Online Trajectory Generation for Robot Manipulators in Dynamic Environment — An Optimization-based Approach

by

Chi-Shen Tsai

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering: Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Fall 2014

Online Trajectory Generation for Robot Manipulators in Dynamic Environment — An Optimization-based Approach

Abstract

Online Trajectory Generation for Robot Manipulators in Dynamic Environment — An Optimization-based Approach

by

Chi-Shen Tsai

Doctor of Philosophy in Engineering: Mechanical Engineering

University of California, Berkeley

Professor Masayoshi Tomizuka, Chair

Interest in robot manipulators interacting with dynamic environments has been continuously growing because of the increasing demand for industrial robot collaboration. Human–robot collaboration and robot–robot collaboration are the two scenarios of robot collaboration that have generally been considered. The difficulties of such applications may be described from two perspectives: a good perception of environment and a proper algorithm to react to the dynamic environment for the robot manipulators. Online trajectory generation is one of the approaches for robot reaction. In the generation of the trajectory, the transformation between joint space and task space is necessary since the sensor measurement of the environment is in task space and the trajectory of the robot manipulator is in joint space. The transformation needs to be done online in a dynamic environment and hence easily results in an exponential increase of the computational load.

This dissertation proposes a safety index and the associated robot safety system in order to assess and ensure the safety of the agent in the collaboration scenarios. The agent could be a human worker in human–robot collaboration or another robot in robot–robot collaboration. In the robot safety system, the online trajectory generation algorithm is formulated in the optimization-based trajectory planning framework. The safety index is evaluated using the ellipsoid coordinates attached to the robot links that represents the distance between the robot manipulator and the agent. To account for the inertial effect, the momentum of the robot links are projected onto the coordinates to generate additional measures of safety. The safety index is used as a constraint in the formulation of the optimization problem so that a collision-free trajectory within a finite time horizon is generated online iteratively for the robot to move toward the desired position. To reduce the computational load for real-time implementation, the formulated optimization problem is further approximated by a quadratic problem. Moreover, a heuristic strategy is proposed to select the active constraints for the next iteration so as to further reduce the computational load. The safety index and the proposed online trajectory generation algorithm are simulated and validated in both a two-link planar robot and a seven-DOF robot in human–robot collaboration and robot–robot collaboration. Simulation results show that the proposed algorithm and robot safety system are capable of generating collision-free and smooth trajectories online.

The proposed algorithm has been extended to consider measurement noise in the agent information. Two possible approaches have been proposed for handling zero-mean Gaussian noise in the agent information.

To my family, Karen, and my friends.

Contents

List of Figures

List of Tables

Acknowledgments

This dissertation is the result of my six years of hard work in UC Berkeley. This dissertation could not have been written without all the help and support from people in my life in the past years. First and foremost, I would like to give my sincere respect and thank to my research advisor Professor Masayoshi Tomizuka. Under his guidance, I have learned much in the area of control theory. But more importantly, he gave me an opportunity to learn how complicated graduated level research can be and what a correct attitude should be when doing research. I would also like to thank Professor Tomizuka's wife, Miwako sensei, for always being nice and cheerful as a host when we had a party in their home.

Special thanks go to the Industrial Technology and Research Institute, Taiwan R.O.C. (ITRI), the sponsor for the work in this dissertation. In particular, I would like to thank Professor Jwu-Sheng Hu for supporting and encouraging me on my research these past years. He was always willing to discuss with me about my research and gave me precious advice.

I would like to thank Professor J. Karl Hedrick, Professor Andrew Parkard, and Professor Shmuel S. Oren for their support and encouragment during the time I prepared for the qualifying examination. They also kindly provided their available time and willingness to serve as my dissertation committee. Their valuable feedback and support have motivated me to further improve the quality of this dissertation.

The endless support from my family and Karen always helped me walk through the good and the bad times in the past years. They are always there to comfort me, listening to my life, my happiness, my depression, and everything.

I would also like to thank many individuals I met in Berkeley. I would like to thank Shih-Yuan, Mike, Max (Xu), Selina, Oak (Kan), Theresa, and Raechel. They are good company on the journey of my study in Berkeley. They definitely have broadened my horizon and gave me a chance to understand many different cultures. I would like to give my thanks to some Mechanical Systems Control Lab (MSC Lab) alumni for their willingness to teach, help, and guide me. They are KC (Kyoungchul), Joonbum, Emma (Shuwen), Hoday, Sanggyum, Evan, Sugi, and Wenjie. I will not forget how much I learned about the attitude of doing research from the experience of working with Emma and Wenjie. I would like to give my thanks to the current members of MSC Lab, namely Chen-Yu, Chung-Yen, Yaoqiong, Kevin, Hsien-Chung, Changliu, Junkai, Te, Dennis, Yizhou, Cong, Xiaowen, Wenlong, Yu, Minghui, and Shiying.

Chapter 1

Introduction

1.1 Human–Robot and Robot–Robot Collaboration in Industrial Automation

Robot manipulators have been widely used in assembly lines for mass production for decades. They are the key components in automated production since they provide highly accurate motions, capability of handling heavy loadings, and high repeatability. Recently, human–robot collaborations and robot–robot collaborations have attracted more and more attention because of their synergistic flexibility in industrial production.

1.1.1 Human–Robot Collaboration

For the purpose of production efficiency, robot manipulators are normally operated at a high speed, and thus are isolated by fences from human workers for safety concerns. Conventionally, when the overlap of robot manipulator workspaces and human worker workspaces is necessary, the motions of robot manipulators are ceased. Recently, with the need of shorter product life cycles and more complex products, more attention from both the industrial field [59, 69, 31] and the academic community [66, 64, 51, 33] has been paid to release robot manipulators from fences and increase the capability of human–robot collaboration. Human–robot collaboration intends to combine the benefits from both sides: the adaptability and the decision making skill from human workers, and the precision and the strength from robot manipulators.

Generally speaking, human–robot collaborations can be classified into two types depending on the existence of intentional physical contact between these two sides. In the type with intentional physical contact, robot manipulators assist human workers to move heavy objects by holding an object and compensating for the object's inertia. On the other hand, for the type without intentional physical contact, robot manipulators act like co-workers and cooperate with human workers while avoiding physical contact and collisions. For example, robot manipulators could be workpiece dispensers that provide assembly components to human workers during the assembly process [64]. Another example from industry is that robot manipulators carry heavy components and finish the non-ergonomic part of the assembly process, and then the human workers continue the part which requires adaptability [69]. In such cases where robot manipulators cooperate with human workers, the workspaces of human workers and robot manipulators often overlap with each other. Because of safety concerns, the human workers and robot manipulators are either separated by space or by time, which largely reduces the production efficiency and limits the applicability of human–robot collaboration. In order to fully exploit the advantages of the human–robot collaborative assembly, ensuring the safety of the human workers inside the shared workspace is the critical issue that needs to be considered.

1.1.2 Decentralized Robot–Robot Collaboration

In addition to human–robot collaborations, robot–robot collaborations are also gaining attention from the industry for effective and flexible automation utilizing robot manipulators. A centralized control structure is normally adopted in robot–robot collaborations, in which the reference trajectories for all robot manipulators are programmed prior to the production operation. The preparation effort for the centralized control structure, however, is huge in order to prevent any collisions with each other, and it gets worse when the number of the robot manipulators in the collaboration team increases. It is an undesired disadvantage for an assembly process with short product life cycles.

Some focus has been shifted to decentralized robot–robot collaboration [17, 45, 49], in which two or more robot manipulators perform either the same tasks, such as load transport [49] and dexterous grasping [46], or different tasks, such as coordinate welding [70] and cooperative assembly, for the same assembly goal. In decentralized robot–robot collaboration, only the reference trajectories of some robot manipulators are designed prior to the production operation. These manipulators with predefined trajectories are called the master robot manipulators, and the others are the slave robot manipulators. Then the trajectories of the slave robot manipulators are generated in an online fashion based on the motions of the master robot manipulators. For all these applications of decentralized robot–robot collaboration, collision avoidance is also an important issue and needs to be addressed.

1.2 Safety Concerns in Robot Collaborations

To release the robot manipulators from fences and put them in the applications of robotics collaborations, the safety concern takes priority over any other matter. For decentralized robot–robot collaboration, the safety concern can be split into two aspects: the control strategy for collision avoidance among the robot team and the detection and reaction to a collision if it occurs. The former corresponds to the safety prior to the occurrence of a collision, and the latter is for the post-collision safety.

Table 1.1: Aspects of safety concerns in human–robot collaboration and decentralized robot–robot collaboration.

	Human–Robot collaboration	Decentralized robot–robot collaboration
Safety concerns prior to collisions	• Perception of the human workers • Control strategy for collision avoidance	• Control strategy for collision avoidance
Post-collision safety concerns	• Detection and reaction to collisions • Mechanical design of the robot manipulator	• Detection and reaction to collisions

The safety of the human workers in human–robot collaboration requires a significantly higher level of attention. Any harm to human workers caused by robot manipulators needs to be prevented. Compared to decentralized robot–robot collaboration, the safety of the human workers needs to be considered in two more aspects: the perception of the human workers, and the mechanical design of the robot manipulators. The perception of the human workers means the acquiring of the knowledge of the vicinity of the robot manipulator, such as the positions of the human workers. This aspect and the control strategy for collision avoidance account for the safety prior to a collision, while the mechanical design of the robot manipulators and the detection and reaction to a collision are for the post-collision safety. These four aspects are summarized in Table 1.1 and introduced as follows.

1.2.1 Perception of the Human Workers

The robot perception of the human workers is the first key step in ensuring the safety of the human workers in the shared workspace. A poor sensory system which cannot provide precise or even adequate information about the position of the human workers could result in failure of keeping the human workers safe.

Proximity sensors, such as sonars and lasers, have been used for years in the field of mobile robotics for the detection of humans and obstacles; for example, [4, 67, 6]. In these cases, the mobile robot and the obstacles are treated as objects moving on a two-dimensional plane. However, the limitation from the working principle of the sonar and laser sensors has hindered the applicability of these sensors in three-dimensional space in an industrial assembly line.

Recently, in consequence of the progress in the computer-vision field and the decline in both the price and the size of cameras, vision-based sensors have been more frequently utilized in the perception of the environment. In team@work proposed by Thiemermann [34, 65], three RGB cameras in trinocular arrangement are used to detect the human workers in the cooperative work cell. The positions of a human's hands and neck inside the workspace of a SCARA robot can be reconstructed by a photogrammetric method using the characteristic color and texture features of the human skin extracted from camera images. Henrich and Gecks [24, 15] presented a four RGB camera system which is used to monitor and guide a robot manipulator in a dynamic environment. The obstacle detection is achieved by comparing the current four images with the reference images of the empty workspace taken in a preliminary setup. Guggi and Riner [19] deployed the distributed smart camera system to monitor the workspace of a crane. Every camera in the distributed camera system performs a local image analysis in order to identify the shape of the obstacles in their field of view. Shape information from the four cameras is then fused together to estimate the 3D structure of the obstacles.

Microsoft Kinect sensor [48] is a commercial product which combines a RGB camera, a infrared projector and a receiver, and a microphone array. It has been broadly used in both the academic community and industry for human position and gesture detection since its debut in 2010. For example, it was utilized in [57, 16] to capture depth images, and the relative distance between the human worker and the robot manipulator can be obtained from a depth space built upon the depth images. Morato *et al.* [50] proposed a multiple Kinect-based exteroceptive sensing framework to capture a human worker as a twenty-joint skeleton model. Instead of processing the depth space directly, their system fused the position data of the skeleton model in the local coordinates from each Kinect in a filter scheme to obtain a refined human model in the global coordinate.

SafetyEYE [59] from Pilz is a commercial safety system with three cameras mounted on the roof of the workspace for 3D workspace surveillance. Whenever a human worker invades into the predefined safety zones, the motions of the robot manipulators are slowed down or even stopped. These safety zones are defined as static polygons with a given height, and cannot be changed during active surveillance.

1.2.2 Control Strategy

In the perspective of robot control, human–robot collaboration can be viewed as a planning problem in a dynamic environment, where the human workers are treated as dynamic agents and the goal of the robot manipulators is to achieve their tasks and simultaneously avoid collisions with the human workers. The same concept can be applied to decentralized robot–robot collaboration, in which the master robot manipulators are the dynamic agents, and the slave robot manipulators are desired to achieve their tasks without any collisions with the master robot manipulators. Two frameworks are generally considered in such planning problems: reactive control framework, and optimization-based path planning framework.

For the reactive control framework, the control input to the robot manipulator is computed based on the current manipulator state and the current agent state. The potential field approach [32] proposed by Khatib is one of the well-known reactive control approaches. In the potential field approach, the distance between the goal and the current positions of the end-effector is transformed into a vector field to generate the attracting control torque. There also exist the repelling control torques generated by another vector field built from the relative distances between the agents to the predefined concern points on the robot manipulator. The attracting control torque and the repelling control torques then are combined to form the control input to the robot manipulator. One drawback of this approach is that the potential field does not capture the relative motion between the robot manipulator and the agents. Another is that it may suffer from the local minimum problem, in which the robot manipulator gets stuck in some other positions before reaching the goal position. This problem is caused by the cancellation between the attracting control torque and the repelling control torque.

Many approaches are inspired by the potential field approach and attempt to overcome its downsides. The circular field approach [63] proposed by Singh *et al.* is motivated by a charged particle in a magnetic field generated by a current flowing around the agents. In the circular field, artificial electro-magnetic-fields are generated by the virtual current around the surface of the agents so that the path of a point robot is rotated around the agents. This method is further elaborated by Haddadin *et al.* [60, 22].

The kinetostatic danger field [41, 72] is built from the configuration and velocity of the robot manipulator, and the integral of the field along the robot links gives the danger index for the robot links. A feedback-like control law is derived from the danger index and the danger field. Since only current states are considered, algorithms in this framework do not guarantee smooth trajectories and often suffer from the local minimum problem.

Another framework is optimization-based path planning, in which the optimal path or trajectory of a short horizon from the current manipulator state is obtained by solving an optimization problem. This framework has gained attention recently since the boost of computing power makes online implementation possible. In [3], the collision-free path of the end-effector is searched by solving an optimization problem in which the end-effector is only allowed to move along a set of predefined directions. In the formulation of the optimization problem, the robot manipulator and the human worker are represented by two sets of spheres, and the reciprocal of the distances between all pairs of spheres from these two sets are penalized in the objective function. In [14] and [13], the geometrical relationships between the polygon-modeled agents and the robot manipulator are decomposed into multiple linear inequality constraints with binary variables to control the activation of each inequality constraint. Mixed Integer Linear Programming (MILP) is then utilized to solve the optimization problem. Chung *et al.* [8] proposed elliptical sets for links of a two-link planar robot as the non-collision zones. These zones are then incorporated in the robot dynamic model as the hard constraints in the optimization-based approach to generate a collision-free trajectory. However, because of the nonlinearity of the robot dynamics, solving the optimization problem becomes the bottleneck and nearly infeasible even in such a simple case of a two-link planar robot.

Among all the approaches in both frameworks, the spatial relationship and the relative motions between the robot manipulator and the agent are normally quantified as an index for safety and evaluated from concern points on the robot manipulator to points on the agent. These values are then used in the algorithms of collision avoidance. In most implementations, however, the extension from a concern point to the entire robot results in an exponential increase of computation. Moreover, since the agent states are measured in task space and the robot manipulator states are in joint space, the nonlinear and non-convex transformation between these two spaces requires even more computing power.

