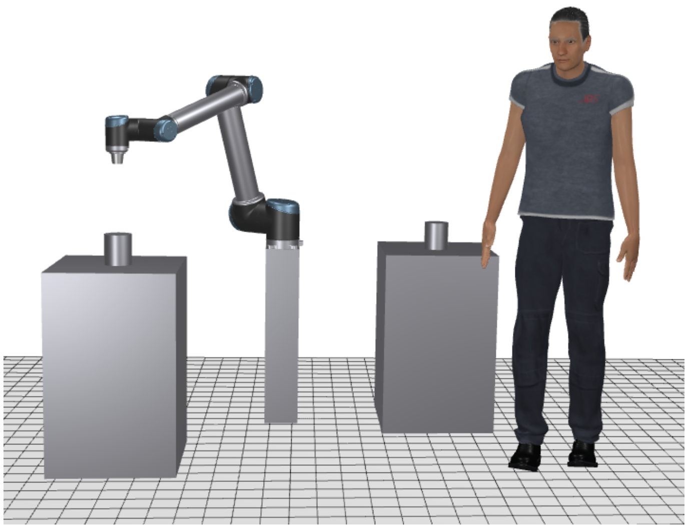


Figure 5.1: Test scene with one robot and one manikin.

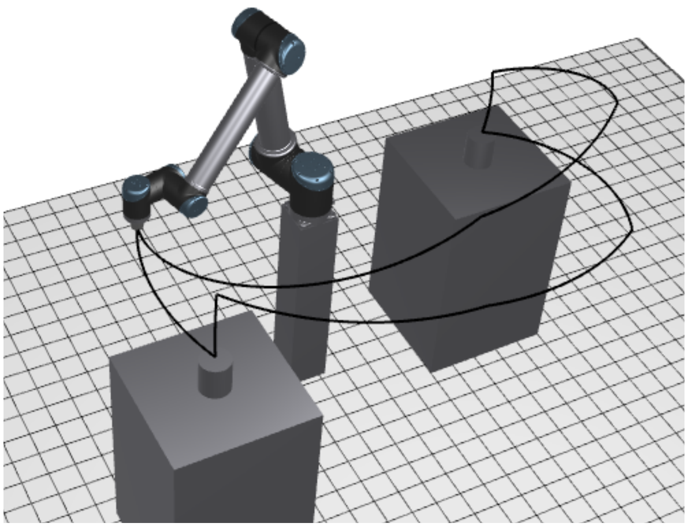


Figure 5.2: The TCP trace.

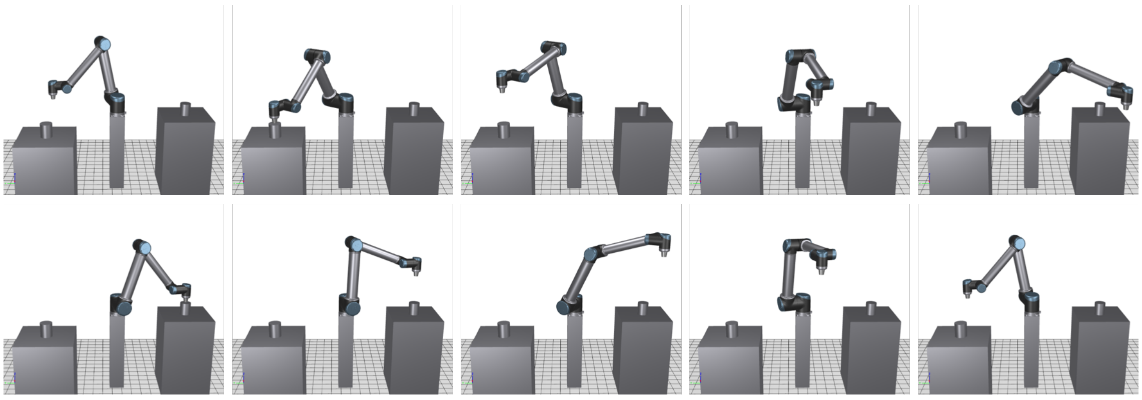


Figure 5.3: The robot positioned at all the nine waypoints.

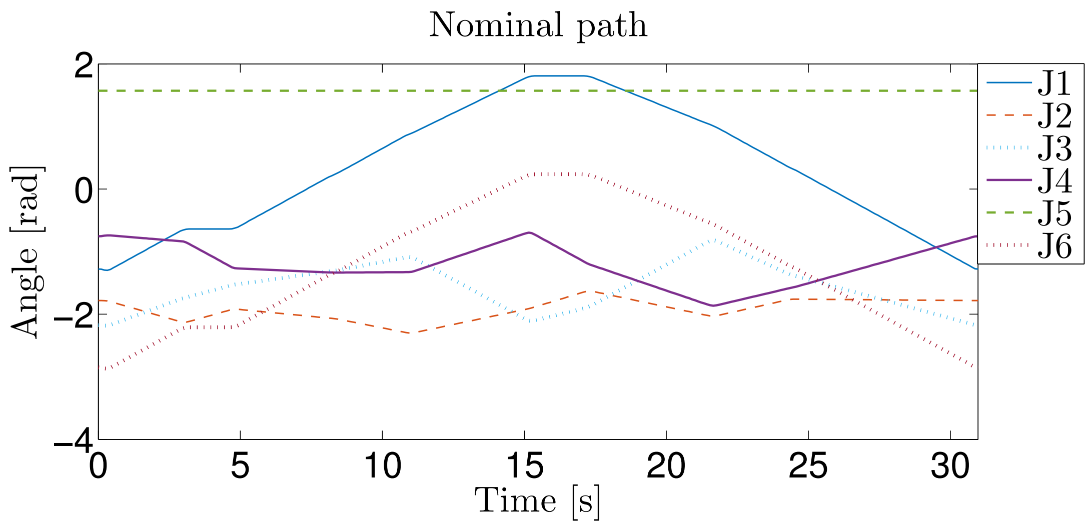


Figure 5.4: The joint angles of the robot arm during execution of the nominal robot path.

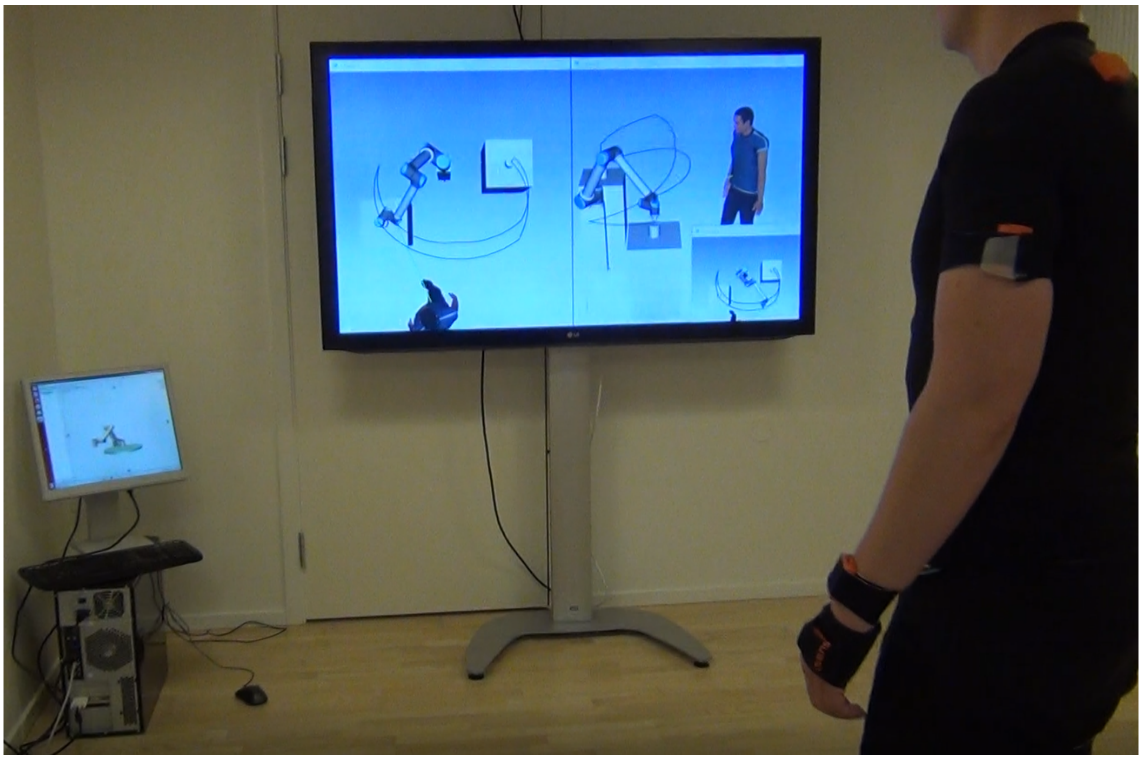


Figure 5.5: The simulated robot is shown on the smaller screen. The human can see both his digital twin (i.e., a manikin) in front of him and the virtual test scene.

the manikin and the robot measured in the real-time IPS scene are shown in the same figure, where the measured distances are plotted as points, and each point corresponds to one measurement used in the the decision algorithm. Decisions are then constantly taken as the robot moves along its path. The decisions effecting the behaviour of the robot are marked as black vertical lines in the graphs showing the results. Additional markings in the figures are small plus signs that correspond to the joint configurations in “Re-plan from specific joint configuration”, where each plus sign indicates a joint configuration from which the new path is calculated and thereby is the first instruction in the new path sent to the robot controller. On the time axis, the plus signs correspond to when the new path is calculated and sent to the robot controller.