1.2.3 Mechanical Design for Compliant Manipulators

The methods in the previous two sections play important roles in the safety considerations prior to the occurrence of collisions. On the other hand, when a collision occurs, the safety consideration highly depends on the detection of the collision and the capability of absorbing the impact force caused by the collision. Many innovative mechanical designs of the robot manipulators are proposed to improve the compliance of the robot manipulator for the purpose of absorbing the impact force.

Series elastic actuation (SEA) was first introduced by Pratt and Williamson [56] in which a passive mechanical spring is deliberately connected between the motor output and the robot link. It has the benefits of a simple design and lowering the robot joint stiffness, and hence increases the safety without making the volume of the robot joints too bulky. SEA has been widely used in many commercial robot manipulators, such as the lightweight robot (LWR) [1] designed by Deutsches Zentrum für Luft- und Raumfahrt (DLR), and Baxter [58] from Rethink Robotics.

Zinn *et al.* [74, 73] proposed a new actuation approach based on the design of the SEA, named distributed macro-mini (DM^2) actuation. In their design, the actuator in the SEA is the main torque source, and is called the macro actuator. Another small actuator with a high frequency bandwidth, referred as the mini one, is attached to the robot link for fast responses. Namely, a pair of actuators is connected in parallel to the same joint, but on different sides with respect to the transmission gear. This design is capable of reducing the effective inertia of the robot links and of overcoming the bandwidth limitation occurring in the SEA. Shin *et al.* [62] proposed an evolution of the DM^2 approach in which compliant pneumatic muscles are utilized as macro actuators for a more compact design, instead of electric actuators.

Variable stiffness transmission (VST) is another approach which aims to dynamically change the robot joint stiffness so that the robot manipulator can exhibit a low stiffness when subjected to a collision force greater than the injury tolerance, but can maintain a high stiffness at other times. Many new mechanisms have been developed to achieve the variable stiffness, for example, [53, 7, 61, 54].

1.2.4 Detection of the Occurrence of Collisions

In addition to the design of the compliant manipulators, some devices are developed for detecting the occurrence of collisions without altering the mechanical structure of the robot joints. The BI-Jacket [71] proposed by Yu *et al.* is a simple deformable collision detection sensor which consists of a foam cover and multiple air pressure sensors. The foam cover contains several air chambers, and each of them is well-sealed and connected to an air pressure sensor. Because of the deformable material and the air chambers, the foam cover serves as a buffering material between the robot link and the agent and absorbs partial impact force when a collision occurs. The deformation of the foam makes the air pressure inside the air chambers change, which enables the robot system to detect the occurrence of a collision from the air pressure measurement. Jeong *et al.* [30, 29] has developed a similar device which consists of multiple flexible air tubes and an air pressure sensor.

De Luca and Haddadin [11, 10] have developed a sensorless algorithm to detect a collision from the total energy and generalized momentum of the robot manipulators using merely the robot manipulator states. When a collision occurs, this method not only detects the occurrence, but also identifies which robot link has collided. This algorithm was implemented on the third generation of the LWR [21].

1.3 Safety Standards

For industrial applications, several standards have been made since 1999 to standardize the safety of human workers who are in the same environment, but do not necessarily share the workspace, with robot manipulators at the same time. Conservative safety guidelines, such as ANSI/RIA R15.06-1999 [2], which is proposed by the Robotic Industries Association and approved by the American National Standards Institute, prescribe the requirements for human workers using robot systems in industry. They require the physical separation between robot manipulators and human workers. A relaxation toward more collaborative requirements was introduced in ISO 10218-1 [27] (International Organization for Standardization). For the case of human–robot collaboration with the absence of intentional physical contact, this standard states that one of the following requirements always has to be fulfilled: the tool center point (TCP) velocity or flange velocity must be at most 0.25 m/s, the maximum dynamic power at most 80 W, or the maximum static force at most 150 N. The ISO 10218-1 was harmonized with the ANSI/RIA R15.06-2012, which is an update to the 1999 version. However, in [21], it was experimentally demonstrated that these requirements tend to be unnecessarily restrictive, and therefore unnecessarily hinder the performance of the robot manipulator.

1.4 Dissertation Overview

1.4.1 Objective and Contributions

The objective of this dissertation is to develop a robot safety system for applications to robot collaborations so that the robot manipulator can achieve its own task under the guarantee of the safety of the agent. The agent, as mentioned, can be either the human worker in human–robot collaboration or the master robot manipulator in decentralized robot–robot collaboration. Ensuring the safety of the agent implies that no collisions will occur between the agent and the robot manipulator during collaboration. Furthermore, this dissertation is focusing on the high-level control strategy of the robot manipulator based on the assumption that the perception of the human workers has been provided.

The contributions of this dissertation are listed below.

- A measure to evaluate the safety of the agent in the shared workspace is designed. This safety index is designed in the ellipsoid coordinates, which are constructed upon the local coordinates of the robot links and hence move along with the robot links. With the design, the safety index for a pair of the robot link and the agent can be computed directly without setting any concern points. The introduction of the ellipsoid coordinates greatly alleviates the amount of computation when the assessment of the agent safety regarding the entire robot manipulator is considered.

- An online trajectory generation algorithm in the optimization-based planning framework is proposed to generate new trajectories for the robot manipulator. The optimization problem is formulated so as to drive the robot manipulator to achieve its task and avoid collisions with the agent simultaneously. Plus, the generated trajectories are inherently smooth by penalizing the joint jerk in the objective function of the optimization problem.

- The formulated optimization problem is originally nonlinear and non-convex because of the use of forward kinematics for transformation of the robot manipulator states from joint space to task space. This formulation then is approximated by a quadratic formulation. The performance of the quadratic formulation is validated by simulations and is shown to be as good as that of the original formulation in terms of achieving the manipulator's task and collision avoidance.

- To further relax the assumption in the perception of the human workers, the position information is extended to contain measurement noise. The proposed algorithm is then extended to handle the stochastic model of the agent positions.

1.4.2 Dissertation Outline

The outline of this dissertation is as follows.

Chapter 2: Simulation Environment and Performance Assessment

In this chapter, the structure of the robot safety system is proposed. Then the simulation environment for the validation of the proposed algorithm with the robot safety system is introduced. Simulations are conducted for the validation of the proposed online trajectory generation algorithm in the MATLAB environment. Two robot manipulators are chosen as the testbed in simulations. The first one is a two-link planar robot, which serves as a good illustration to demonstrate the concept of the proposed algorithm because of its simple mechanical structure. The second robot manipulator is a seven degree of freedom (DOF) robot manipulator designed by Industrial Technology Research Institute (ITRI), Taiwan. The second robot manipulator is used to illustrate the scalability of applying the proposed algorithm to a robot manipulator with joint redundancy in three-dimensional task space. This chapter concludes with performance considerations in three aspects for the proposed algorithm, including safety of agents, smoothness of generated trajectories, and feasibility for real-time implementation.

Chapter 3: Design of Safety Index

In this chapter, the design of a safety index for the agents, either a human worker or another robot manipulator, inside the shared workspace is introduced. The safety index is designed based not only on the relative distance level, but also on the robot links momentum level. The relative distance is selected intuitively as a factor since the physical separation between the robot manipulator and the agent implies that no collisions happen. Robot manipulators are normally operated at a high speed for production efficiency resulting in a large momentum. The larger the momentum, the more the impact of collision. Furthermore, a manipulator with larger momentum normally requires a longer time and a longer distance to fully stop its motion after the stop command is issued. Hence, the momentum of the robot links is another factor that needs to be considered. The safety index is computed in the ellipsoid coordinates attached to robot links, and hence the computational load can be greatly alleviated.

Chapter 4: Online Trajectory Generation Algorithm

In this chapter, the proposed online trajectory generation algorithm is stated in detail. This algorithm is developed under the optimization-based path-planning framework with the use of a receding horizon strategy, discretized kinematics model, and the safety index designed in Chapter 3. By solving the optimization problem, the solution is a trajectory within a short time horizon which achieves the functions of leading the robot manipulator toward the goal position and avoiding any collisions with agents simultaneously.

The formulated optimization problem, with the use of the transformation between joint space and task space, is highly nonlinear and non-convex. In order to reduce the computational load for the online trajectory generation, the formulated optimization problem is approximated by a quadratic problem with linear constraints.

In the formulation of the quadratic problem, the inequality constraints that account for the safety are considered for each pair of the ellipsoid coordinate and the agent ellipsoid in every time step within the planning time horizon. The approximation of these inequality constraints results in a large increase of the computational load. A heuristic selection strategy is proposed to effectively choose those inequality constraints which are active or almost active so as to prevent the increase of the computational load.

Chapter 5: Simulation Studies

In this chapter, the proposed online trajectory generation algorithm along with the robot safety system proposed in Section 2.2 are implemented and simulated. Three formulation—the original formulation of the optimization problem, the quadratic problem, and the quadratic problem with the selection strategy—are implemented and simulated for comparison. The quadratic problem is an approximation of the original optimization problem. The selection strategy is an heuristic method to further reduce the size of the quadratic optimization problem. Simulations are conducted on the two robot manipulators for validation. Simulation results show that the quadratic formulation has an equivalent performance as the original formulation in terms of ensuring the safety of the agents in the shared workspace. The quadratic formulation, however, can be solved much faster compared to the nonlinear formulation. The simulations of the quadratic formulation with and without the selection strategy show almost identical performance, but the selection strategy has greatly reduce the computational load.

Chapter 6: Measurement Noise in the Agent Information

In this chapter, noisy measurement of the agent position information is considered. A Gaussian random vector is used to represent the measurement noise in sensing the agents in the shared workspace. Two approaches are proposed to handle the contaminated agent information. The first one is to consider the measurement noise in the agent position estimation block, and then this block outputs the best estimate of the agent position to the decision making block. An alternative approach is to handle the measurement noise in the formulation of the optimization problem in the decision making block. The inequality constraints accounting for the safety in the optimization problem become chance inequality constraints. To realize the inequality chance constraints in implementation, these chance constraints are transformed to normal deterministic constraints by utilizing the mathematical similarity between the probability density function (PDF) of a Gaussian random vector and the ellipsoid coordinates. The quadratic approximation then can be applied again for a fast online trajectory generation.

Chapter 7: Conclusion and Future Research

Remarks about the proposed algorithm and the conclusions of this dissertation are summarized in this chapter. Then some open issues and possible future work are discussed.

Chapter 2

Simulation Environment and Performance Assessment

2.1 Introduction

In this chapter, the structure of the robot safety system for the online trajectory generation algorithm proposed in this dissertation is introduced. Then the simulation environment for the validation of the proposed algorithm and the robot safety system is given. In the simulation environment, two robot manipulators are used for validation, two scenarios of simulations are selected, and one assumption about the robot perception is made. For the assessment of the proposed algorithm in simulations, including the safety of the agents, the smoothness of the generated trajectories, and the suitability for the real-time implementation are considered.

2.2 Robot Safety System

The robot safety system in this dissertation modifies the original desired trajectory to ensure that the robot completes the task while ensuring the safety of the agent. An overall structure of the robot safety system is proposed and presented in Figure 2.1. It consists of four components: a sensor block, an agent position estimation block, a controller block, and a decision making block. The sensor block may contain several different sensors, such as stereo cameras, proximity sensors, or even a Microsoft Kinect sensor [48], to detect the agents inside the robot workspace. Note that the agents could be either human workers or the master robot manipulators, depending on the application of robot collaborations. The agent position estimation block then transforms the measurements from the sensor block into a set of ellipsoids which can effectively envelop the space occupied by the agents. Figure 2.2 shows examples of a human worker and a master robot manipulator which are represented by two sets of ellipsoids. The sensor block and the agent position estimation block accounts for the robot manipulator perception so as to provide the knowledge of

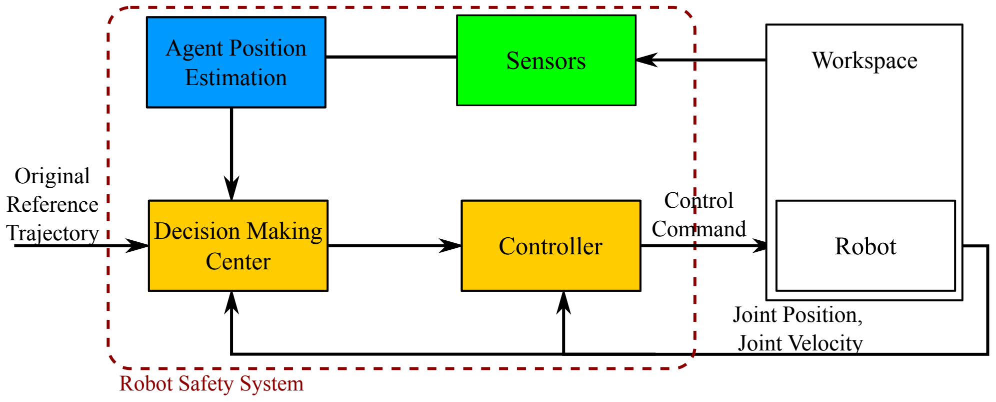
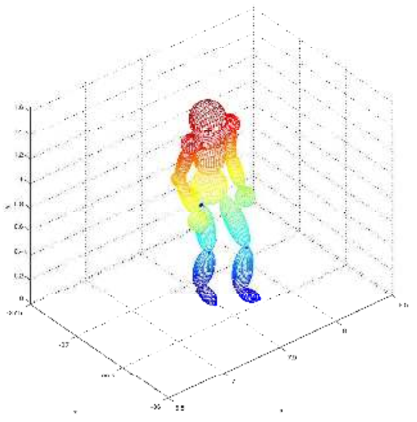
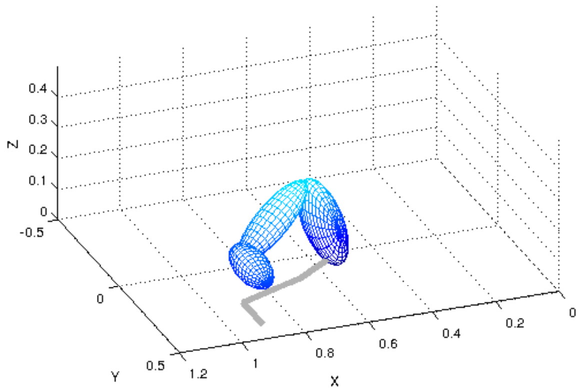


Figure 2.1: The overall structure of the safety system for robot manipulators.



(a)



(b)

Figure 2.2: Examples of (a) a human worker and (b) a robot manipulator being enveloped by two sets of ellipsoids.

the vicinity of the robot manipulator. The reason that ellipsoids, rather than spheres or polytopes, are selected to represent the agents is that the volume of an ellipsoid can be written in a simple mathematical form without using any logical operations. Furthermore, the shape of an ellipsoid can be controlled separately and easily along each dimension via its parameters. With these advantages, an agent can be enveloped efficiently and simply using a few ellipsoids.

The decision making block is the place where the online generation of trajectory exists. It takes the states of the agent ellipsoids and the robot manipulator, computes the safety index, and then generates a new trajectory by solving an optimization problem. The trajectory includes the position, the velocity, and the acceleration of the robot joints which will lead the robot manipulator toward its goal position and avoid collisions with the agents. The controller block then generates lower-level control commands for the robot manipulator to track the collision-free trajectory generated by the decision making block.

2.3 Simulation Environment

2.3.1 Assumption

In the later chapters, the focus will be put on the proposed online trajectory generation in the decision making block while the perception of the robot workspace is assumed to be ready. Specifically, it is assumed that sensors in the sensor block are well-selected to provide effective measurements for the agent position estimation block to transform the measurements into a set of ellipsoids.