We logged also the critical decisions during the test. The decision process log contains the following possible decisions:

Last state in collision
A collision is detected during prediction. However, no collision-free configuration after the collision is detected and hence the prediction time is increased and the robot continues along its current path.

Stop: First state in collision
A collision is predicted just ahead of the robot’s current position and the robot is therefore stopped.

Re-plan from waypoint
A collision is detected and the path planner is likely to be able to calculate a new path before the robot reaches the upcoming waypoint while avoiding the collision. A re-planning from the waypoint is therefore performed.

Re-plan from specific joint configuration
A collision is detected but the robot will probably pass the upcoming waypoint before a new path is calculated. The new path is therefore calculated from the last robot joint configuration before the collision.

Decelerate: Re-plan from specific joint configuration
A new path is not likely calculated before the robot is in collision with the human. The robot is therefore decelerated while a new path is calculated from the last collision free configuration.

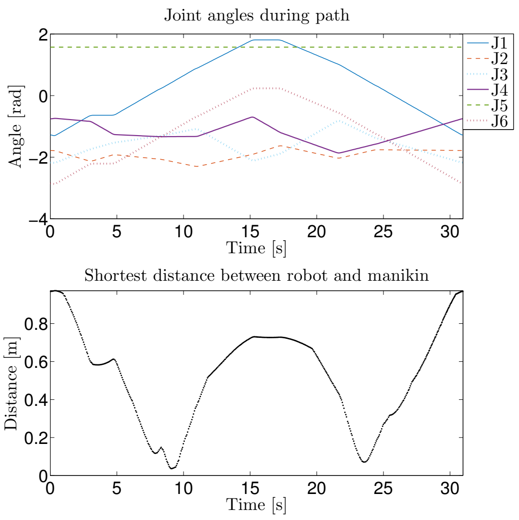
Time for path planning
The time (in seconds) it took to calculate a new path.

Script sent to robot
A new path is calculated and sent to the robot for execution.

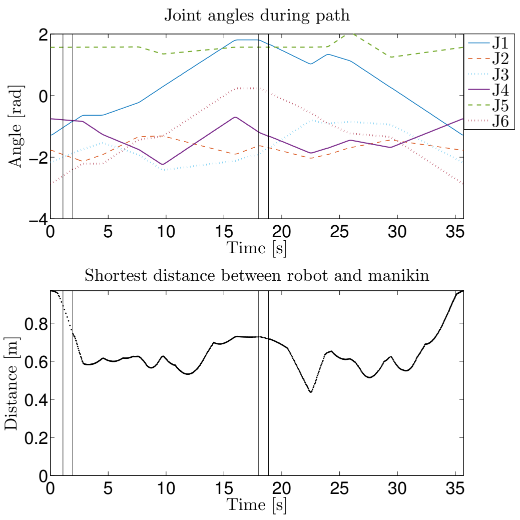
Stop: Robot stopped during Path-Planning
A parallel thread is started at the same time as the path planner to measure the distance between the robot and the manikin in the real-time IPS scene. If the distance is below the threshold, the robot is stopped and this message is received.

Continue on the current path
The robot has been stopped but the path is now clear and the robot can continue to trace its previous path.

5.1 Test 1



(a) The robot does not adjust its path to avoid the human.



(b) The robot adjust its path to avoids the human.

Figure 5.6: Test 1. Joint angles and distance to the human with and without the re-planning algorithm.

Time instance [s]	Event
0.78	Last state in collision
1.09	Re-plan from waypoint
1.92	Time for path planning: 0.77
1.94	Script sent to robot
17.67	Last state in collision
18.00	Re-plan from waypoint
18.84	Time for path planning: 0.82
18.85	Script sent to robot

Table 5.1: Test 1. Event log of decisions made during test shown in Fig. 5.6, where the bold numbers correspond to vertical black lines.

In the first test the manikin was positioned in a fixed pose, as shown in Fig. 5.6, whereas the robot was tested both with and without the re-planning algorithm to test the basic functionality of the algorithm. The results show that the algorithm predicts the collision and replans the path before the robot gets too close to the human. Furthermore, the system changes the path almost unnoticeably for the human user since it does not slow down but simply continues on the new path when it is calculated, because the collision is detected well in advanced and hence the new path can be generated quickly (less than a second). Consequently, the robot receives the new path before it reaches the waypoint from which the new path is calculated from.

5.2 Test 2

During the further tests, the human is linked to the manikin so that it can move in the virtual environment. As shown in Fig. 5.7, the distance between the human and the robot decreases rapidly and a collision is predicted which triggers a path re-planning. When the re-planning starts, the distance between the human and the robot is almost one meter. However, the re-planning takes 7.11 seconds due to the narrow space and the short distance between the human and the robot. Consequently, the robot comes to close to the human and the system stops the robot which has to wait for the path planner to finish. Due of this stop, the new robot path is also discarded and another path is calculated as soon as the human has moved away from the robot. During the same time period, the distance between the human and the robot is not measured until the path planner is finished due to the serial implementation of the main control program. Moreover, a similar behaviour can be noticed around the time 40 seconds, where a new path is generated almost directly after the robot is stopped because the human has moved away.

Another noticeable thing can be seen around 30 seconds into the test, where the robot stops and later continues its path. As shown in the lower figure of Fig. 5.7, the human is approaching the robot fast and forces it to stop due to the proximity. After the human moves away, the robot continues moving along its previous path.

5.3 Test 3

As shown in Fig. 5.8, path planning in the third test takes less than a half second, but it is difficulty to isolate the collision. This can be seen in the log at 3.02 seconds, where a collision is detected but no configuration after the collision is found. The robot is later stopped at 4.80 seconds due to short distance in the real-time scene.

Furthermore, the deceleration decision is used at 9.28 seconds. The robot velocity is decreased to allow the path planner more time. The robot is though stopped just after the new path is calculated but continues on the newly calculated path as the human moves away. Note that the robot continues to move along the new path as long as the distance is greater than the minimum threshold.

5.4 Test 4

In the last test, the human moves a lot and tries to get close to the robot multiple times resulting in re-planning of the path as shown in Fig. 5.9. Multiple path plannings are performed but the system keeps running and perform predictions and calculate paths. However, poor prediction performance was registered since a new

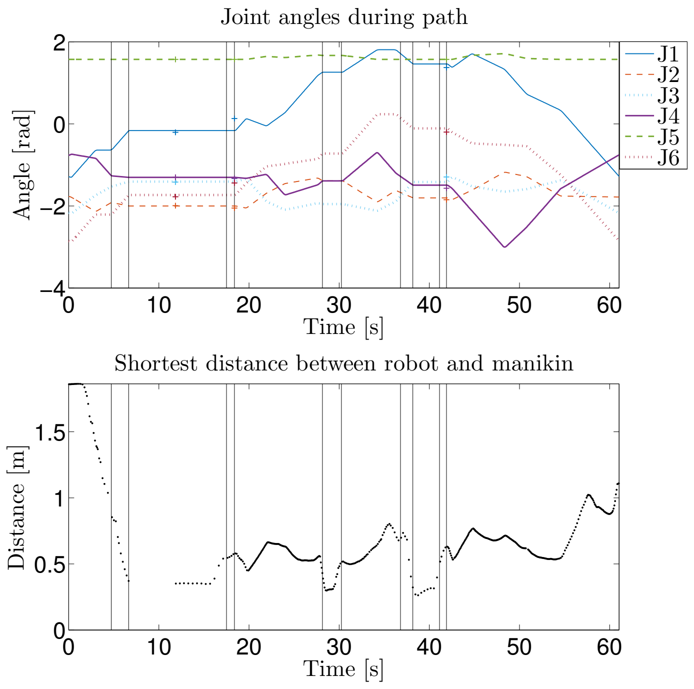


Figure 5.7: Test 2. Robot joint angles, and distances between the human and the robot during the test.