For decentralized robot–robot collaboration, this assumption is inherently valid since the joint position of the master manipulator should be available through communication with the robot system of the master robot in every controller time step. The parameters of ellipsoids to envelop the entire master robot manipulator can be properly selected prior to the collaboration operation. Hence, the information of these agent ellipsoids can be computed in every time step.

For human–robot collaboration, this assumption is reasonable since there are existing literature studies and commercial products devoted to the perception of the workspace for the robot manipulator as detailed in Section 1.2.1. Among all the sensors, computer vision-based sensors, such as a stereo camera or Microsoft Kinect, are widely utilized among all possible sensors in both the academic community and industrial applications. For instance, Kulić and Croft [38] used a stereo camera to build the depth map, and then the information of the sphere set is generated to represent the human worker by the hierarchy spheres representation [47]. A Microsoft Kinect [48] is able to output the position of each joint in the twenty-joint skeleton model in the local coordinate of the Kinect. Because of the limitation of these sensors, the sampling rate of this type of sensor is normally around 30 Hz [18, 48], which is much lower than the controller sampling rate in most robot systems.

In this dissertation, the computer vision-based sensors are assumed to be selected in the sensor block of the robot safety system. The information of the agent ellipsoids is assumed to be available in every T_c controller time steps.

2.3.2 Robot Manipulators

In the simulations for validation of the proposed algorithm, two robot manipulators are selected. The first one is a two-link planar SCARA (Selective Compliance Assembly Robot Arm) robot as shown in Figure 2.3. The axes of the two joints in the two-link planar robot are parallel with the Z-axis of task space, and hence the motion of the robot end-effector is

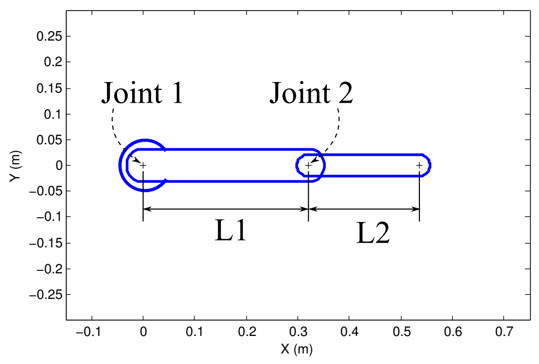


Figure 2.3: The two-link planar robot for algorithm validation.

Table 2.1: Parameters of the two-link planar robot manipulator.

	Joint limits		
	Joint position	Joint velocity	Joint acceleration
Joint 1	$(-\pi, \pi)$	(-1.2,1.2)	(-50,50)
Joint 2	$(-\pi, \pi)$	(-1.28,1.28)	(-20,20)
(unit)	(rad)	(rad/s)	(rad/s^2)

	Length
Link 1	$L1 = 0.320\ m$
Link 2	$L2 = 0.215\ m$

Table 2.2: DH Parameters of the two-link planar robot manipulator.

	a_i	α_i	d_i	θ_i
Link 1	$L1$	0	0	θ_1
Link 2	$L2$	0	0	θ_2

limited on a two-dimensional plane. For simplicity, the two links and the end-effector of the two-link planar robot are considered in the X–Y plane, and the base is at the origin of the X–Y plane. The length and motion limits for position, velocity, and acceleration for each link are listed in Table 2.1. The DH parameters [12] of the two-link planar robot arm are listed in Table 2.2 where θ_1 and θ_2 are the two joint positions.

The second robot manipulator for algorithm validation is a seven-DOF robot manipulator

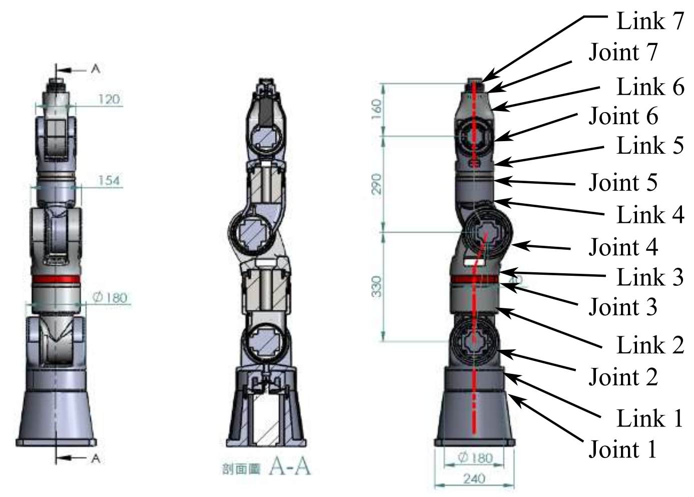


Figure 2.4: The ITRI seven-DOF robot manipulator for algorithm validation.

Table 2.3: Parameters of the ITRI seven-DOF robot manipulator

	Joint limits		
	Joint position	Joint velocity	Joint acceleration
Joint 1	$(-\pi/2, \pi/2)$	(-1, 1)	(-60,60)
Joint 2	$(-1.0, 1.0)$	(-1, 1)	(-60,60)
Joint 3	$(-1.8, 1.8)$	(-1, 1)	(-60,60)
Joint 4	$(-3.0, 3.0)$	(-1, 1)	(-36,36)
Joint 5	$(-3.0, 3.0)$	(-1, 1)	(-36,36)
Joint 6	$(-\pi/2, \pi/2)$	(-1, 1)	(-24,24)
Joint 7	$(-1.2, 1.2)$	(-1, 1)	(-24,24)
(unit)	(rad)	(rad/s)	(rad/s^2)

designed by Industrial Technology and Research Institute (ITRI), the CAD model of which is shown in Figure 2.4. The kinematic structure of the ITRI seven-DOF robot manipulator and the coordinate systems of robot links are illustrated in Figure 2.5. For simplicity, the Y-axes of all robot link coordinates are not shown in the figure, but can be built via the right-hand rule. The motion limits for joint position, velocity, and acceleration for the ITRI seven-DOF robot manipulator are listed in Table 2.3, and the DH parameters are listed in Table 2.4, where $\theta_1, \cdots, \theta_7$ are the joint angles.

The choice of the two-link planar robot manipulator is to serve as an illustration of the concept and the behavior of the proposed algorithm since it has a simple mechanical structure. On the other hand, the selection of the ITRI seven-DOF robot manipulator

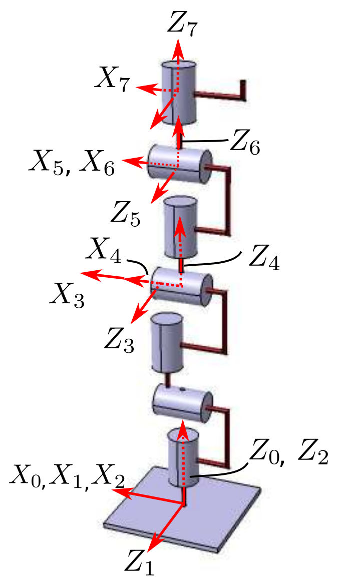


Figure 2.5: The kinematic structure and the coordinate systems of links in the ITRI seven-DOF robot manipulator. (For simplicity, the Y-axes of all coordinates are not shown in the figure, but can be built via right-hand rule.)

Table 2.4: DH Parameters of the ITRI seven-DOF robot manipulator.

	a_i	α_i	d_i	θ_i
Link 1	0	$-\pi/2$	0	θ_1
Link 2	0	$\pi/2$	0	θ_2
Link 3	0.04	$-\pi/2$	0.33	θ_3
Link 4	-0.04	$\pi/2$	0	θ_4
Link 5	0	$-\pi/2$	0.29	θ_5
Link 6	0	$\pi/2$	0	θ_6
Link 7	0	0	0.16	θ_7
(unit)	(m)	(rad)	(m)	(rad)

demonstrates the scalability of applying the proposed algorithm to a robot manipulator with joint redundancy in a three-dimensional task space.

2.3.3 Simulation Setup

The proposed algorithm and the robot safety system is validated on the two robot manipulators in the MATLAB environment. For the visualization of the simulation results, the

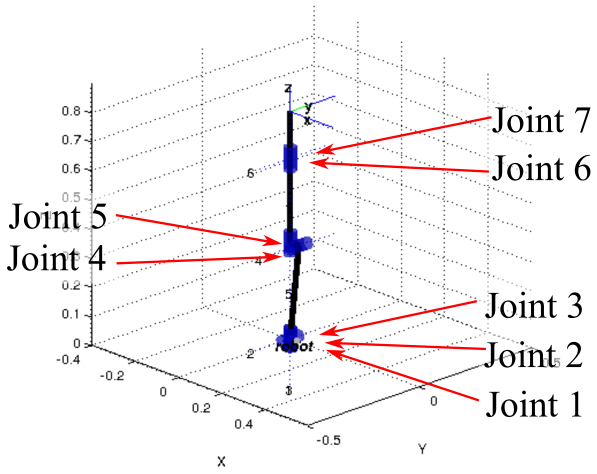


Figure 2.6: The model of the ITRI seven-DOF robot built by Robotics Toolbox [9] in MATLAB. The blue cylinders indicate joint 1 to joint 7.

two-link planar robot is built and plotted in the X–Y plane as illustrated in Figure 2.3. The ITRI seven-DOF robot manipulator is built in MATLAB utilizing Robotics Toolbox [9] as shown in Figure 2.6. In the figure, each of the blue cylinders, labeled as joint 1 to joint 7, indicates the corresponding joint.

2.3.4 Simulation Scenarios

In validation, scenarios of both human–robot collaboration and decentralized robot–robot collaboration are simulated since the human worker and the master robot manipulator are unified as the agents in the shared workspace. The details of both scenarios are stated in the following.

Human–Robot Collaboration

In human–robot collaboration, the simulation scenario is that the robot manipulator working on the non-ergonomic task next to a human worker in a close distance, similar to the situation in [69]. In the scenario, the motion of the robot manipulator is to pick up a work piece in one place, and then move to another place and finish this iteration of the assembly task. During the collaboration, the human worker reaches out his/her hand unexpectedly to the place of the assembly task when the robot manipulator is approaching toward the place or already in the place of the assembly task. Without loss of generality, the end-effector path of the robot manipulator is a straight line between two places when there is no human worker in its workspace. The details of the motions of the human worker who is in the workspace of either the two-link planar robot or the ITRI seven-DOF robot are stated below.

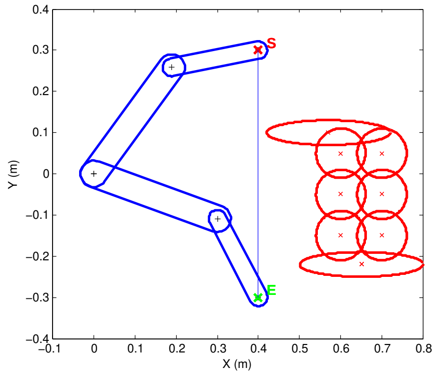


Figure 2.7: Two desired configurations of the two-link planar robot manipulator in simulations of human–robot collaboration. The straight line is the desired path for the robot end-effector. The human worker is modeled by six circles and two ellipses.

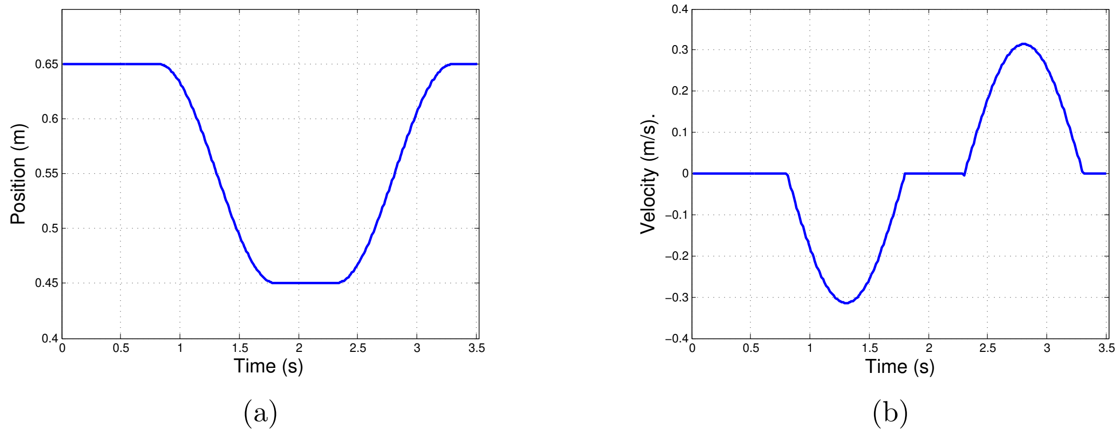


Figure 2.8: (a) The position and (b) velocity of the center of the ellipse representing the left hand of the human worker.

In the case of the two-link planar robot, the picking and the assembling places are located at $(0.4, 0.3)$ and $(0.4, -0.3)$ respectively, and are denoted by the red and the green crosses labeled as S and E in Figure 2.7. The straight line between the two positions is the desired path for the robot end-effector, and the corresponding configurations for reaching these two positions are also shown in the figure. As shown in Figure 2.7, the human worker is modeled by six circles and two ellipses in which two ellipses represent two hands, and the six circles are used to model the human worker's body. The two ellipses are identical with the radii of 0.15 m and 0.03 m, and the radius of each circle is 0.06 m. The right-hand ellipse of the human worker stays in the way of the robot manipulator during the motion of the robot manipulator. The human worker stretches out the left-hand ellipse toward the E position

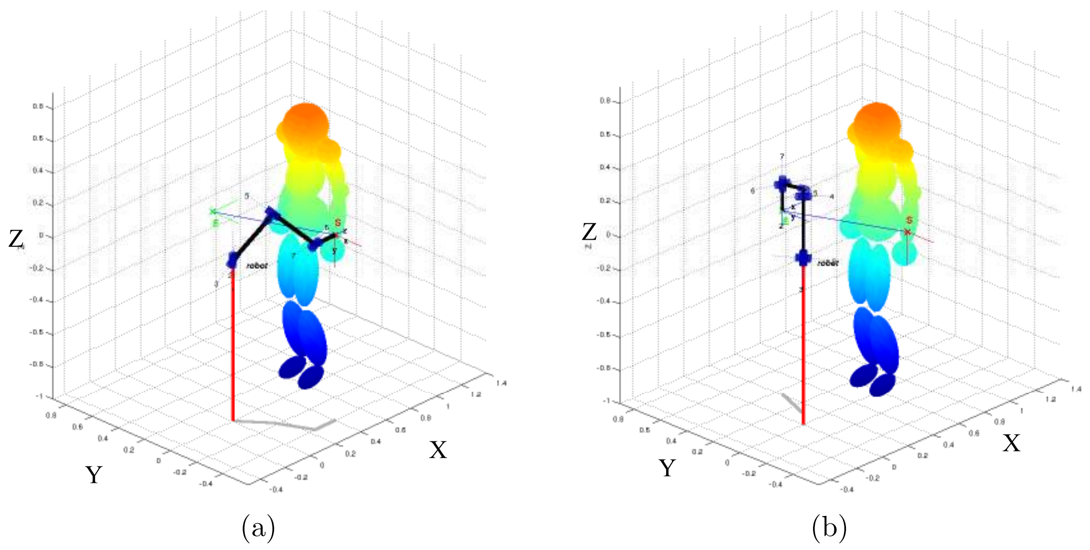
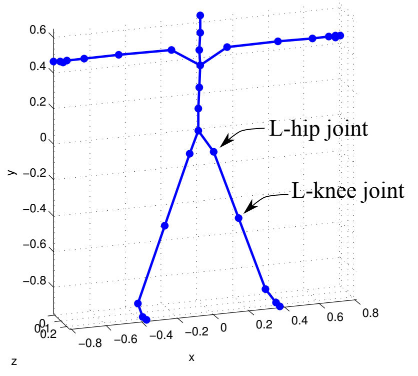


Figure 2.9: (a) The picking and (b) the assembly configurations of the seven-DOF ITRI robot manipulator in simulations of human–robot collaboration. The human worker is in the initial gesture.