Time instance	Event
3.45	Last state in collision
4.72	Re-plan from specific joint configuration
6.65	Stop: Robot stopped during Path-Planning
11.84	Time for path planning: 7.12
17.50	Re-plan from specific joint configuration
18.37	Time for path planning: 0.73
18.38	Sent script to robot
28.14	Stop: First state in collision
30.25	Start: Continue on the current path
36.25	Last state in collision
36.80	Re-plan from waypoint
38.17	Stop: Robot stopped during Path-Planning
38.33	Time for path planning: 1.53
41.12	Re-plan from specific joint configuration
41.88	Time for path planning: 0.61
41.90	Sent script to robot

Table 5.2: Test 2. Event log from the test shown in Fig. 5.7, where the bold numbers in the table correspond to the black vertical lines in the figure.

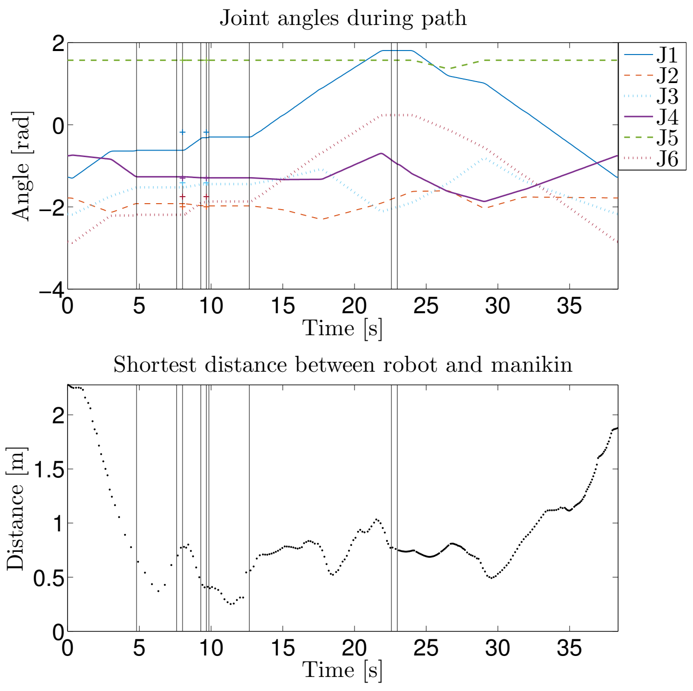


Figure 5.8: Test 3. Robot joint angles, and distances between the human and the robot during the test.

Time instance	Event
3.02	Last state in collision
4.80	Stop: First state in collision
7.60	Re-plan from specific joint configuration
7.99	Time for path planning: 0.32
8.01	Sent script to robot
9.28	Decelerate: Re-plan from specific joint configuration
9.65	Time for path planning: 0.29
9.67	Sent script to robot
9.84	Stop: First state in collision
12.67	Start: Continue on the current path
22.57	Re-plan from waypoint
22.97	Time for path planning: 0.34
22.98	Sent script to robot

Table 5.3: Test 3. Event log from the test shown in Fig. 5.8, where the bold numbers in the table correspond to the black vertical lines in the figure.

Time instance	Event
2.82	Last state in collision
3.31	Re-plan from waypoint
5.34	Time for path planning: 1.91
5.37	Sent script to robot
6.43	Last state in collision
7.69	Re-plan from specific joint configuration
8.63	Stop: Robot stopped during Path-Planning
10.37	Time for path planning: 2.68
17.68	Re-plan from specific joint configuration
19.15	Time for path planning: 1.32
19.17	Sent script to robot
19.36	Decelerate: Re-plan from specific joint configuration
20.54	Time for path planning: 1.07
20.56	Sent script to robot
20.75	Re-plan from specific joint configuration
21.34	Time for path planning: 0.38
21.36	Sent script to robot
35.80	Stop: First state in collision
38.62	Re-plan from waypoint
39.05	Time for path planning: 0.36
39.06	Sent script to robot
42.48	Last state in collision
44.66	Current state in collision. Stopped from prediction
49.81	Re-plan from specific joint configuration
50.43	Time for path planning: 0.41
50.44	Sent script to robot
52.66	Decelerate: Re-plan from specific joint configuration
52.95	Time for path planning: 0.26
52.96	Sent script to robot
53.14	Stop: First state in collision
53.30	Re-plan from specific joint configuration
53.68	Time for path planning: 0.31
53.69	Sent script to robot
62.29	Last state in collision
63.03	Re-plan from specific joint configuration
63.46	Time for path planning: 0.36
63.47	Sent script to robot

Table 5.4: Test 4. Event log from the test shown in Fig. 5.9, where the bold numbers in the table correspond to the black vertical lines in the figure.

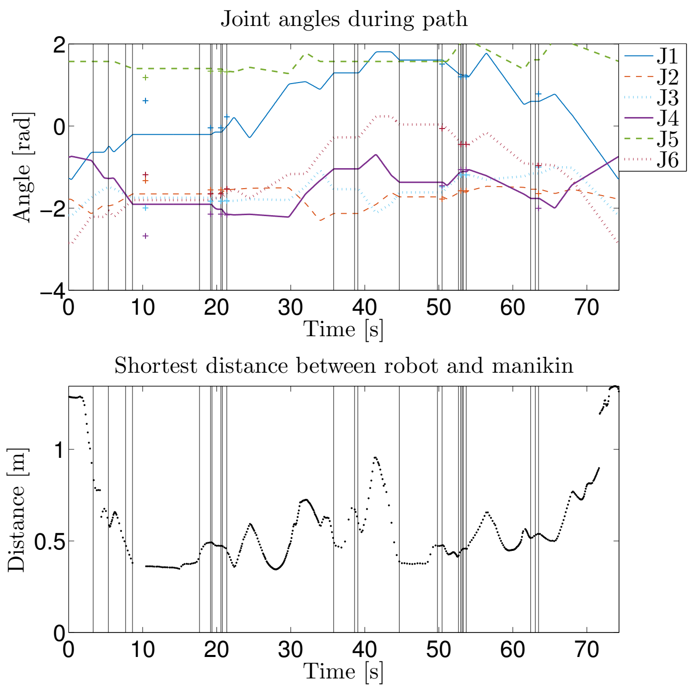


Figure 5.9: Test 4. Robot joint angles, and distances between the human and the robot during the test.

path is often required shortly after a new path has been calculated, indicating a better prediction of the human's movement is required.

6 Discussion

6.1 Tracking and Human Representation

Our tracking system has several limitations. Firstly, the tracking system requires physically measurements of the human worker and the IMUs need to be calibrated before a recording session. Additionally, the sensors might also be in the way for the worker, and hence prevent him/her from performing the tasks in a correct way. This can, however, be solved using other types of sensors such as cameras.

Secondly, the tracking system can only track one human worker at moment, which is in contrast to assembly lines where multiple humans may work in the same station and additional equipments may be present. These objects can be tracked by depth cameras or IMUs attached to tools, where each tool has a unique QR-codes or ID-numbers. The identification can then be used to match the objects with predefined objects in the virtual environment.

Thirdly, the manikin representation of the human is solely based on the skeleton dimensions of a human and does not take into consideration the soft tissues. Consequently, the decision algorithm uses a manikin representation that is not totally accurate to the real human, and hence it cannot always maintain a minimum separation distance between the real human and the robotic arm.

6.2 Prediction

Both the human prediction algorithm and the robot prediction algorithm have their limitations. The human prediction algorithm creates wagging movement of the predicted human when the human walks due to the linear prediction model used to predict the position of the translation-joint located within the body of the manikin. In addition, the same linear prediction model limits the prediction time since it results in low accuracy when predicting multiple seconds ahead. A more advanced prediction model that considers the tasks that are likely for the human to perform, or learning from the previous motions, would give a better prediction of the human worker's longer-term motion.

Moreover, the current robot prediction algorithm assumes that the robot joints never require excessive speeds, since it is based on a constant velocity model. However, if the robot works in higher speed, a better approximation of the path duration is required. Consequently, a model generating a trajectory between the waypoints has to handle not only the velocity limit but also the acceleration limit.

6.3 Path Planing

The IPS path planner we used only identifies a feasible path and does not adjust the path according to human social behaviour and hence the resulting path could be unsuitable for the human. The path planning algorithm could be complemented with for example human zones to improve the human-robot interaction.

Furthermore, since the path planning is performed in the same thread as the main program, the main program is blocked and no further actions can be taken if the path planner gets stuck when calculating a difficult path. One solution is to use multiple threads where one path planning thread is aborted if it fails to generate a path within a number of seconds. During the tests of the system, the IPS path planner needed between 0.3 and 7.0 seconds to generate a path which indicates a need for multithreading.

6.4 Decisions

The decisions regarding re-planning are made based on only distances, which could result in re-planning of the path even though the human is only passing by. By using both velocity and direction, it is possible to determine whether the human is only passing by, approaching, or leaving. It would thereby be possible choose a more appropriate behaviour of the robot such as slowing down or entering a waiting mode.

7 Conclusions and Future Work

We presented in this thesis a framework to predict and prevent collisions between a human and a robotic arm. Furthermore, this framework cannot only stop the robotic arm but also modify the robot trajectory in real-time by utilizing the path planning module in IPS.

By comparing between different manikin models, we have demonstrated that the pose of the IPS manikin is consistent with the one in Xsens MVN, even though the corresponding body segments are not linked. Moreover, we were able to obtain a good prediction of the future positions of the human (1 second ahead) by utilizing a Kalman filter. This prediction is essential since it is the main input to the decision process that has to make decissions in real-time regarding whether the robot path has to be re-planned.

Another important part of the work is the control and path planning of the robot. A new path is generated by only re-planning smaller parts of the path and then merging the re-planned path with the original path. After testing the control and path planning, the results indicate that it is faster to re-plan from existing waypoints of the path than from a new robot joint configuration. Moreover, the path planning algorithm is efficient since it can generate new paths in between 0.37 to 1.57 seconds.

Finally, the prediction and decision algorithm was mainly developed for human standing relatively still. For such test cases, we were able to generate new path in real-time and hence update path almost unnoticeable.

7.1 Future Work

There are many avenues for future work. Firstly, we would like to improve the prediction of the human motion. The linear model used gives a wagging prediction that is only accurate for shorter time periods in advance. A more advanced model that can learn from motions and/or take behaviours and tasks into consideration could be used instead.

Secondly, we would like to improve the robot instructions. The robot path is now only re-planed to avoid a collision. However, the same re-planning framework could also be used to change the robot path by, for example, changing the goal position of the robot. This can be done by choosing a specific TCP position in the room and plan a path to this position. An implementation like this enables a more collaborative behaviour since the robot could be programmed to hand over tools based on the location of the human hand or be moved to a position in the room to assist the human performing a task. Moreover, a small jerk in the robot movement is noticeable when the robot controller receives new instructions and thereby has to terminate the current motion in order to execute the new instructions. To avoid this jerk, a better way of sending robot instructions has to be implemented so that the robot can either slow down or adjust the current trajectory.

Thirdly, the communication could be improved to speed up the framework. Currently, all the instructions from the main program to the IPS scenes are sent by LUA commands at a speed of around 50Hz. This speed decreases if many commands have to be exchanged since many JSON strings have to be encoded/decoded. Furthermore, the IPS scenes also affect the communication speed because they have to be rendered and the visualization has to be updated in order to retrieve information such as distance measurements. Taken together, these issues could slow down the communication speed to less than 10 Hz. Therefore, we would like to speed up the communication by using communication that does not require encoding and making IPS non-dependent om visualization.

Fourthly, the decision process could be improved by smoothing the behaviour of the robot. If a collision is detected but no collision-free robot position after the collision is found, the robot keeps following its nominal path. The robot is then stopped based on information from the real-time loop that indicates that the robot is too close to the human. The robot remains motionless until the human moves away, even though the robot could have been stopped earlier and a new path could have been calculated to avoid the human. Similarly, we would like to keep the robot following its nominal path as long as possible. This behaviour results in re-planning from a configuration close to the human. If the human then moves closer to the robot, the robot is stopped due to insufficient clearance and remains motionless until the human moves away. The results from the tests of the path planner also indicate that the path planning takes longer time if the robot is placed near the human than in a configuration further away. It is, therefore, a prioritized improvement to change the behaviour to stop the robot when it is further away. Furthermore, the decision program could be improved by dividing it into multiple threads. Currently, only one thread is used for both prediction and path planning. Therefore, these tasks are performed sequentially. Although they are handled in two different IPS instances which imply that it would be possible to continues the predictions while the path planner is calculating a new path. By dividing the prediction and the path planning, it would be possible for the robot to change behaviour while a new path is being calculated.

Finally, we would like to address problems related to the sensors. The drift problem could be solved using a camera to update the position of the human. Since it is not necessary to get an updated position from the camera at every time instance the human does not have to be in the line of sight all the time.

Bibliography

[1] Daniel Althoff, Dirk Wollherr, and Martin Buss. Safety assessment of trajectories for navigation in uncertain and dynamic environments. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 5407–5412, May 2011.

[2] Maren Bennewitz, Wolfram Burgard, Grzegorz Cielniak, and Sebastian Thrun. Learning motion patterns of people for compliant robot motion. *The International Journal of Robotics Research*, 24(1):31–48, 2005.

[3] Robert Bohlin, Niclas Delfs, Lars Hanson, Dan Högberg, and Johan S Carlson. Automatic creation of virtual manikin motions maximizing comfort in manual assembly processes. In *Proceedings of 4th CIRP Conference on Assembly Technologies and Systems, May 20-22, 2012, Ann Arbor, Michigan, USA*, pages 209–212. Conference on Assembly Technologies & Systems (CIRP), 2012.

[4] Robert Bohlin and Lydia E Kavraki. Path planning using lazy prm. In *Proceedings of 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 1, pages 521–528. IEEE, April 2000.

[5] Luca Caltagirone. Human interaction solutions for intuitive motion generation of a virtual manikin. M.s thesis, Chalmers University of Thechnology, Gothenburg, Sweden, 2014.

[6] Andrea Cherubini, Robin Passama, André Crosnier, Antoine Lasnier, and Philippe Fraisse. Collaborative manufacturing with physical human–robot interaction. *Robotics and Computer-Integrated Manufacturing*, 40:1–13, 2016.

[7] J. C. K. Chow. Drift-free indoor navigation using simultaneous localization and mapping of the ambient heterogeneous magnetic field. *ISPRS - International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, XLII-2/W7:339–344, 2017.

[8] Shu-Yun Chung and Han-Pang Huang. Incremental learning of human social behaviors with feature-based spatial effects. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2417–2422.

IEEE, Oct 2012.

[9] J. A. Corrales, F. A. Candelas, and F. Torres. Safe human–robot interaction based on dynamic sphere-swept line bounding volumes. *Robotics and Computer-Integrated Manufacturing*, 27(1):177 – 185, 2011.

[10] J. A. Corrales, G. J. Garcia Gomez, F. Torres, and Véronique Perdereau. Cooperative tasks between humans and robots in industrial environments. *International Journal of Advanced Robotic Systems*, 9(3):94, 2012.

[11] W. H. K. De Vries, H. E. J. Veeger, C. T. M. Baten, and F. C. T. Van Der Helm. Magnetic distortion in motion labs, implications for validating inertial magnetic sensors. *Gait & posture*, 29(4):535–541, 2009.

[12] Wenwen Ding, Kai Liu, Fei Cheng, and Jin Zhang. Learning hierarchical spatio-temporal pattern for human activity prediction. *Journal of Visual Communication and Image Representation*, 35:103–111, 2016.

[13] Dirk Ebert, Takashi Komuro, Akio Namiki, and Masatoshi Ishikawa. Safe human-robot-coexistence: emergency-stop using a high-speed vision-chip. In *Proceedings of 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2923–2928. IEEE, Aug 2005.

[14] Amalia F Foka and Panos E Trahanias. Predictive autonomous robot navigation. In *Proceedings of 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 490–495. IEEE, Sept 2002.

[15] Consuelo Granata and Philippe Bidaud. A framework for the design of person following behaviors for social mobile robots. In *Proceedings of 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4652–4659. IEEE, Oct 2012.

[16] Marc Hanheide, Annika Peters, and Nicola Bellotto. Analysis of human-robot spatial behaviour applying a qualitative trajectory calculus. In *Proceedings of 2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*, pages 689–694. IEEE, Sept 2012.

[17] Soren Tranberg Hansen, Mikael Svenstrup, Hans Jorgen Andersen, and Thomas Bak. Adaptive human aware navigation based on motion pattern analysis. In *Proceedings of RO-MAN 2009 - The 18th IEEE International Symposium on Robot and Human Interactive Communication*, pages 927–932. IEEE, Sept 2009.

[18] Lars Hanson, Dan Högberg, Johan S Carlson, Robert Bohlin, Erik Brolin, Niclas Delfs, Peter Mårdberg, Gustafsson Stefan, Ali Keyvani, and Ida-Märta Rhen. Imma–intelligently moving manikins in automotive applications. In *Proceedings of Third International Summit on Human Simulation (ISHS2014)*, 2014.

[19] Jochen Heinzmann and Alexander Zelinsky. Quantitative safety guarantees

for physical human-robot interaction. *The International Journal of Robotics Research*, 22(7-8):479–504, 2003.

[20] Frank Hoeller, Dirk Schulz, Mark Moors, and Frank E Schneider. Accompanying persons with a mobile robot using motion prediction and probabilistic roadmaps. In *Proceedings of 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1260–1265. IEEE, Oct 2007.