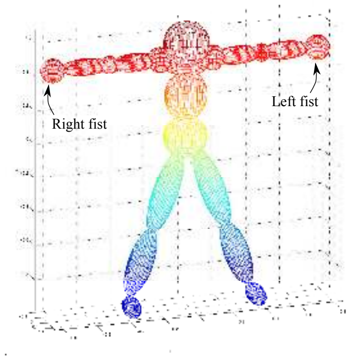
during the simulation. The starting position of the left hand ellipse center is $(0.65, -0.22)$. The motion of the ellipse is along the X-axis, and the leftmost point of the ellipse reaches $(0.3, 0. -0.22)$ during the motion. The position and velocity of the left-hand ellipse is shown in Figure 2.8.

For the case of ITRI seven-DOF robot manipulator, the picking and the assembling positions for the end-effector are located at $(0.4, -0.4, 0.15)$ and $(0.2, 0.35, 0.1)$. These two positions are indicated by S and E, and the corresponding desired configurations for the robot manipulator are shown in Figure 2.9a and Figure 2.9b, respectively. In the two figures, the short lines connected to S and E represent the normal direction and the sliding direction of the end-effector coordinate in the corresponding configuration. Note that the orientation of the end-effector changes from the picking configuration to the assembling configuration. The path of the end-effector in the original reference trajectory is a straight line with length of 0.7778 m, as shown in blue in the figure. The base of the ITRI seven-DOF robot manipulator is set to the world coordinate origin as seen in the figures.

In simulations, the human worker is standing close to the assembling position of the robot manipulator. The worker leans his/her torso forward with two hands stretching out toward the assembling position, and then returns to initial posture. The motion of the human worker is modeled using data from the Carnegie Mellon Motion Capture Database [68] and MATLAB Motion Capture Toolbox [44]. The CMU Motion Capture Database contains the position information of joints of the human body in a skeleton model, and then the set of ellipsoids representing the human worker then can be constructed from the skeleton model. Figure 2.10a and Figure 2.10b show the skeleton model of a human subject from the

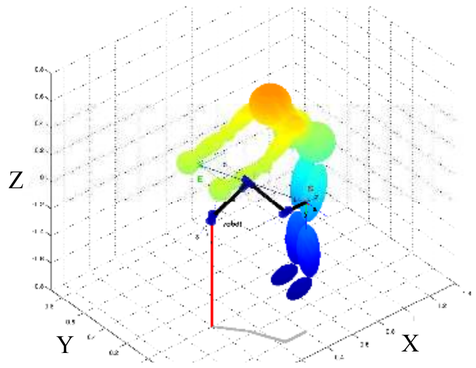


(a)

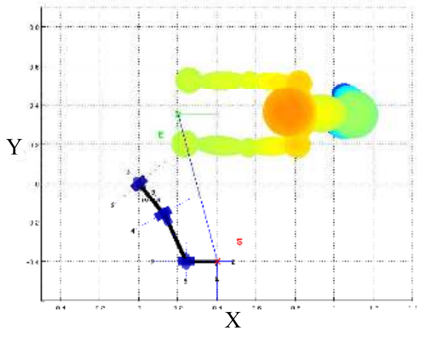


(b)

Figure 2.10: (a) The skeleton of a human subject from the Carnegie Mellon Motion Capture Database [68], and (b) the ellipsoid representation of the human worker based on the skeleton model.



(a)

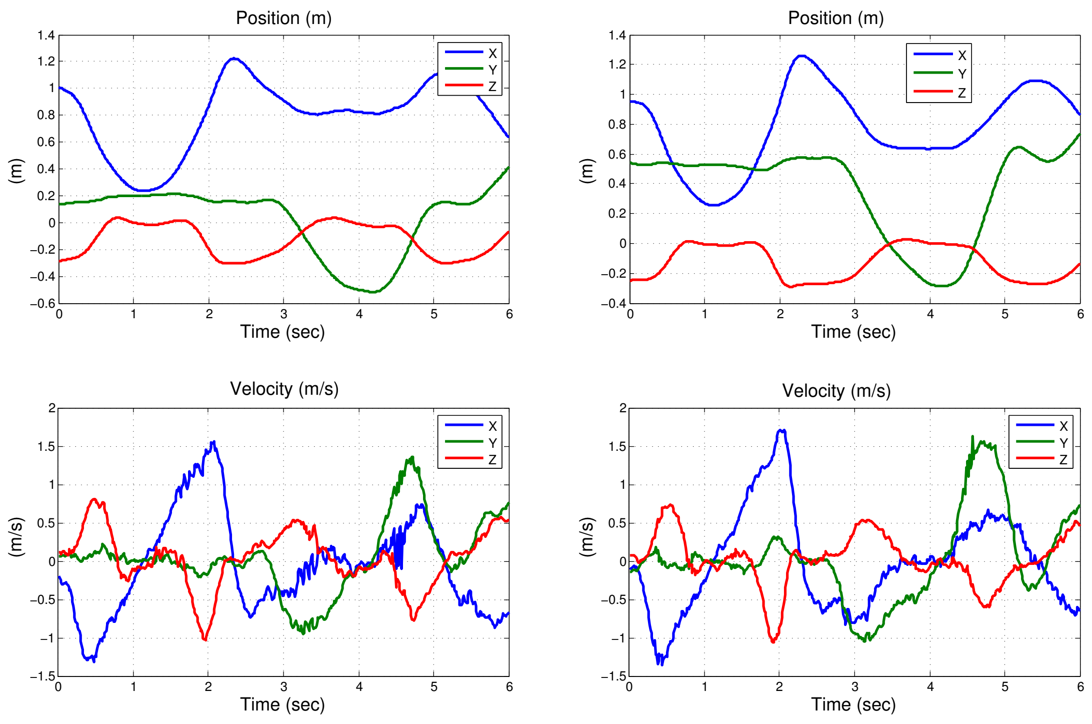


(b)

Figure 2.11: The snapshot of the human worker in a gesture which the worker's hands reach the farthest positions in simulations. (a) Isometric view, and (b) top views.

database and the corresponding ellipsoid representation. The human worker is chosen to be represented by twenty ellipsoids: three ellipsoids for each of legs, five ellipsoids for each arm, three ellipsoids for the torso, and one ellipsoid for the head. The radii of each ellipsoid are defined based on the skeleton model. For example, one radius of the ellipsoid representing the left thigh is half of the length of the link between L-hip joint and L-knee joint, and reasonable values are set to the other two radii of this ellipsoid.

Figure 2.11 shows the snapshot in isometric and top views of the human worker in a gesture in which his/her hands reach the farthest positions in his/her motion in simulations. The centers of ellipsoids representing the two fists in the gesture are located at



(a) (b)

Figure 2.12: Motions of the agent ellipsoids representing the two fists of the human worker. (a) Left fist, and (b) right fist.

$(0.255, 0.527, -0.007)$ and $(0.234, 0.201, -0.011)$. It shows that when the human worker is in this gesture, the two fists of the human worker are very close to the assembling position. The position and velocity of both fists are shown in Figure 2.12.

Decentralized Robot–Robot Collaboration

In decentralized robot–robot collaboration, the scenario of a master robot manipulator cooperating with a slave robot manipulator within close proximity is simulated. The task of the master robot manipulator has a higher priority while the task of the slave robot manipulator is less important or depends on the completeness of the master robot task. For instance, the slave robot manipulator is working on a pick-and-place task and the master robot manipulator is focusing on the assembling task on the workpiece provided by the slave robot. Because of the decentralized collaboration structure, it is possible that the slave robot is approaching the place-down spot while the master robot manipulator is still working at the same spot.

For the slave robot manipulator, the pick-up and place-down locations and the corresponding configurations in simulations remain the same as those defined in human–robot collaboration. For simplicity but without loss of generality, the same model of the robot

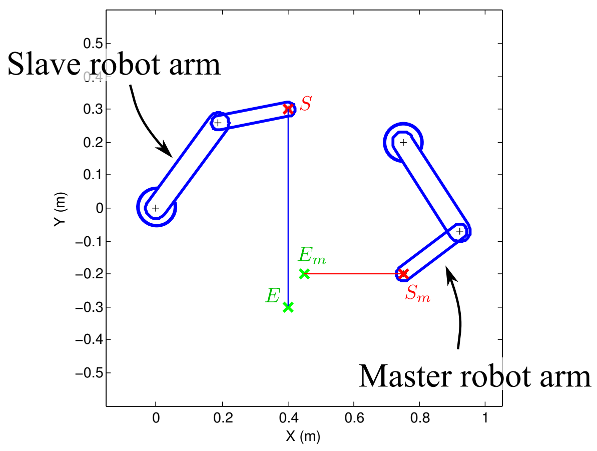


Figure 2.13: Initial condition in simulations of decentralized robot–robot collaboration for the case of two two-link planar robot manipulators. The points S_m, E_m, S, and E are the desired positions for the end-effectors of the master and the slave robot manipulators.

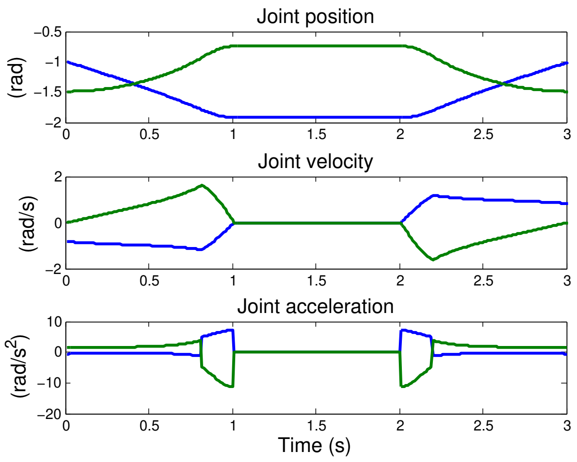
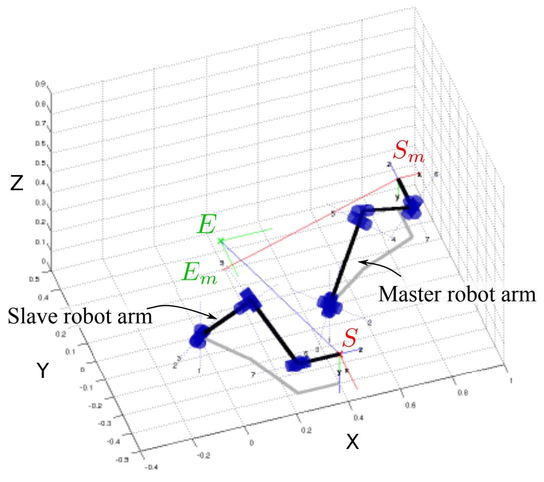


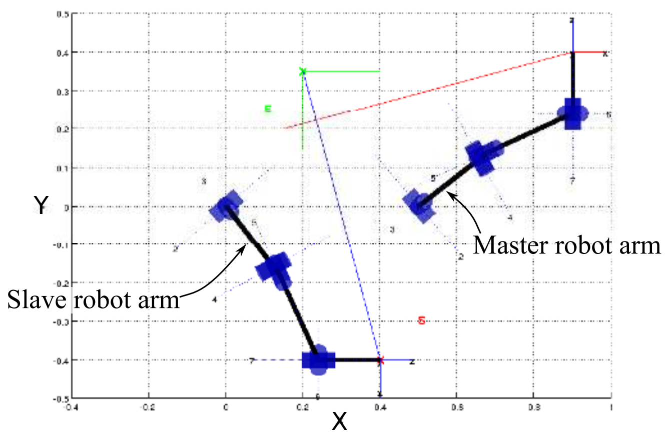
Figure 2.14: The trajectory of the master robot manipulator in the simulations of decentralized robot–robot collaboration for the case of the two-link planar robot.

manipulator is used for the master and the slave robot manipulators in the simulation study. The motion of the master robot manipulator is detailed below.

In the case of a two-link planar robot, the base of the master robot arm is located at $(0.75, 0.2)$ in the X–Y plane. Two predefined positions for the end-effector of the master robot arm are $(0.7517, -0.2)$ and $(0.45, -0.2)$, and denoted as S_m and E_m in Figure 2.13. The straight line in red between S_m and E_m is the path of the master robot end-effector. The master robot manipulator moves from S_m to E_m with a maximum task space velocity



(a)



(b)

Figure 2.15: Initial condition in simulations of decentralized robot–robot collaboration for the case of two ITRI seven-DOF robot manipulators. Points S_m, E_m, S, and E are the desired positions for the end-effectors of the master and the slave robot manipulators. (a) Isometric view, and (b) top view.

of 0.33 m/s on its end-effector, stays at E_m position for one second to finish its task, and then moves back to S_m with the same but mirrored motion. The reference trajectory of the master robot manipulator is shown in Figure 2.14.

In the case of ITRI seven-DOF robot manipulator, the base of the master robot manipulator is at $(0.5, 0, 0)$. The two predefined positions for the end-effector are $(0.9, 0.4, 0.15)$ and $(0.15, 0.2, 0.1)$, as shown in Figure 2.15 as S_m and E_m. In simulations, the master robot manipulator moves back and forth between these two positions and the maximum velocity in task space for the end-effector is 0.583 m/s. The reference trajectory of the master robot manipulator for one route, i.e., from S_m to E_m and then back to S_m, is shown in Figure 2.16.

2.4 Performance Assessment

To evaluate the effectiveness of the proposed algorithm in human–robot collaboration and decentralized robot–robot collaboration in simulations, three factors are considered.

Safety of the Agents

The guarantee of no collisions during the collaboration is the most significant factor and has the highest priority over other performance factors. An index to quantify the safety of the agents inside the shared workspace is designed and introduced in the next chapter. The safety

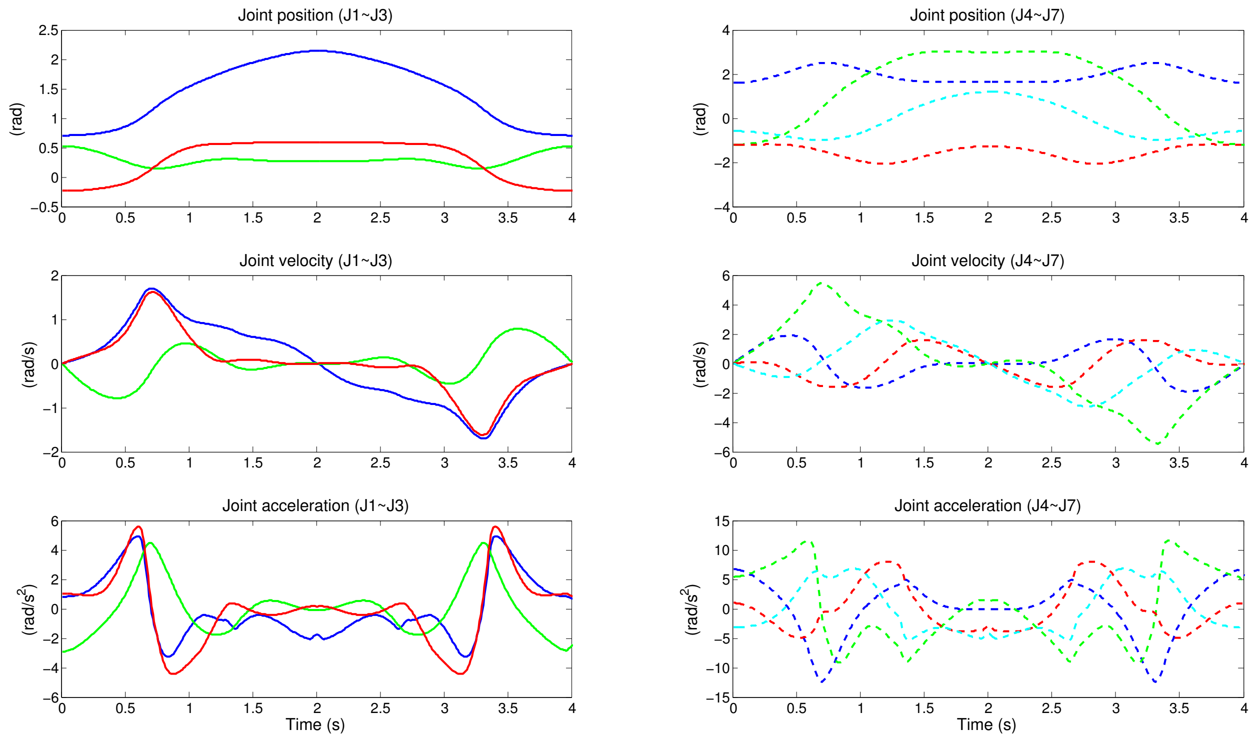


Figure 2.16: The reference trajectory of the master robot manipulator in simulations of decentralized robot–robot collaboration for the case of the ITRI seven-DOF robot. The solid blue, green, and red lines in the left column represent joint 1, joint 2, and joint 3. The dashed blue, green, red, and cyan lines in the right column represent the joint 4 to joint 7, respectively.

index is desired to stay inside the predefined safe interval during the entire collaboration in simulations.

Smoothness of Generated Trajectories

The smoothness of the generated trajectory is another important factor in the trajectory planning for the industrial robot manipulator. It is well-known that a trajectory with high values of joint jerk, which is the derivative of joint acceleration, increases the mechanical structure wear of the robot manipulator and deteriorates the tracking accuracy [40, 55, 28]. To avoid this, a smooth trajectory is normally defined by having a bounded joint jerk, or a continuous joint acceleration. In the case of human–robot collaboration, from the perspective of the human worker, a smooth trajectory also implies a more predictable motion of the robot manipulator and hence causes less mental stress to the human worker. The joint jerk of the generated trajectory, thus, becomes a factor to examine the proposed trajectory generation algorithm.

Feasibility for Real-Time Implementation

Since the trajectory for the robot manipulator is generated online to lead the manipulator toward the desired position and avoid any collisions, the feasibility of a real-time implementation of the proposed algorithm is important and needs to be considered.

2.5 Summary

In this chapter, the structure of the robot system for controlling the robot manipulator and handling the safety of the agents inside the shared workspace was proposed. The simulation environment, including the assumption of the robot perception, the robot manipulators used in simulations, and the scenarios, was introduced. The two robot manipulators are the two-link planar robot manipulator and the ITRI seven-DOF robot manipulator. The scenarios of human–robot collaboration and decentralized robot–robot collaboration are both simulated in the cases of two robot manipulators. Finally, the considerations of assessing the performance of the proposed algorithm were given.

Chapter 3

Design of Safety Index

3.1 Introduction

As aforementioned, a safety system in charge of the safety for the human worker and the robots is necessary in human–robot or robot–robot collaboration. A structure of the robot safety system was proposed in the previous chapter and assumptions were made about the agent position estimation block in the system. In this chapter, a measure, or a safety index, to assess the safety of the agents in the shared workspace is introduced. Firstly, a literature review about the safety index for the agents inside the shared workspace is given. Then, the design of the safety index for the agent ellipsoids, either representing the human worker or the master robot manipulator, inside the shared workspace is introduced. Two factors are taken into account: the distances between the robot manipulator and the set of the agent ellipsoids, and the momentum of the robot links toward the agent ellipsoids. Distance safety index (DSI) and momentum safety index (MSI) are designed based on these two factors, correspondingly. In order to decrease the computational load, ellipsoid coordinates are introduced in the design of the two safety indices. Two candidates are proposed to further combine the two safety indices into an overall safety index.

3.2 Concept and Related Work of the Safety Index

Preventing collisions with the agent inside the shared workspace is the most crucial issue in robot collaborations. For human–robot collaboration, standards and regulations as stated in Section 1.2 are the basic guidelines to follow. These guidelines, however, tend to conservatively limit either the maximum motion or the maximum power output of the robot manipulator so that the safety of the agent is guaranteed even when collisions occur. It substantially attenuates the performance of robot manipulators and fails to fully exploit the benefits of human–robot collaboration. In order to ensure the safety of the human worker in human–robot collaboration while minimizing the sacrifice of the robot manipulator performance, a measure is necessary to quantitatively indicate the safeness, or the dangerousness, of the human worker in the shared workspace in collaboration. The robot manipulator then can be controlled in the way that the performance is maximized while the measure stays within a predefined safe interval. For instance, when the entire robot manipulator is moving away from the human worker, it can be viewed as a less dangerous situation and the robot manipulator should be allowed to operate at a higher speed. In the scenario of decentralized robot–robot collaboration, the same measure can be applied to indicate the safeness of the master robot manipulator since the human robot and the master robot manipulator in the shared workspace are both treated as dynamic agents.

Many researchers have proposed different indices to quantitatively describe the safety of the agents in robot collaborations. In [25, 26, 52], Ikuta *et al.* proposed a danger index defined as a product of factors constructed by evaluating the potential impact force from the aspects of the mechanical design and the control of the robot manipulator. In the control aspect of the robot manipulator, the relative distance between the agent and the robot end-effector, the approaching velocity, and the momentum of the robot end-effector are considered. As a result, the danger index provides an assessment in the mechanical design and the control strategy of the robot manipulator so that the safety of the agent is guaranteed even when collisions occur.

Heinzmann *et al.* [23] designed a measure named impact potential which is defined as the maximum impact force that can be generated from a collision between a point of concern on the robot manipulator and a static obstacle. For each point of concern, an inequality constraint over the joint torque then can be obtained to restrict the command torque to the entire robot manipulator. In the work of Kulić and Croft [36, 35, 39, 38, 37], a danger index is estimated based on factors influencing the impact force during a collision, such as effective robot inertia, the relative velocity, and the relative distance.

All the measures proposed in above approaches are defined between a predefined point of concern on the robot manipulator and a point on the agent. When the entire robot manipulator is considered in the assessment of the safety or danger index, a normal method is to assign multiple points of concern on each link or searching for the point nearest to the agent. An analogous situation happens when the whole body of the human worker or the master robot manipulator are considered. As a result, these approaches encounter an exponential increase of the computational load.

In [41, 42, 72, 43], Lacevic *et al.* proposed the kinetostatic danger field which is built from the configuration and velocity of the robot arm, and the integral of the field along a robot link gives the danger index for the specific robot link. The kinetostatic danger field provides a danger index of the entire robot manipulator for every point in the workspace. In the practical implementation, extra computational power is still needed to find the closest point on the agents.

In this chapter, two safety indices, DSI and MSI, are proposed based on the distance from the robot manipulator to the agent and the robot links momentum to assess the safety of the agents. To suppress the exponentially increasing computational load due to the number of points of concern on the robot links or the search of the closest point on the agents, the safety indices are evaluated in the ellipsoid coordinates attached to the robot links.

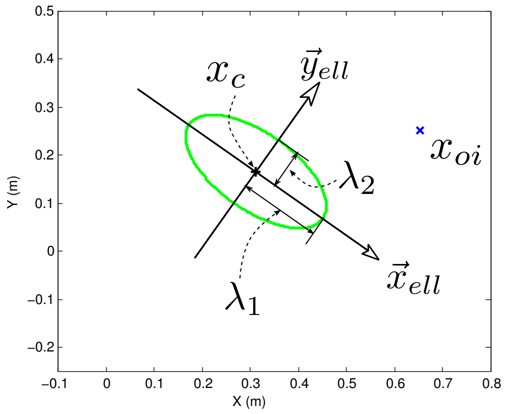


Figure 3.1: An example of an ellipsoid coordinate constructed on an ellipse in the X–Y plane.

3.3 The Design of the Distance Safety Index

Intuitively, the relative distance between the robot and the agent is a direct indicator of the safety for the agent. A shorter distance implies a higher chance of collisions and is less safe. Hence one candidate for the DSI is the reciprocal of the distance between a point of concern on the robot manipulator and a point on the agent model. A point of concern could be, for example, the robot end-effector or the robot joint. However, to relieve the computational load issue as aforementioned, the ellipsoid coordinate for each robot link is introduced.

3.3.1 The Ellipsoid Coordinates on Robot Links

An ellipsoid, $\mathcal{E}$, can be written in mathematical form as

$$\mathcal{E} = \{x \in \mathbb{R}^m \mid (x - x_c)^T Q (x - x_c) = 1\}, \tag{3.1}$$

where $x_c \in \mathbb{R}^m$ and $Q \in \mathbb{R}^{m \times m}$ are the center and the matrix of the ellipsoid, respectively. m is the dimension of the space where the ellipsoid exists. Q can be explicitly expressed as

$$Q = RA^{-1}R^T, \tag{3.2}$$

where

$$A = \begin{bmatrix} \lambda_1^2 & 0 & \cdots & 0 \\ 0 & \lambda_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_m^2 \end{bmatrix}. \tag{3.3}$$

$R \in \mathbb{R}^{m \times m}$ is an unitary matrix, and $A \in \mathbb{R}^{m \times m}$ is a diagonal matrix. The positive square roots of the diagonal elements of A, denoted as $\lambda_1, \ldots, \lambda_m$, are the radii of the ellipsoid.

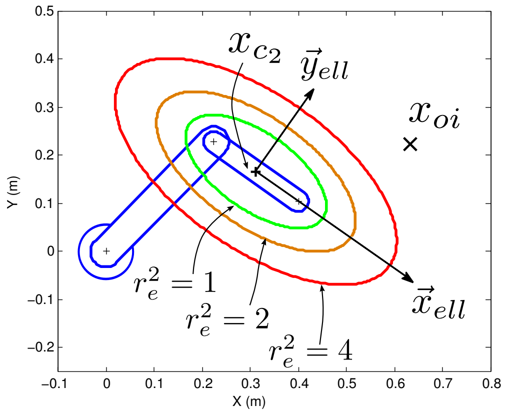


Figure 3.2: An example of an ellipsoid coordinate constructed on the second link of a two-link planar robot.

An ellipsoid coordinate then can be built upon the ellipsoid by selecting the ellipsoid center as the ellipsoid coordinate origin and aligning coordinate axes with ellipsoid axes. Figure 3.1 shows an example of an ellipsoid coordinate constructed on the green ellipse in the X–Y plane. The center, the major radius, and the minor radius of the ellipse are x_c, λ_1, and λ_2. The origin of the ellipsoid coordinate is then located at x_c and the ellipsoid coordinate axes, denoted as $\vec{x}_{ell}$ and $\vec{y}_{ell}$, are aligned with the major and the minor axes of the ellipse. A point, x_{oi}, is in the X–Y plane, and the distance in the ellipsoid coordinate from this point to x_c can be obtained by

$$d_{oic} = \left[(x_{oi} - x_c)^T Q (x_{oi} - x_c)\right]^{\frac{1}{2}}. \tag{3.4}$$

For each robot link, an ellipsoid coordinate is constructed by properly selecting the ellipsoid radii and the ellipsoid center in the local link coordinate so that each link is enclosed by a unit ellipsoid in the ellipsoid coordinate, and the ellipsoid coordinate moves along with the link. Figure 3.2 shows an example of the ellipsoid coordinate denoted by $\vec{x}_{ell}$ and $\vec{y}_{ell}$ attached on the second link of a two-link planar robot. The base of the robot arm is located at the origin of the X–Y plane. The ellipsoid coordinate origin is at x_{c_2}, which is the middle point of the second link, and the ellipsoid radii are 0.17 m and 0.08 m. Since two radii of the ellipsoid are different, the magnitude of unit vectors in the ellipsoid coordinate along $\vec{x}_{ell}$ and $\vec{y}_{ell}$ are different. The green, brown, and red contours in Figure 3.2 represent the squared distance r_e^2 of 1, 2, and 4 to x_{c_2} in the ellipsoid coordinate. The green contour with $r_e^2 = 1$ represents the unit ellipsoid. For instance, every point on the red contour is of the same distance to x_{c_2} in the ellipsoid coordinate, which is 2. Then the distance between a robot link and a point in the agent model, say x_{oi}, can be represented by the distance in the corresponding ellipsoid coordinate from x_{oi} to the ellipsoid coordinate origin. Thus, the squared distance from x_{oi} to each ellipsoid coordinate origin is computed as

$$d_{ji}^2 = (x_{oi} - x_{cj}(q))^T Q_j(q)(x_{oi} - x_{cj}(q)), \tag{3.5}$$

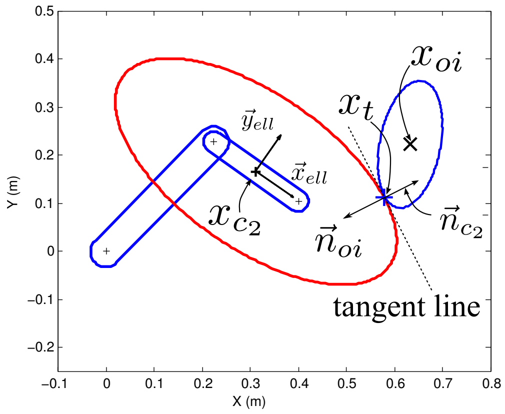


Figure 3.3: An example of an ellipsoid coordinate constructed on the second link of a two-link planar robot with an ellipse agent.

where $x_{cj}(q)$ and $Q_j(q)$ are the center and the matrix of the j-th ellipsoid coordinate attached to the j-th robot link, respectively. Since the ellipsoid coordinates move along with the robot links, the origin and the matrix of each ellipsoid coordinate are functions of the joint space position vector, q, where $q \in \mathbb{R}^{N_j}$ and N_j is the number of robot joints.

3.3.2 Ellipsoids in an Ellipsoid Coordinate

The agent model, however, is composed of ellipsoids rather than a set of points. The shortest squared distance from the surface of an agent ellipsoid to each ellipsoid coordinate origin needs to be computed. Figure 3.3 shows an ellipse which is centered at x_{oi} and represents a simple agent model in the same example case of the two-link planar robot. It is seen that the agent ellipsoid is tangent to the red contour at x_t, and hence, the shortest squared distance in the ellipsoid coordinate from the agent ellipsoid to the ellipsoid coordinate origin is 4. Note that three points, x_{c_2}, x_{oi}, and x_t, are not aligned. In fact, there is no analytical form solution for finding the tangent point in the general case.

Alternatively, the tangent point, x_t, and the shortest squared distance, d_{ji}^2, can be found via solving the optimization problem,

$$\begin{aligned} d_{ji}^2 = \min_{x_t} \quad & (x_t - x_{cj}(q))^T Q_j(q)(x_t - x_{cj}(q)), \\ s.t. \quad & (x_t - x_{oi})^T P_i (x_t - x_{oi}) = 1 \end{aligned} \tag{3.6}$$

where P_i is the matrix of the agent ellipsoid.