[21] Justin Horowitz, Tejas Madhavan, Christine Massie, and James Patton. Reaching is better when you get what you want: Realtime feedback of intended reaching trajectory despite an unstable environment. *Frontiers in behavioral neuroscience*, 9:365, 2016.

[22] Chen-Yu Hsu et al. *Capturing the human figure through a wall.* PhD thesis, Massachusetts Institute of Technology, 2017.

[23] Yugo Katsuki, Yuji Yamakawa, and Masatoshi Ishikawa. High-speed human/robot hand interaction system. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction Extended Abstracts*, pages 117–118. ACM, 2015.

[24] Torsten Kröger and Friedrich M Wahl. Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events. *IEEE Transactions on Robotics*, 26(1):94–111, 2010.

[25] Jörg Krüger, Terje K Lien, and Alexander Verl. Cooperation of human and machines in assembly lines. *CIRP Annals-Manufacturing Technology*, 58(2):628–646, 2009.

[26] Jörg Krüger, Bertram Nickolay, P. Heyer, and Günther Seliger. Image based 3d surveillance for flexible man-robot-cooperation. *CIRP Annals-Manufacturing Technology*, 54(1):19–22, 2005.

[27] Thibault Kruse, Amit Kumar Pandey, Rachid Alami, and Alexandra Kirsch. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 61(12):1726–1743, 2013.

[28] Aleksandr Kushleyev and Maxim Likhachev. Time-bounded lattice for efficient planning in dynamic environments. In *Proceedings of 2009 IEEE International Conference on Robotics and Automation*, pages 1662–1668. IEEE, May 2009.

[29] Heiner Lasi, Peter Fettke, Hans-Georg Kemper, Thomas Feld, and Michael Hoffmann. Industry 4.0. *Business & Information Systems Engineering*, 6(4):239–242, 2014.

[30] Steven M Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, Computer Science Dept. Iowa State University, 1998.

[31] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[32] Kang Li and Yun Fu. Prediction of human activity by discovering temporal sequence patterns. *IEEE transactions on pattern analysis and machine intelligence*, 36(8):1644–1657, 2014.

[33] Hongyi Liu and Lihui Wang. Gesture recognition for human-robot collaboration: A review. *International Journal of Industrial Ergonomics*, 2017.

[34] Jim Mainprice, Rafi Hayne, and Dmitry Berenson. Goal set inverse optimal control and iterative replanning for predicting human reaching motions in shared workspaces. *IEEE Transactions on Robotics*, 32(4):897–908, 2016.

[35] Andreas Mark, Robert Bohlin, Daniel Segerdahl, Fredrik Edelvik, and Johan S Carlson. Optimisation of robotised sealing stations in paint shops by process simulation and automatic path planning. *International Journal of Manufacturing Research 5*, 9(1):4–26, 2014.

[36] Edgar A Martinez-Garcia, Ohya Akihisa, et al. Crowding and guiding groups of humans by teams of mobile robots. In *Proceedings of 2005 IEEE Workshop on Advanced Robotics and its Social Impacts*, pages 91–96. IEEE, June 2005.

[37] Giovanni Mirabella. Should i stay or should i go? conceptual underpinnings of goal-directed actions. *Frontiers in systems neuroscience*, 8:206, 2014.

[38] M. G. Mohanan and Ambuja Salgoankar. A survey of robotic motion planning in dynamic environments. *Robotics and Autonomous Systems*, 100:171 – 185, 2018.

[39] P. Mårdberg, Y. Yan, R. Bohlin, N. Delfs, S. Gustafsson, and J. S. Carlson. Controller hierarchies for efficient virtual ergonomic assessments of manual assembly sequences. *Procedia CIRP*, 44(Supplement C):435 – 440, 2016. 6th CIRP Conference on Assembly Technologies and Systems (CATS).

[40] Elisa Negri, Luca Fumagalli, and Marco Macchi. A review of the roles of digital twin in CPS-based production systems. *Procedia Manufacturing*, 11:939–948, 2017. 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy.

[41] Stefania Pellegrinelli, Andrea Orlandini, Nicola Pedrocchi, Alessandro Umbrico, and Tullio Tolio. Motion planning and scheduling for human and industrial-robot collaboration. *CIRP Annals*, 66(1):1–4, 2017.

[42] Agnieszka Radziwon, Arne Bilberg, Marcel Bogers, and Erik Skov Madsen. The smart factory: exploring adaptive and flexible manufacturing solutions. *Procedia Engineering*, 69:1184–1190, 2014.

[43] Universal Robots. Safety FAQ - 17750 kernel description. `https:`

//www.universal-robots.com/how-tos-and-faqs/faq/ur-faq/safety-faq-17750. Accessed: 2017-07-02.

[44] Universal Robots. *User Manual UR10/CB3, Original instructions (en).* Universal Robots.

[45] Daniel Roetenberg, Henk Luinge, and Per Slycke. Xsens MVN: full 6DOF human motion tracking using miniature inertial sensors. *Xsens Motion Technologies BV, Tech. Rep*, 1, 2009.

[46] Subhendu Roy, Sraboni Ghosh, Aratrika Barat, Madhurima Chattopadhyay, and Debjyoti Chowdhury. Real-time implementation of electromyography for hand gesture detection using micro accelerometer. In *Proceedings of Artificial Intelligence and Evolutionary Computations in Engineering Systems*, pages 357–364. Springer, 2016.

[47] Simo Särkkä. *Bayesian filtering and smoothing*, volume 3. Cambridge University Press, 2013.

[48] Satoru Satake, Takayuki Kanda, Dylan F Glas, Michita Imai, Hiroshi Ishiguro, and Norihiro Hagita. How to approach humans?: strategies for social robots to initiate interaction. In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction (HRI)*, pages 109–116. ACM, 2009.

[49] Johan Segeborn, Daniel Segerdahl, Fredrik Ekstedt, Johan S Carlson, Mikael Andersson, Anders Carlsson, and Rikard Söderberg. An industrially validated method for weld load balancing in multi station sheet metal assembly lines. *Journal of Manufacturing Science and Engineering*, 136(1):011002, 2014.

[50] D. Spensieri, J. S. Carlson, F. Ekstedt, and R. Bohlin. An iterative approach for collision free routing and scheduling in multirobot stations. *IEEE Transactions on Automation Science and Engineering*, 13(2):950–962, April 2016.

[51] Jesus Suarez and Robin R Murphy. Hand gesture recognition with depth images: A review. In *Proceedings of 2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*, pages 411–417. IEEE, Sept 2012.

[52] Satoshi Tadokoro, Masaki Hayashi, Yasuhiro Manabe, Yoshihiro Nakami, and Toshi Takamori. On motion planning of mobile robots which coexist and cooperate with human. In *Proceedings of 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, volume 2, pages 518–523. IEEE, Aug 1995.

[53] Xsens Technologies. Xsens MVN product: MVN Awinda. https://www.xsens.com/products/xsens-mvn-analyze/, 2017. Accessed: 2018-02-12.

[54] Simon Thompson, Takehiro Horiuchi, and Satoshi Kagami. A probabilistic

model of human motion and navigation intent for mobile robot path planning. In *Proceedings of 2009 4th International Conference on Autonomous Robots and Agents*, pages 663–668. IEEE, Feb 2009.

[55] Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics (ToG)*, 33(5):169, 2014.

[56] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1653–1660, June.

[57] Peter Trautman and Andreas Krause. Unfreezing the robot: Navigation in dense, interacting crowds. In *Proceedings of 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 797–803. IEEE, Oct 2010.

[58] Lihui Wang, Mohammad Givehchi, Göran Adamson, and Magnus Holm. A sensor-driven 3d model-based approach to remote real-time monitoring. *CIRP Annals-Manufacturing Technology*, 60(1):493–496, 2011.

[59] Lihui Wang, Bernard Schmidt, and Andrew Y. C. Nee. Vision-guided active collision avoidance for human-robot collaborations. *Manufacturing Letters*, 1(1):5–8, 2013.

[60] Xi Vincent Wang, Zsolt Kemény, József Váncza, and Lihui Wang. Human–robot collaborative assembly in cyber-physical production: Classification framework and implementation. *CIRP Annals*, 66(1):5 – 8, 2017.

[61] Sida Yang, Wenjun Xu, Zhihao Liu, Zude Zhou, and Duc Truong Pham. Multi-source vision perception for human-robot collaboration in manufacturing. In *Proceedings of 2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)*, pages 1–6. IEEE, March 2018.

[62] Brian D Ziebart, Nathan Ratliff, Garratt Gallagher, Christoph Mertz, Kevin Peterson, J Andrew Bagnell, Martial Hebert, Anind K Dey, and Siddhartha Srinivasa. Planning-based prediction for pedestrians. In *Proceedings of 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3931–3936. IEEE, Oct 2009.