Instead of solving the optimization problem in (3.6) directly, the problem is transformed into a root-finding problem for a polynomial function. Firstly, it is given from the geometrical relationship that the normal vector passing x_t outward from the agent ellipsoid and another normal vector outward from the contour of the ellipsoid coordinate align with each other. These two vectors are denoted by $\vec{n}_{oi}$ and $\vec{n}_{c_2}$ as shown in Figure 3.3. The alignment between

$\vec{n}_{oi}$ and $\vec{n}_{c_2}$ can be written mathematically as

$$Q_j(q)(x_t - x_{cj}(q)) = tP_i(x_t - x_{oi}), \tag{3.7}$$

where t is a scalar variable whose physical meaning is the ratio of the magnitude between these two normal vectors. With some algebraic manipulation, it becomes,

$$x_t = (Q_j(q) - tP_i)^{-1}(Q_j(q)x_{cj}(q) - tP_i x_{oi}). \tag{3.8}$$

Hence x_t depends on t. Since x_t is always on the surface of the agent ellipsoid, a polynomial function, $f_d(t)$ can be obtained by plugging x_t in (3.8) into the expression of the agent ellipsoid as,

$$f_d(t) = (x_t - x_{oi})^T P_i (x_t - x_{oi}) - 1. \tag{3.9}$$

From the physical meaning of t, it can be shown that $f_d(t)$ has one real positive root and one real negative root. The negative one indicates the external tangent of the agent ellipsoid and the contour, and the positive one is for the internal tangent. The negative root of $f_d(t)$ is the t we are looking for, and x_t and d_{ji}^2 can be computed after t is found.

3.3.3 The Definition of the Distance Safety Index

With the introduction of the ellipsoid coordinate, the distance safety index for the pair of (j, i), the j-th ellipsoid coordinate attached on the robot link and the i-th ellipsoid in the agent model, can be designed as

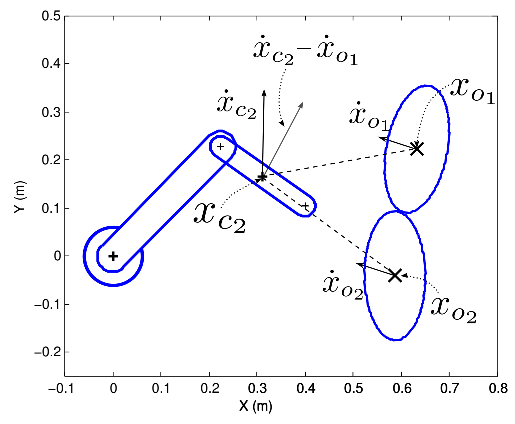
$$DSI_{ji}(q) = \frac{1}{d_{ji}^2}. \tag{3.10}$$

3.4 The Design of the Momentum Safety Index

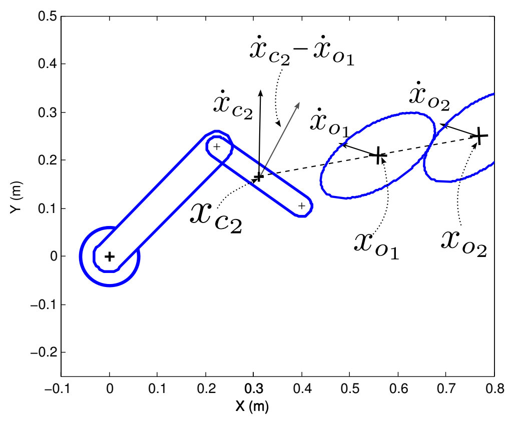
Robots with larger momentum normally require a longer time and a longer distance to fully stop the motion after the stop command is issued. Hence the momentum of the robot links is also a factor need to be considered. The momentum safety index is designed to represent the linear momentum of the robot links computed from the origins of the corresponding ellipsoid coordinates toward the direction to the agents. This can be obtained by taking the projection of the linear momentum vector onto the vector from the origin of the ellipsoid coordinate pointing to the center of the agent ellipsoid as,

$$p_{ji}(q, \dot{q}) = M_j(\dot{x}_{cj}(q, \dot{q}) - \dot{x}_{oi})^T \frac{x_{oi} - x_{cj}(q)}{\|x_{oi} - x_{cj}(q)\|_2}, \tag{3.11}$$

where M_j and $\dot{x}_{cj}(q, \dot{q})$ are the mass of the j-th link and the linear velocity of the j-th ellipsoid coordinate origin in task space. The linear velocity $\dot{x}_{cj}(q, \dot{q})$ is a function of the joint position and velocity of the robot manipulator because the ellipsoid coordinate is attached and moving



(a)



(b)

Figure 3.4: Examples of the momentum safety index (MSI) in the case of a two-link planar robot with two agent ellipses which have the same motion, i.e., $\dot{x}_{o_1} = \dot{x}_{o_2}$, but are at different positions. All dashed lines connect the origin of the ellipsoid coordinate with the agent ellipse centers. (a) The angles between the relative motion vector and two dashed lines for two agent ellipses are different. (b) The angles are the same for two ellipses, but the two ellipses are in the different distances to the origin of the ellipsoid coordinate.

with the robot link. $\dot{x}_{oi}$ is the linear velocity in task space of the center of the i-th agent ellipsoid. The MSI for the pair of (j, i) is defined based on $p_{ji}(q, \dot{q})$ as,

$$MSI_{ji}(q, \dot{q}) = \frac{1}{d_{ji}^2}\Big[\max\Big\{p_{ji}(q, \dot{q}), 0\Big\}\Big]^2. \tag{3.12}$$

In other words, $MSI_{ji}(q, \dot{q})$ can be rewritten as

$$MSI_{ji}(q, \dot{q}) = \max\Big\{\text{sign}\big(p_{ji}(q, \dot{q})\big)\Big(\frac{p_{ji}(q, \dot{q})}{d_{ji}}\Big)^2, 0\Big\}. \tag{3.13}$$

The maximization in (3.12) indicates that only the case when the link moves toward the agent model is considered in the MSI. Once the angle between the relative motion direction, $(\dot{x}_{cj}(q, \dot{q}) - \dot{x}_{oi})$, and the vector pointing from the origin of the ellipsoid coordinate to the agent ellipsoid center, $(x_{oi} - x_{cj}(q))$, is larger than or equal to $\frac{\pi}{2}$, it is considered as a safe situation and the momentum safety index for this pair is zero. Figure 3.4a is an example showing the concept. Two agent ellipsoids exist in the workspace of a two-link planar robot, and both of them have the same velocity, i.e., $\dot{x}_{o_1}$ is equal to $\dot{x}_{o_2}$. These two agent ellipsoids could be, for instance, a part of a human arm or a link of the master robot arm. In this scenario, only the MSI_{21} is greater than zero since the angle between $(\dot{x}_{c_2} - \dot{x}_{o_1})$ and the vector $(x_{o_1} - x_{c_2})$ is less than $\frac{\pi}{2}$. On the other hand, the MSI_{22} is zero because of the obtuse angle between $(\dot{x}_{c_2} - \dot{x}_{o_1})$ and the vector $(x_{o_2} - x_{c_2})$.

The denominator, d_{ji}^2, the squared distance in the ellipsoid coordinate, is used to differentiate the MSI between cases that a robot has the same magnitude of linear velocity at the ellipsoid coordinate origins but is in a different relative positions to an agent. A robot link that has a high momentum toward an agent but still far away should be considered as less dangerous. Figure 3.4b is another example. In this example, the linear momenta of the second link toward the two agent ellipsoids, p_{21} and p_{22}, are identical, but the MSI_{21} is larger than MSI_{22} since the first agent ellipsoid is closer to the robot link. The squared term and d_{ji}^2 in (3.12) makes the MSI_{ji} quadratic in form analogous to the squared distance in the DSI_{ji}.

3.5 The Overall Safety Index

In Section 3.3 and Section 3.4, the distance safety index and the momentum safety index are defined for the pair of (j, i), the j-th ellipsoid coordinate attached to the robot link and the i-th agent ellipsoid. It can be observed that they have the nature of a smaller value indicating a safer situation. Hence the overall safety index should be designed by combining all DSI_{ji}'s and MSI_{ji}'s and then selecting the maximum value among all pairs of (j, i) to capture the worst-case scenario. Two candidates of the overall safety index are defined below.

Candidate 1:

$$SI(q, \dot{q}) = \max_{j,i} \Big\{ k_{dsi} DSI_{ji}(q) + k_{msi} MSI_{ji}(q, \dot{q}) \Big\}. \tag{3.14}$$

The weightings, k_{dsi} and k_{msi}, are chosen as the reciprocal of the maximum acceptable distance safety index and momentum safety index by the Bryson-like rule [5] for the purpose of normalizing the DSI_{ji} and the MSI_{ji}:

$$k_{dsi} = \frac{1}{S_{ad}}, \quad k_{msi} = \frac{1}{S_{am}}, \tag{3.15}$$

where S_{ad} and S_{am} are the maximum acceptable distance safety index and momentum safety index, respectively. This candidate of the overall safety index is taking the linear combination of DSI_{ji} and MSI_{ji} into consideration, and selecting the maximum value of the linear combination among all pairs of (j, i).

Candidate 2:

$$SI(q, \dot{q}) = \max_{j,i} \Big\{ \max \big\{ k_{dsi} DSI_{ji}(q),\ k_{msi} MSI_{ji}(q, \dot{q}) \big\} \Big\}, \tag{3.16}$$

where k_{dsi} and k_{msi} are defined in the same way as in (3.15) in the first candidate. For this candidate, the DSI_{ji} and MSI_{ji} are considered independently without any combination,

Table 3.1: Parameters of the ellipsoid coordinates in the two-link planar robot manipulator in simulations.

	Link	Ellipsoid coordinate origin on each link coordinate	Unit ellipsoid radii	
			λ_1	λ_2
Ellipsoid coordinate 1	1	$(0.16, 0)$	0.22	0.1
Ellipsoid coordinate 2	2	$(0.1075, 0)$	0.17	0.08
(unit)		(m)		

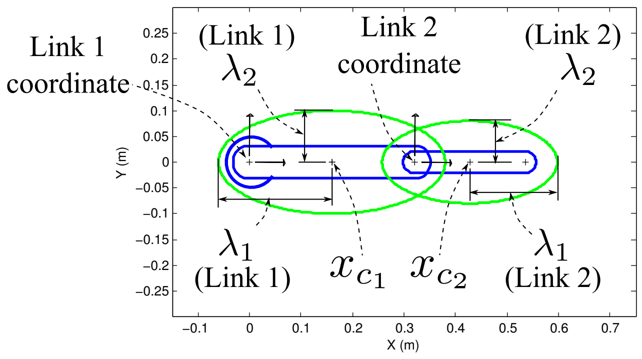


Figure 3.5: The ellipsoid coordinates constructed on the two-link planar robot manipulator in simulations.

and the overall safety index is the maximum value among all DSI_{ji}'s and MSI_{ji}'s. For the definition, we can find that the second candidate is less conservative than the first candidate.

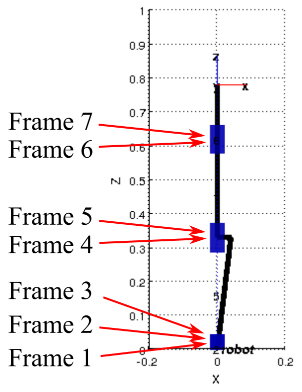
3.6 Ellipsoid Coordinates of the Two Robot Manipulators

Table 3.1 and Figure 3.5 show the two ellipsoid coordinates constructed on the two-link planar robot for algorithm validation in simulations. The origins of the two ellipsoid coordinates are both at the centers of the robot links. The two radii of the unit ellipsoid in each coordinate are chosen reasonably so that each unit ellipsoid can enclose the corresponding link.

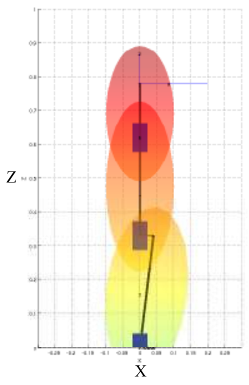
Table 3.2 and Figure 3.6 show the ellipsoid coordinates for the ITRI seven-DOF robot manipulator. It can be observed from Figure 2.4 that the rotation axis of joint 5 is parallel to link 4 and link 5, and hence the space occupied by these two links remains the same during the rotation of joint 5. The same observations can be made for joint 3 and joint 7,

Table 3.2: Parameters of the ellipsoid coordinates on the ITRI seven-DOF robot in simulations.

	Link	Ellipsoid coordinate origin on each link coordinate	Unit ellipsoid radii		
			λ_1	λ_2	λ_3
Ellipsoid coordinate 1	2	(0,0,0.165)	0.12	0.12	0.25
Ellipsoid coordinate 2	4	(0,0,0,145)	0.10	0.10	0.25
Ellipsoid coordinate 3	6	(0,0,0.07)	0.10	0.10	0.20
(unit)		(m)			



(a)



(b)

Figure 3.6: The ellipsoid coordinates constructed on the ITRI seven-DOF robot manipulator.

and the links connected to them. With these properties, only three ellipsoid coordinates are necessary for evaluating the safety index. The centers of the three ellipsoid coordinates attached to link 2, link 4, and link 6 are selected reasonably with their radii set accordingly. The unit ellipsoids of the three coordinates are shown in Figure 3.6.

3.7 Summary

In this chapter, the design of the quantitative measure, called safety index, was given in order to assess the safety of the agent in the shared workspace. The concept of the safety index was explained and related works were cited. In the design of the proposed safety index, two factors are considered: the relative distances between the robot manipulator and the agent, and the linear momentum of the robot links toward the agent. The distance safety index and the momentum safety index are constructed correspondingly based on the two factors. In the design of the two safety indices, the ellipsoid coordinates were introduced for the purpose of decreasing the computational load. An overall safety index can then be constructed from the linear combination of the distance safety index and momentum safety index. Two candidates of the overall safety index were proposed. In the last, the setup of ellipsoid coordinates in the two testbed robot manipulators were given.

Chapter 4

Online Trajectory Generation Algorithm

4.1 Introduction

In this chapter, the proposed online trajectory generation algorithm is stated in detail. This algorithm is developed under the optimization-based path planning framework with the use of receding horizon strategy, discretized kinematics model, and the safety index. By solving the optimization problem, the solution is a trajectory within a short time horizon which achieves the functions of leading the robot manipulator toward the goal position and avoiding any collisions with agents simultaneously. The formulated optimization problem, because of the transformation of variables between joint space and task space, is highly nonlinear and non-convex. In order to achieve online trajectory generation, the formulated optimization problem is approximated by a quadratic formulation. To further attenuate the computational load, a heuristic method is proposed to select the inequality constraints which are likely to become active in the next iteration. Simulations of two scenarios, human–robot collaboration and decentralized robot–robot collaboration, are conducted on the two robot manipulators for validation.

4.2 Dynamic Model of Agent Ellipsoids

In Section 2.3, an assumption was made about the sensor block being capable of providing position measurements of the agents in every $T_c\Delta t$ second. The Δt is the robot controller sampling time, and T_c is a positive integer. The agent position estimation block then transforms the position measurements into a set of agent ellipsoids and outputs the information of the agent ellipsoids. The information of agent ellipsoids includes the task space positions and velocities of the agent ellipsoid centers and the matrices of the agent ellipsoids accounting for the orientation of agent ellipsoids. This is the necessary information for evaluating the safety index between an agent ellipsoid and an ellipsoid coordinate attached on a robot link.

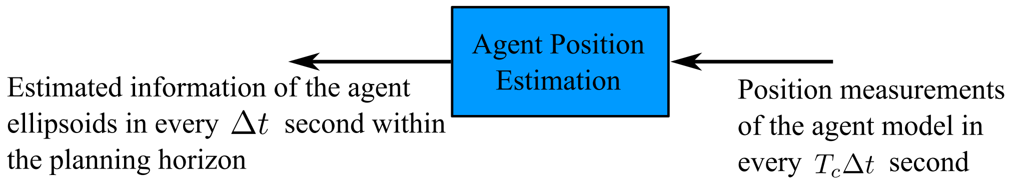

Agent Position Estimation
Estimated information of the agent ellipsoids in every Δt second within the planning horizon
Position measurements of the agent model in every $T_c\Delta t$ second

Figure 4.1: The input and output of the agent position estimation block. Δt is the robot controller sampling time and position measurements are assumed to be available for every $T_c\Delta t$ second. The information of agent ellipsoids includes the task space positions and velocities of the agent ellipsoid centers and the matrices of the agent ellipsoids.

In the proposed online trajectory generation algorithm, the safety index of the future time steps within a horizon are taken into consideration. Thus, the agent position estimation block needs not only to transform the current sensor measurements into the information of agent ellipsoids, but also estimate the information of agent ellipsoids in the future and within the planning horizon. The input and output of the agent position estimation block is shown schematically in Figure 4.1.

In this thesis, a simple method is used to estimate the information of the agent ellipsoids within the planning time horizon in both human–robot collaboration and decentralized robot–robot collaboration. This method in the two robot collaborations are detailed below.

4.2.1 Dynamic Model of Agent Ellipsoids in Human–Robot Collaboration

In human–robot collaboration, since the position measurements from the sensor block are position measurements of task space, a simple and intuitive way is to utilize the constant-velocity model to estimate the agent ellipsoids information. In other words, the task space positions of agent ellipsoid centers transformed from the current and previous sensor measurements are used to compute the velocity of the agent ellipsoids. For instance, the velocity of the i-th agent ellipsoid in the k-th time step can be computed by

$$v_{oi(k)} = \frac{1}{T_c\Delta t}\big(x_{oi(k)} - x_{oi(k-T_c)}\big), \tag{4.1}$$

where $x_{oi(k)}$ and $x_{oi(k-T_c)}$ are the task space positions of the i-th agent ellipsoid center transformed from the current and previous sensor measurements. Δt is the controller sampling time. Using the velocity $v_{oi(k)}$ and the current position $x_{oi(k)}$, the positions of the i-th agent ellipsoid after the k-th time step can be estimated by

$$x_{oi(k+l)} = x_{oi(k)} + (l\Delta t)v_{oi(k)}, \quad l = 0, \dots, N, \tag{4.2}$$

where l is a non-negative integer and is less than or equal to N, and N is the predefined planning time horizon. In the constant-velocity model, the rotations of the agent ellipsoids are ignored, and hence the matrices of agent ellipsoids $P_{i(k)}$'s remain the same for all k within the time horizon, i.e.,

$$P_{i(k+l)} = P_{i(k)}, \quad l = 0, \ldots, N. \tag{4.3}$$

For the radii of the agent ellipsoids, the same radii are used as well.

4.2.2 Dynamic Model of Agent Ellipsoids in Decentralized Robot–Robot Collaboration

In decentralized robot–robot collaboration, the constant-velocity method is utilized in joint space since the position measurements of the master robot manipulator are the joint space positions. The joint velocity of the master robot at the k-th time step then is estimated as

$$\dot{q}_{m(k)} = \frac{1}{T_c \Delta t}(q_{m(k)} - q_{m(k-T_c)}), \tag{4.4}$$

where $q_{m(k)}$ and $q_{m(k-T_c)}$ are the joint positions of the master robot manipulator at the k-th and $(k - T_c)$-th time step, and $\dot{q}_{m(k)}$ is the estimate joint velocity of the master robot manipulator. The estimated joint position within the time horizon can be obtained as

$$q_{m(k+l)} = q_{m(k)} + (l\Delta t)\dot{q}_{m(k)}, \quad , l = 0, \ldots, N. \tag{4.5}$$

Then, the information of the agent ellipsoids for each time step within the planning horizon, N, can be computed based on the estimated joint position of the master robot.

4.3 Formulation of the Optimization Problem

4.3.1 Discretized Kinematics Model

In order to generate a trajectory rather than just a geometrical path, the joint position, joint velocity, and joint acceleration need to be included as variables in the optimization problem. To maintain the kinematic relationships among the joint position, velocity, and acceleration, the robot kinematic model is introduced as an equality constraint in the optimization problem. The kinematic model can be written in a state-space form as

$$\frac{d}{dt}\begin{bmatrix} q_{(t)} \\ \dot{q}_{(t)} \end{bmatrix} = \begin{bmatrix} 0 & I_{N_j} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} q_{(t)} \\ \dot{q}_{(t)} \end{bmatrix} + \begin{bmatrix} 0 \\ I_{N_j} \end{bmatrix} \ddot{q}_{(t)}, \tag{4.6}$$

where $q_{(t)}$, $\dot{q}_{(t)}$, and $\ddot{q}_{(t)}$ are the joint position, velocity, and acceleration in continuous time, respectively. All of them are vectors with length of N_j, where N_j is the number of joints in the robot manipulator. $I_{N_j} \in \mathbb{R}^{N_j \times N_j}$ is the identity matrix. Discretizing (4.6) yields

$$\begin{bmatrix} q_{(k+1)} \\ \dot{q}_{(k+1)} \end{bmatrix} = \begin{bmatrix} I_{N_j} & (\Delta t)I_{N_j} \\ 0 & I_{N_j} \end{bmatrix} \begin{bmatrix} q_{(k)} \\ \dot{q}_{(k)} \end{bmatrix} + \begin{bmatrix} \frac{1}{2}(\Delta t)^2 I_{N_j} \\ (\Delta t)I_{N_j} \end{bmatrix} \ddot{q}_{(k)}, \tag{4.7}$$

where Δt is the controller sampling time. The discretized kinematic model in (4.7) can be rewritten in $z_{(k)}$ as,

$$z_{(k+1)} = A_d\, z_{(k)} + B_d\, u_{(k)}, \tag{4.8}$$

where

$$\begin{aligned} z_{(k)} &= \begin{bmatrix} q_{(k)} \\ \dot{q}_{(k)} \end{bmatrix} \in \mathbb{R}^{2N_j}, \\ u_{(k)} &= \ddot{q}_{(k)} \in \mathbb{R}^{N_j}, \\ A_d &= \begin{bmatrix} I_{N_j} & (\Delta t) I_{N_j} \\ 0 & I_{N_j} \end{bmatrix} \in \mathbb{R}^{2N_j \times 2N_j}, \text{ and} \\ B_d &= \begin{bmatrix} \frac{1}{2}(\Delta t)^2 I_{N_j} \\ (\Delta t) I_{N_j} \end{bmatrix} \in \mathbb{R}^{2N_j \times N_j}. \end{aligned} \tag{4.9}$$

4.3.2 Formulation

The concept of optimization-based path planning is to generate the trajectory by solving an optimization problem. As mentioned there are two objectives for trajectories generated by the proposed online trajectory generation algorithm. The first is that the generated trajectory leads the robot manipulator to reach its goal position and avoids collisions with the agent ellipsoids simultaneously. The second objective is that the generated trajectory is smooth. A smooth trajectory is normally defined by having a bounded joint jerk, or a continuous joint acceleration. Thus, the optimization problem needs to be formulated to achieve these two objectives.

The proposed formulation of the optimization problem is

$$\min_{U_k, Z_k} \quad k_p J_p(z_{(N)}) + k_u \sum_{k=0}^{N-1} J_u(u_{(k)}, u_{(k-1)}) \tag{4.10a}$$

$$s.t. \quad z_{(k+1)} = A_d z_{(k)} + B_d u_{(k)},\ k = 0, \ldots, (N-1) \tag{4.10b}$$

$$SI(z_{(k)}) \le S_a,\ k = 1, \ldots, N \tag{4.10c}$$

$$\underline{u} \le u_{(k)} \le \overline{u},\ k = 0, \ldots, (N-1) \tag{4.10d}$$

$$\underline{z} \le z_{(k)} \le \overline{z},\ k = 1, \ldots, N \tag{4.10e}$$

where

$$\begin{aligned} J_p(z_{(N)}) = &\big(x_e(z_{(N)}) - x_g\big)^T \big(x_e(z_{(N)}) - x_g\big) \\ &+ k_{pns}\big(x_{ne}(z_{(N)}) - x_{gne}\big)^T \big(x_{ne}(z_{(N)}) - x_{gne}\big) \\ &+ k_{pns}\big(x_{se}(z_{(N)}) - x_{gse}\big)^T \big(x_{se}(z_{(N)}) - x_{gse}\big) \end{aligned} \tag{4.11}$$

$$J_u(u_{(k)}, u_{(k-1)}) = \big(u_{(k)} - u_{(k-1)}\big)^T \big(u_{(k)} - u_{(k-1)}\big) \tag{4.12}$$

$$U_k = \{u_{(0)}, \ldots, u_{(N-1)}\} \tag{4.13}$$

$$Z_k = \{z_{(1)}, \ldots, z_{(N)}\} \tag{4.14}$$

Variables of the optimization problem are two sets, U_k and Z_k, which include all joint positions, $q_{(k)}$, joint velocities, $\dot{q}_{(k)}$, and joint accelerations, $\ddot{q}_{(k)}$, within the time horizon of N. The subscript k in parentheses indicates the time index. For simplicity, $z_{(k)}$ is used to represent the stack of $q_{(k)}$ and $\dot{q}_{(k)}$, and $u_{(k)}$ is $\ddot{q}_{(k)}$ as shown in (4.9).

The equality constraints in (4.10b) are the discretized kinematic model of the robot manipulator for the purpose of enforcing the relationships among the joint position, velocity, and acceleration. The inequality constraint on the safety index in (4.10c) guarantees that no collisions will occur along the generated trajectory, and S_a is a predefined safety threshold. Equations (4.10d) and (4.10e) enforce the bounds on the inputs and the states, respectively, where $\underline{u}$ and $\overline{u}$ are the lower and upper bounds of $u_{(k)}$, and $\underline{z}$ and $\overline{z}$ are the lower and upper bounds of $z_{(k)}$.

The objective function is composed of $J_p(z_{(N)})$ and $J_u(u_{(k)}, u_{(k-1)})$ which are weighed by k_p and k_u, and elaborated in (4.11) and (4.12). In (4.11), $x_e(z_{(N)})$ and x_g are the task space positions of the end-effector at the last step of the planning horizon and at the goal configuration, respectively. $x_{ne}(z_{(N)})$ and x_{gne} are the unit vectors in task space starting from $x_e(z_{(N)})$ and x_g along the normal directions of the end-effector coordinate at configuration of $z_{(N)}$ and at the goal configuration, respectively. $x_{se}(z_{(N)})$ and x_{gse} are the unit vectors defined in the same way for the sliding directions. The first term in $J_p(z_{(N)})$ penalizes the squared distance in task space from the end-effector at the last step of the planning horizon to the final goal position. The last two terms are to penalize the deviation of the end-effector orientation from the configuration of $z_{(N)}$ to the goal configuration. The constant k_{pns} is the weighting factor for the penalty of the last two terms. Note that these three terms are formulated in quadratic form of the task space vectors, instead of the joint space position vector. In other words, $J_p(z_{(N)})$ in (4.11) is designed to drive the robot manipulator by the deviation of the position and orientation of the end-effector between $z_{(N)}$ and the goal configuration. $J_u(u_{(k)}, u_{(k-1)})$ in (4.12) is designed to penalize the difference of the consecutive joint accelerations, $u_{(k)}$ and $u_{(k-1)}$, for a smooth trajectory.

This formulation results in an optimization problem with $3NN_j$ variables, $2NN_j$ equality constraints accounting for kinematics relationships among problem variables, N inequality constraints for the safety, and bound constraints on all variables.

4.4 Approximation

With a closer look, one can find that positions and velocities in both joint space and task space are involved in the formulation of the optimization problem in (4.10). The variables of the optimization problem are in joint space, but $J_p(z_{(N)})$ and $SI(z_{(k)})$ are designed based on the task space positions because of practical issues. For example, the goal position and the end-effector orientation are more likely to be designed in task space on assembly lines. For human–robot collaboration, the positions of human workers are also available directly in task space. As a result, computing $J_p(z_{(N)})$ and $SI(z_{(k)})$ requires the forward kinematics and forward differential kinematics to transfer the joint space positions and velocities of the robot manipulator to the task space positions and velocities. These two transformations, however, are highly nonlinear and non-convex with respect to the joint position, and hence makes the formulation in (4.10) nonlinear and non-convex. Solving such a problem is computationally expensive and therefore is difficult to be implemented in a online fashion.

To overcome this, approximations for $J_p(z_{(N)})$ and $SI(z_{(k)})$ are proposed. These are in a quadratic form and an affine form in joint space respectively. With the approximation, the original optimization formulation in (4.10) becomes a convex optimization formulation with a quadratic objective function and linear constraints. The approximation of $J_p(z_{(N)})$ and $SI(z_{(k)})$ are detailed below.

4.4.1 Approximation of $SI(z_{(k)})$

In Section 3.5, two candidates of the overall safety index are proposed, and both provide a way to combine the distance safety index and the momentum safety index. The approximation of the two candidates are similar and presented in the following.

Candidate 1:

Based on the definition of candidate 1 of the safety index in (3.14), equation (4.10c) implies that the maximum of the safety index among all pairs of (j, i) has to be less than or equal to S_a. This is equivalent to that the safety index of all pairs have to be less than or equal to S_a as,

$$k_{dsi}DSI_{ji}(z_{(k)}) + k_{msi}MSI_{ji}(z_{(k)}) \leq S_a, \quad \forall k, j, i. \tag{4.15}$$

From the definition of $MSI_{ji}(z_{(k)})$ in (3.12), we can find that the value of $MSI_{ji}(z_{(k)})$ depends on the sign of $p_{ji}(z_{(k)})$. Hence, the inequality in (4.15) is equivalent to two inequalities as,

$$k_{dsi}DSI_{ji}(z_{(k)}) \leq S_a, \ \forall k, j, i, \tag{4.16a}$$

$$k_{dsi}DSI_{ji}(z_{(k)}) + k_{msi}\overline{MSI}_{ji}(z_{(k)}) \leq S_a, \ \forall k, j, i, \tag{4.16b}$$

where

$$\overline{MSI}_{ji}(z_{(k)}) = \operatorname{sign}\big(p_{ji}(z_{(k)})\big)\frac{p_{ji}^2(z_{(k)})}{d_{ji}^2}. \tag{4.17}$$

The equivalence between (4.15) and (4.16) means that the feasible sets of them are identical. For instance, when $p_{ji}(z_k)$ is negative, the left-hand side of (4.16b) is always less than that of (4.16a), and hence the feasible set of (4.16) is controlled by (4.16a). A negative $p_{ji}(z_k)$ also implies that $MSI_{ji}(z_k)$ is zero, and thus (4.15) is equivalent to (4.16a). The same conclusion can be drawn for the case of a positive $p_{ji}(z_k)$.