A

Appendix 1

A.1 Result for Prediction: Translation Joint

Evaluation and tuning of the Kalman filter used for state estimation and prediction were done with recorded sequences of the manikin moving. The recorded joints are the Translation joint that describe the humans position in the world frame and the joints describing the position of the human arms. To enable prediction using constant velocity models the state velocities were estimated from measurements of the state position using a Kalman filter. Here results with increased prediction time to 2, 3, or 5 seconds that complement the result presented in section 4.4.1.1 are shown. In the figures there are some categories of data;

- Position measurements of the human worker and his/her body limbs, these are denoted "Measured" or "Y".
- Kalman filtered states of the human worker and his/her body limbs, these are denoted "Filtered" or "$variable_{KF}$".
- Estimated position of the human worker and his/her body limbs, these are denoted "Predicted" or "$variable_{pre}$".
- Prediction time denoted dt_{pre}.
- Mean value in normal distribution generated from the data denoted μ.
- Standard deviation in normal distribution generated from the data denoted σ.

The histograms show the error of the prediction compared to the measured values or the filtered states. From that data, a normal distribution is generated that is shown as a red line. The result is also displayed over time where, for example, an overshoot that is similar to the prediction time is easy to identify.

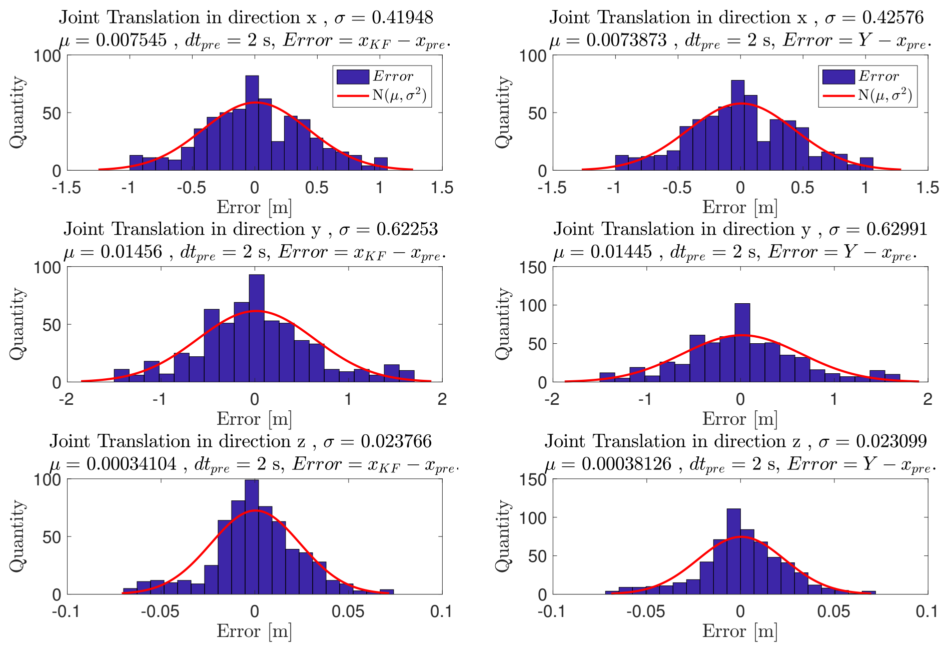


Figure A.1: Differences between the predicted human positions and the Kalman filtered positions are shown in the subfigures to the left, whereas differences between the predicted human positions and measured positions are shown in the subfigures on the right. The red line in each subfigure is a normal distribution approximated from the errors. The prediction is 2 seconds ahead.

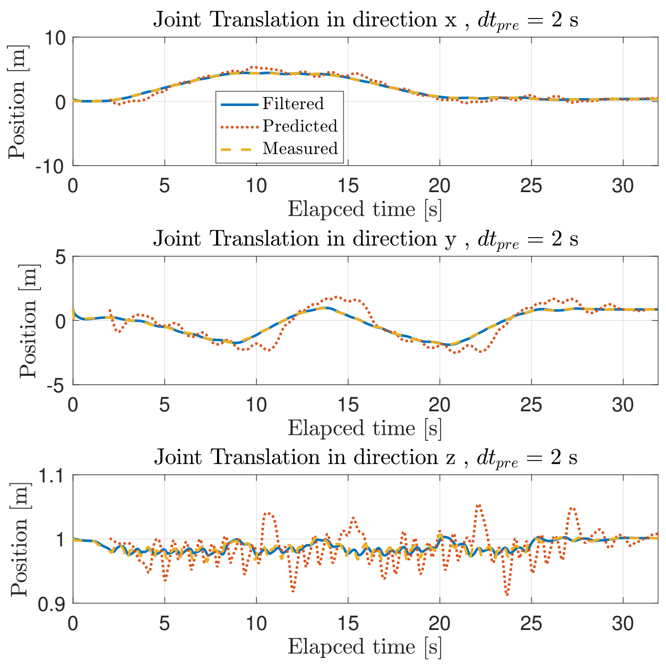


Figure A.2: Comparison between measured, filtered and predicted positions of the human predicted 2 seconds ahead. The positions in the world frame correspond to the "Translation" joint.

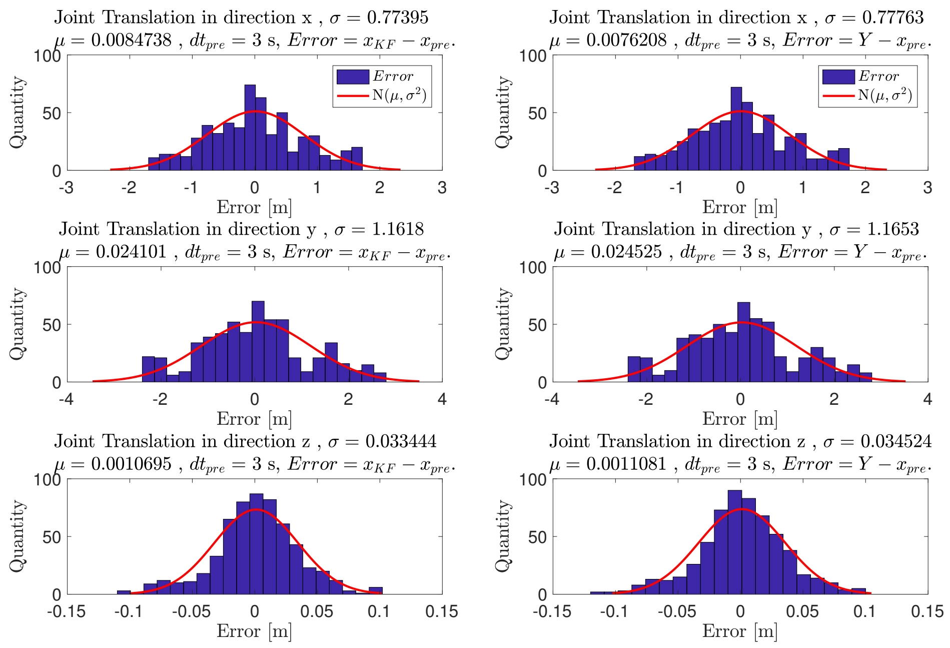


Figure A.3: Differences between the predicted human positions and the Kalman filtered positions are shown in the subfigures to the left, whereas differences between the predicted human positions and measured positions are shown in the subfigures on the right. The red line in each subfigure is a normal distribution approximated from the errors. The prediction is 3 seconds ahead.

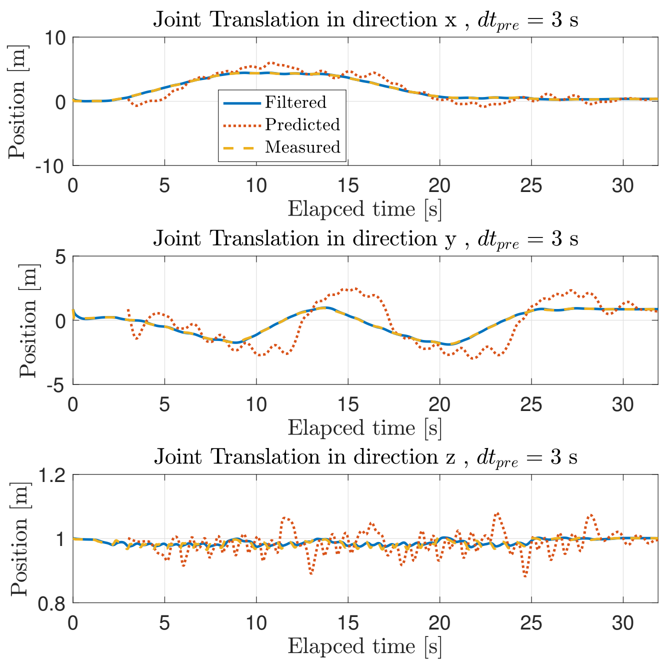


Figure A.4: Comparison between measured, filtered and predicted positions of the human predicted 3 seconds ahead. The positions in the world frame correspond to the “Translation” joint.

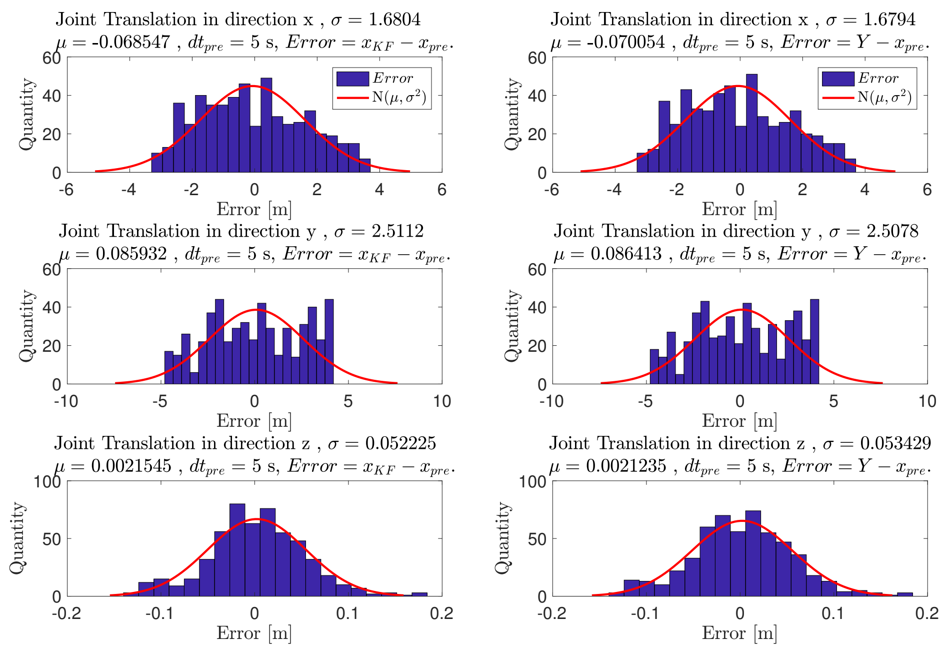


Figure A.5: Differences between the predicted human positions and the Kalman filtered positions are shown in the subfigures to the left, whereas differences between the predicted human positions and measured positions are shown in the subfigures on the right. The red line in each subfigure is a normal distribution approximated from the errors. The prediction is 5 seconds ahead.

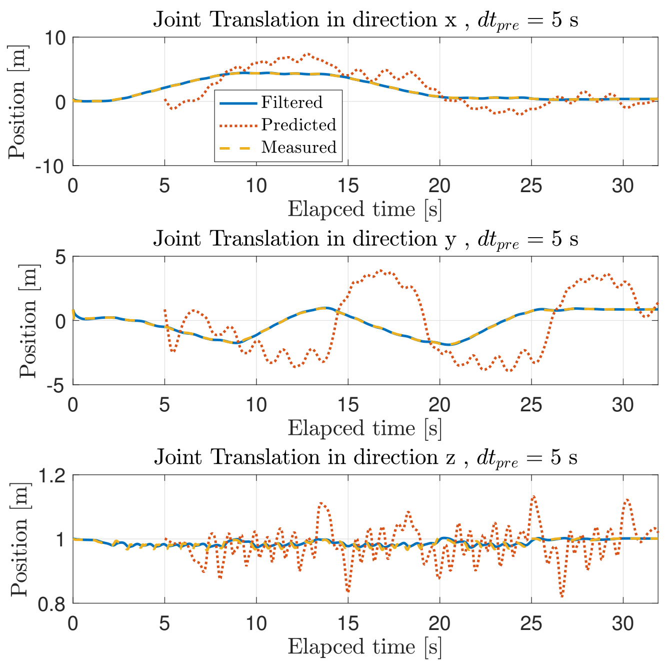


Figure A.6: Comparison between measured, filtered and predicted positions of the human predicted 5 seconds ahead. The positions in the world frame correspond to the “Translation” joint.

A.2 Result for Prediction: Arm Joints

The result for prediction of the arm joints are shown below for different prediction time (i.e., dt_{pre} is equal to 2, 3, or 5 seconds).

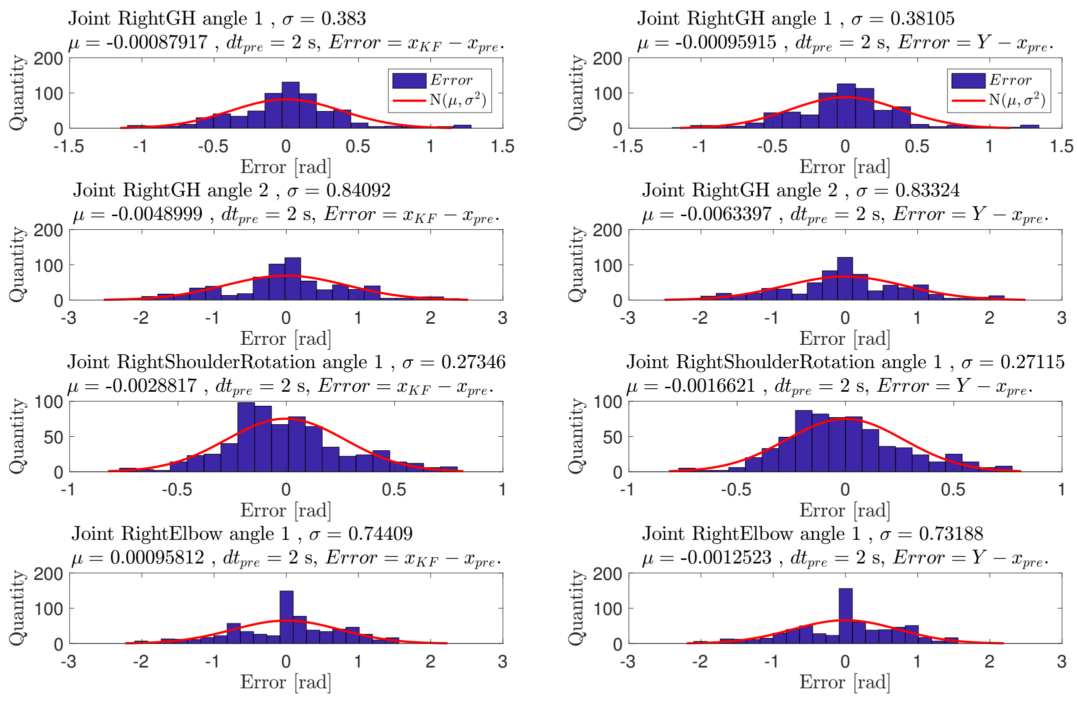


Figure A.7: Differences between the predicted human arm angles and the Kalman filtered arm angles are shown in the subfigures to the left, whereas differences between the predicted human arm angles and measured arm angles are shown in the subfigures on the right. The red line in each subfigure is a normal distribution approximated from the errors. The prediction is 2 seconds ahead.

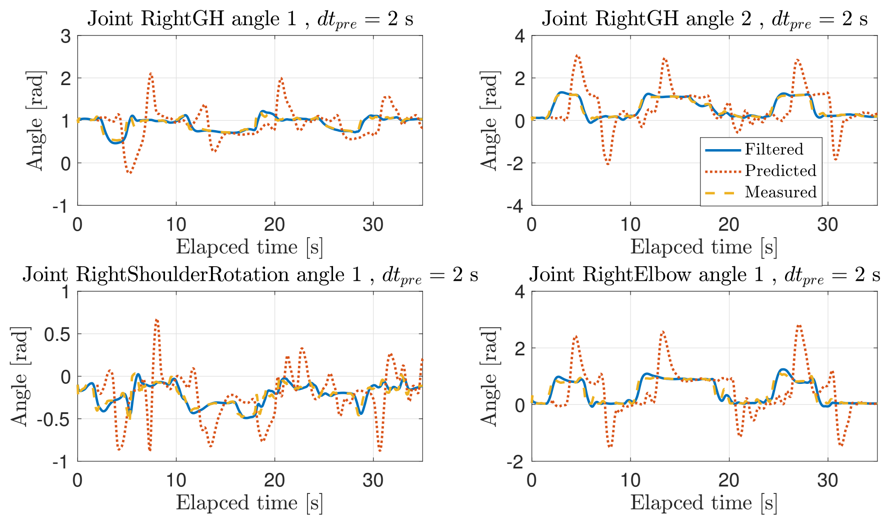


Figure A.8: Prediction, measurements and filtered data of the joints in the arm during a whole sequence. Prediction time dt_{pre} is set to 2 seconds.

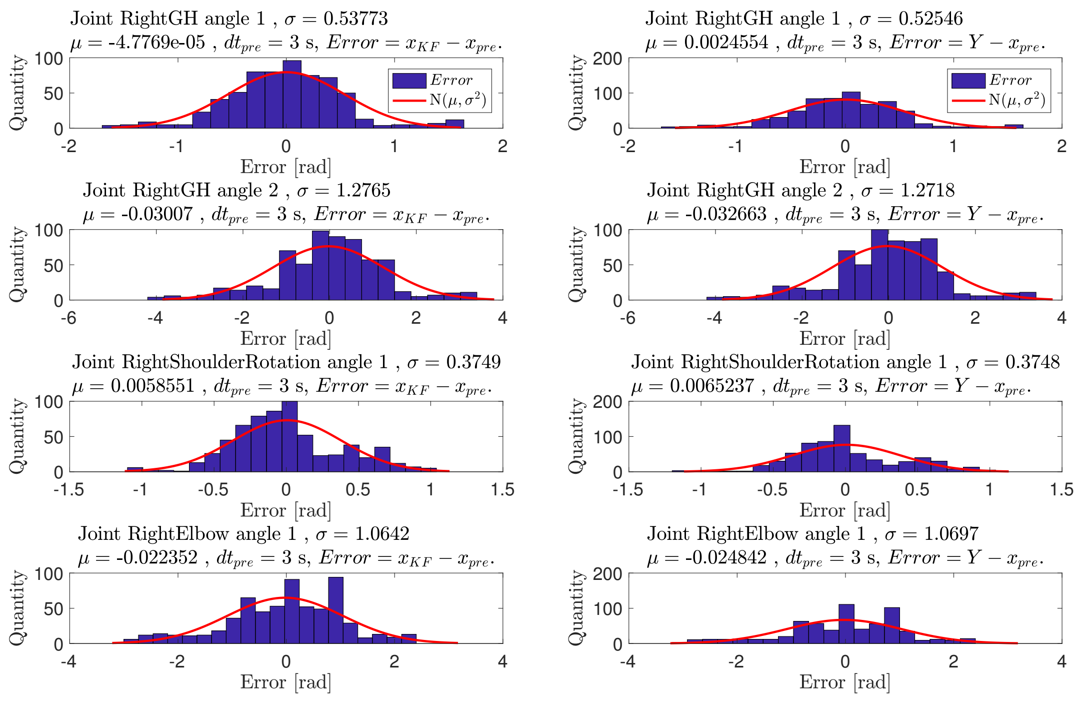


Figure A.9: Differences between the predicted human arm angles and the Kalman filtered arm angles are shown in the subfigures to the left, whereas differences between the predicted human arm angles and measured arm angles are shown in the subfigures on the right. The red line in each subfigure is a normal distribution approximated from the errors. The prediction is 3 seconds ahead.

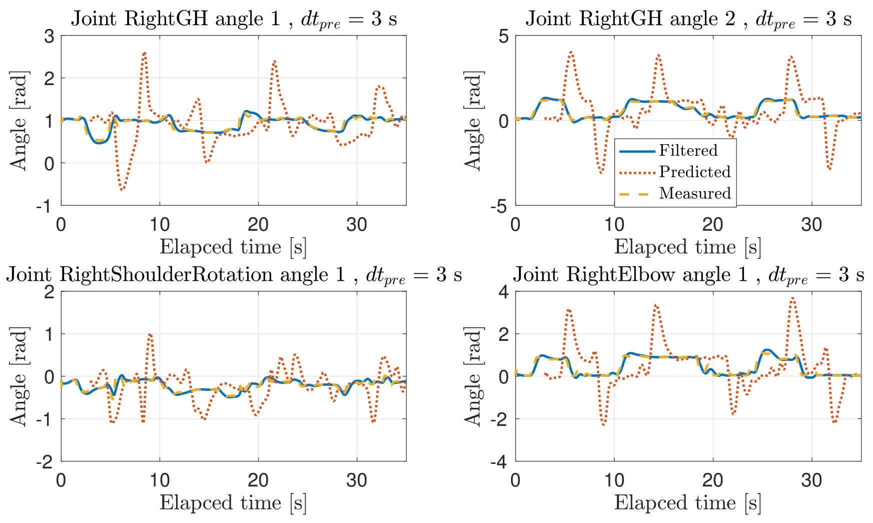


Figure A.10: Prediction, measurements and filtered data of the joints in the arm during a whole sequence. Prediction time dt_{pre} is set to 3 seconds.

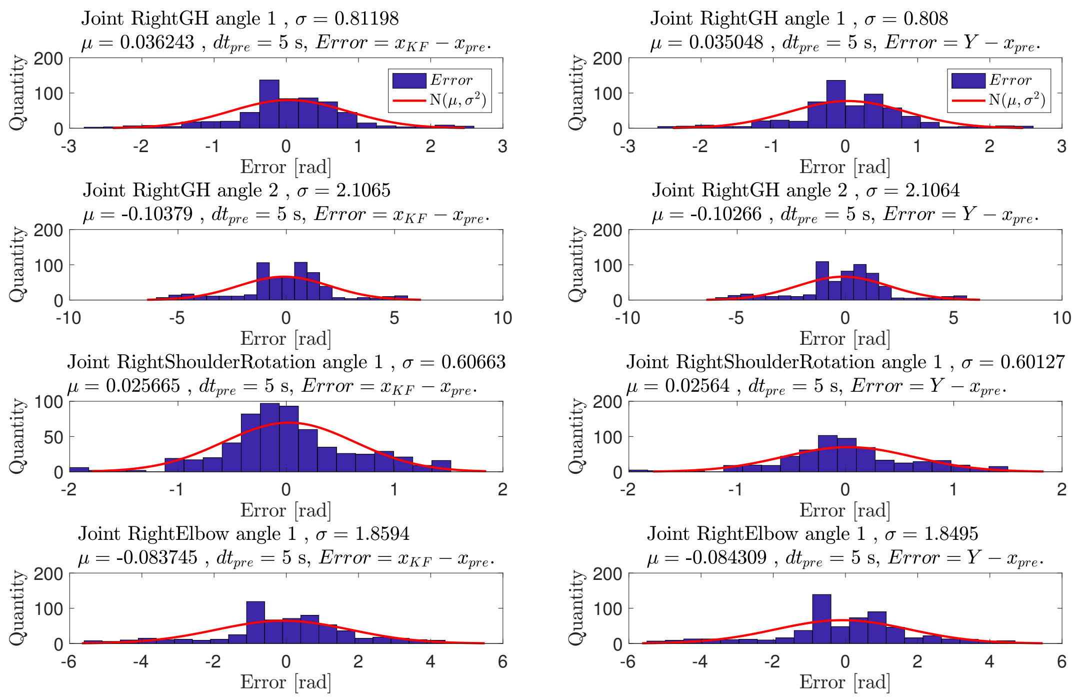


Figure A.11: Differences between the predicted human arm angles and the Kalman filtered arm angles are shown in the subfigures to the left, whereas differences between the predicted human arm angles and measured arm angles are shown in the subfigures on the right. The red line in each subfigure is a normal distribution approximated from the errors. The prediction is 5 seconds ahead.

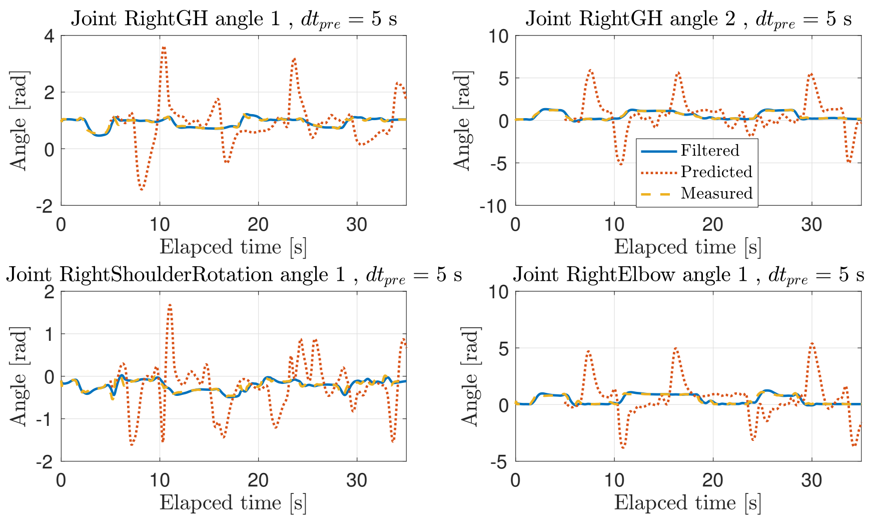


Figure A.12: Prediction, measurements and filtered data of the joints in the arm during a whole sequence. Prediction time dt_{pre} is set to 5 seconds.