Candidate 2:

Similarly, from the definition of candidate 2 of the safety index in (3.16), equation (4.10c) can be equivalently substituted by the inequalities of $DSI_{ji}(z_{(k)})$ and $MSI_{ji}(z_{(k)})$ among all

(j, i, k) less than or equal to S_a, as shown below.

$$k_{dsi} DSI_{ji}(z_{(k)}) \leq S_a, \ \forall k, j, i \tag{4.18a}$$

$$k_{msi} MSI_{ji}(z_{(k)}) \leq S_a, \ \forall k, j, i. \tag{4.18b}$$

The predefined safety threshold, S_a, is always positive, so the $MSI_{ji}(z_{(k)})$ in (4.18b) can be replaced by $\overline{MSI}_{ji}(z_{(k)})$. Equation (4.18) then becomes

$$k_{dsi} DSI_{ji}(z_{(k)}) \leq S_a, \ \forall k, j, i \tag{4.19a}$$

$$k_{msi} \overline{MSI}_{ji}(z_{(k)}) \leq S_a, \ \forall k, j, i. \tag{4.19b}$$

Taylor Series Expansion

The overall safety index in (4.16) and (4.19) is the equivalent form of (4.10c) in the perspective of having the same feasible set. Equation (4.16), however, needs to be evaluated for every triple of (j, i, k), where j is the index of the ellipsoid coordinate, i is the index of the agent ellipsoid, and k is the time index within the horizon, N. Namely, the number of the inequality constraints accounting for safety increases from N to $2N_eN_oN$, where N_e is the number of ellipsoid coordinates on the robot manipulator, and N_o is the size of agent ellipsoids.

The left-hand sides of (4.16) and (4.19) can then be approximated by the first-order Taylor expansion around z_k^0 as

$$c_{akji} + m_{akji}^T (z_{(k)} - z_k^0) \leq S_a, \ \forall k, j, i. \tag{4.20a}$$

$$c_{bkji} + m_{bkji}^T (z_{(k)} - z_k^0) \leq S_a, \ \forall k, j, i, \tag{4.20b}$$

where (4.20a) represents the first-order Taylor expansion of (4.16a) and (4.19a) around z_k^0, and (4.20b) is for (4.16b) and (4.19b). The vector z_k^0 is the initial guess value of $z_{(k)}$ and defined in the same way as $z_{(k)}$ as $z_k^0 = [q_k^{0T}, \dot{q}_k^{0T}]^T$. c_{akji} and c_{bkji} are the constant terms of the first-order Taylor expansion around z_k^0, and m_{akji} and m_{bkji} are the gradient vectors of the left-hand side of (4.16) and (4.19) evaluated at z_k^0. The c_{akji} and m_{akji} for both candidates are the same, but c_{bkji} and m_{bkji}, on the other hand, are different. All of them are shown below,

Candidate 1:

$$c_{akji} = k_{dsi} DSI_{ji}(z_k^0), \tag{4.21a}$$

$$m_{akji} = k_{dsi} \left. \frac{\partial DSI_{ji}(z_{(k)})}{\partial z_{(k)}} \right|_{z_k^0}, \tag{4.21b}$$

$$c_{bkji} = k_{dsi} DSI_{ji}(z_k^0) + k_{msi} \overline{MSI}_{ji}(z_k^0), \tag{4.21c}$$

$$m_{bkji} = k_{dsi} \left. \frac{\partial DSI_{ji}(z_{(k)})}{\partial z_{(k)}} \right|_{z_k^0} + k_{msi} \left. \frac{\partial \overline{MSI}_{ji}(z_{(k)})}{\partial z_{(k)}} \right|_{z_k^0}. \tag{4.21d}$$

Candidate 2:

$$c_{akji} = k_{dsi} DSI_{ji}(z_k^0), \tag{4.22a}$$

$$m_{akji} = k_{dsi} \frac{\partial DSI_{ji}(z_{(k)})}{\partial z_{(k)}}\bigg|_{z_k^0}. \tag{4.22b}$$

$$c_{bkji} = k_{msi} \overline{MSI}_{ji}(z_k^0), \tag{4.22c}$$

$$m_{bkji} = k_{msi} \frac{\partial \overline{MSI}_{ji}(z_{(k)})}{\partial z_{(k)}}\bigg|_{z_k^0}. \tag{4.22d}$$

Because of the complexity of the DSI_{ji} and MSI_{ji}, m_{akji} and m_{bkji} in (4.21) and (4.22) are computed numerically in simulation validation. The algorithm for computing the gradient vector of a function numerically at a point is shown in Appendix A.

4.4.2 Approximation of $J_p(z_{(N)})$

The first part of the objective function in (4.11), i.e., $J_p(z_{(N)})$, contains three terms in quadratic form of task space positions. All of these three terms can be approximated by a quadratic form of joint space positions. In the first term of $J_p(z_{(N)})$, $x_e(z_{(N)})$ can be rewritten by using the velocity of the end-effector in task space and the differential kinematics equation, $\dot{x}_e(z_{(k)}) = J_e(q_{(k)})\dot{q}_{(k)}$, as

$$\begin{aligned} x_e(z_{(N)}) &= x_e(z_{(0)}) + \sum_{k=0}^{N-1} \dot{x}_e(z_{(k)})\Delta t \\ &= x_e(z_{(0)}) + \sum_{k=0}^{N-1} J_e(q_{(k)})\dot{q}_{(k)}\Delta t, \end{aligned} \tag{4.23}$$

where $J_e(q_{(k)}) \in \mathbb{R}^{N_x \times N_j}$ is the Jacobian matrix for the end-effector of the robot manipulator evaluated at the configuration of $q_{(k)}$, and N_x is the dimension of task space. The evaluation of the Jacobian matrix, $J_e(q_{(k)})$, is nonlinear and non-convex with respect to the joint position. The Jacobian matrix, $J_e(q_{(k)})$, is approximated by $J_e(q_k^0)$. Then $(x_e(z_{(N)}) - x_g)$ in the first term of $J_p(z_{(N)})$ becomes,

$$\begin{aligned} x_e(z_{(N)}) - x_g &\approx x_e(z_{(0)}) + \sum_{k=0}^{N-1} J_e(q_k^0)\dot{q}_{(k)}\Delta t - x_g \\ &= x_e(z_{(0)}) + \Delta t\, J_e(q_{(0)})\, \dot{q}_{(0)} + \\ &\qquad \Delta t\, J_e(q_{(1)}^0)\, \dot{q}_{(1)} + \cdots + \Delta t\, J_e(q_{N-1}^0)\, \dot{q}_{(N-1)} - x_g \\ &= \Delta t B_{pe} s_z + a_{pe}, \end{aligned} \tag{4.24}$$

where

$$a_{pe} = \big(x_e(z_{(0)}) + \Delta t\, J_e(q_{(0)})\dot{q}_{(0)} - x_g\big) \in \mathbb{R}^{N_x}, \tag{4.25}$$

$$B_{pe} = \Big[\mathbf{0}, J_e(q_1^0), \ldots, \mathbf{0}, J_e(q_{N-1}^0), \mathbf{0}, \mathbf{0}\Big] \in \mathbb{R}^{N_x \times 2NN_j}, \tag{4.26}$$

$$s_z = \Big[z_{(1)}^T, z_{(2)}^T, \ldots, z_{(N)}^T\Big]^T \in \mathbb{R}^{2NN_j}. \tag{4.27}$$

$q_{(0)}$ and $\dot{q}_{(0)}$ are the joint position and velocity of the current time step, which are known. a_{pe} is a vector of length N_x. Every $\mathbf{0}$ in B_{pe} represents a zero matrix of the same size as the Jacobian matrix, $J_e(\bullet)$. Thus the size of B_{pe} is N_x by $2N_jN$. Note that both a_{pe} and B_{pe} are constant and can be computed before solving the optimization problem in each iteration. s_z is a vector stacked by all $z_{(k)}$'s in Z_k, and has the length of $2N_jN$. With this approximation, the first term in $J_p(z_{(N)})$ is transformed into a quadratic form of all $z_{(k)}$'s as,

$$\big(x_e(z_{(N)}) - x_g\big)^T \big(x_e(z_{(N)}) - x_g\big) \approx \big(\Delta t B_{pe} s_z + a_{pe}\big)^T \big(\Delta t B_{pe} s_z + a_{pe}\big). \tag{4.28}$$

In the last two terms of $J_p(z_{(N)})$, $x_{ne}(z_{(N)})$ and $x_{se}(z_{(N)})$ are unit vectors starting from $x_e(z_{(N)})$ and can be obtained by

$$x_{ne}(z_{(N)}) = x_n(z_{(N)}) - x_e(z_{(N)}), \tag{4.29}$$

$$x_{se}(z_{(N)}) = x_s(z_{(N)}) - x_e(z_{(N)}), \tag{4.30}$$

where $x_n(\bullet)$ and $x_s(\bullet)$ are the points one unit away from $x_e(\bullet)$ along the normal and the sliding directions, respectively. Then the last two terms of $J_p(z_N)$ can be approximated by quadratic form of all $z_{(k)}$'s using the same procedure as

$$\begin{aligned} &\big(x_{ne}(z_{(N)}) - x_{gne}\big)^T \big(x_{ne}(z_{(N)}) - x_{gne}\big) \\ &\qquad \approx \Big(a_{pn} - a_{pe} + \Delta t(B_{pn} - B_{pe}) s_z\Big)^T \Big(a_{pn} - a_{pe} + \Delta t(B_{pn} - B_{pe}) s_z\Big), \end{aligned} \tag{4.31}$$

$$\begin{aligned} &\big(x_{se}(z_{(N)}) - x_{gse}\big)^T \big(x_{se}(z_{(N)}) - x_{gse}\big) \\ &\qquad \approx \Big(a_{ps} - a_{pe} + \Delta t(B_{ps} - B_{pe}) s_z\Big)^T \Big(a_{ps} - a_{pe} + \Delta t(B_{ps} - B_{pe}) s_z\Big), \end{aligned} \tag{4.32}$$

where

$$a_{pn} = x_n(z_{(0)}) + \Delta t\, J_n(q_{(0)})\, \dot{q}_{(0)} - x_{gn}, \tag{4.33}$$

$$a_{ps} = x_s(z_{(0)}) + \Delta t\, J_s(q_{(0)})\, \dot{q}_{(0)} - x_{gs}, \tag{4.34}$$

$$B_{pn} = \big[\mathbf{0},\ J_n(q_1^0),\ \mathbf{0},\ J_n(q_2^0),\ \ldots,\ \mathbf{0},\ J_n(q_{N-1}^0),\ \mathbf{0},\ \mathbf{0}\big], \tag{4.35}$$

$$B_{ps} = \big[\mathbf{0},\ J_s(q_1^0),\ \mathbf{0},\ J_s(q_2^0),\ \ldots,\ \mathbf{0},\ J_s(q_{N-1}^0),\ \mathbf{0},\ \mathbf{0}\big]. \tag{4.36}$$

Matrices $J_n(\bullet)$ and $J_s(\bullet)$ are the Jacobian matrices of $x_n(\bullet)$ and $x_s(\bullet)$ evaluated at $\bullet$. Similarly, all $a_\bullet$'s and $B_\bullet$'s are constant vectors and constant matrices and can be computed before solving the optimization problem in each iteration. Hence the last two terms are in the quadratic form of joint space variables.

4.4.3 The Resulting Quadratic Formulation

Because of the approximation of $J_p(z_{(N)})$ and $SI(z_{(k)})$, the objective function in (4.10) becomes in quadratic form, and all inequality constraints become linear inequalities. With some algebraic manipulation, the optimization problem in (4.10) then can be transformed into a standard quadratic programming problem with linear constraints as

$$\min_{s} \quad s^T H s + f^T s \tag{4.37a}$$

$$s.t. \quad A_{eq} s = b_{eq} \tag{4.37b}$$

$$A_{ineq} s \leq b_{ineq} \tag{4.37c}$$

$$\underline{s} \leq s \leq \overline{s}. \tag{4.37d}$$

where

$$s = [s_z^T, u_{(0)}^T, \cdots, u_{(N-1)}^T]^T \tag{4.38}$$

$$\underline{s} = [\underline{z}_{(1)}^{qp}, \cdots, \underline{z}_{(N)}^{qp}, \underline{u}, \cdots, \underline{u}] \tag{4.39}$$

$$\overline{s} = [\overline{z}_{(1)}^{qp}, \cdots, \overline{z}_{(N)}^{qp}, \overline{u}, \cdots, \overline{u}] \tag{4.40}$$

and

$$\underline{z}_{(k)}^{qp} = \max\{\underline{z}, z_k^0 - \epsilon\}, \quad \forall k \in \{1, \cdots, N\}$$
$$\overline{z}_{(k)}^{qp} = \min\{\overline{z}, z_k^0 + \epsilon\}, \quad \forall k \in \{1, \cdots, N\}.$$

Since the quadratic formulation is derived by using z_k^0 to approximate $z_{(k)}$, a trust bound, ϵ, for the approximation is necessary.

The resulting quadratic formulation has $3NN_j$ variables, $2NN_j$ equality constraints, $2N_eN_oN$ inequality constraints for the safety of the agent ellipsoids, and upper and lower bounds for all variables. A comparison of the size of the formulated optimization problem between (4.10) and (4.37) is summarized in Table 4.1.

4.5 Receding Horizon Strategy

The solution of the optimization problem in (4.10) is a trajectory with the length of N time steps. The decision making block then outputs the generated trajectory to the controller block and the control command is generated for the robot manipulator to track this trajectory. The resulting control is then applied in a fashion of receding horizon strategy. Since the sensor measurement about the agent is available in every T_c robot controller time step, it is reasonable to apply the receding horizon strategy when a new measurement is available. In other words, a collision-free trajectory is generated iteratively in every T_c time step, and only the first T_c control command is applied to the robot manipulator. Figure 4.2 shows the schematic of the receding horizon strategy, where k_c is the current time step. In the figure, the upper box represent the trajectory generated at the current time step, and the lower box is the previously generated trajectory.

Table 4.1: Summary of the optimization problems in the proposed trajectory generation algorithm.

	(4.10)	(4.37)
Formulation of optimization problem	The nonlinear and non-convex optimization problem	The approximated quadratic optimization problem
# of variables	$3NN_j$	
# of equality constraints	$2NN_j$	
# of inequality constraints	N	$2NN_eN_o$

N: The planning time horizon.
N_j: The number of the robot joints.
N_e: The number of the ellipsoid coordinates in the robot.
N_o: The number of the ellipsoids used to represent the agent.

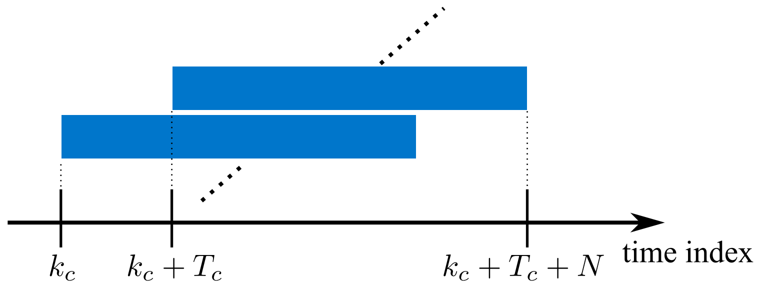


Figure 4.2: The schematic of the receding horizon strategy, where k_c indicates the current time step, and N is the predefined planning time horizon. Sensor measurements are available in every T_c time step.

4.6 Inequality Constraints Selection Strategy

The approximation derived in Section 4.4 shows the transformation of the original nonlinear and non-convex formulation into a standard quadratic formulation with linear constraints. In the formulation of the quadratic problem, some vectors and matrices that are treated as constant parameters, nevertheless, vary from iteration to iteration. For instance, a_{pe} and B_{pe} in $J_p(z_{(N)})$, or c_{akji} and m_{akji} in $SI(z_{(k)})$. In implementation, these parameters need to be computed before solving the quadratic problem in each iteration. Particular attention should be paid to the parameters in the linearized safety inequality constraints, i.e., $c_{\bullet kji}$, and $m_{\bullet kji}$ in (4.20). The number of these inequality constraints depends on three numbers: the number of the ellipsoid coordinates on the robot, N_e, the number of agent ellipsoids, N_o,

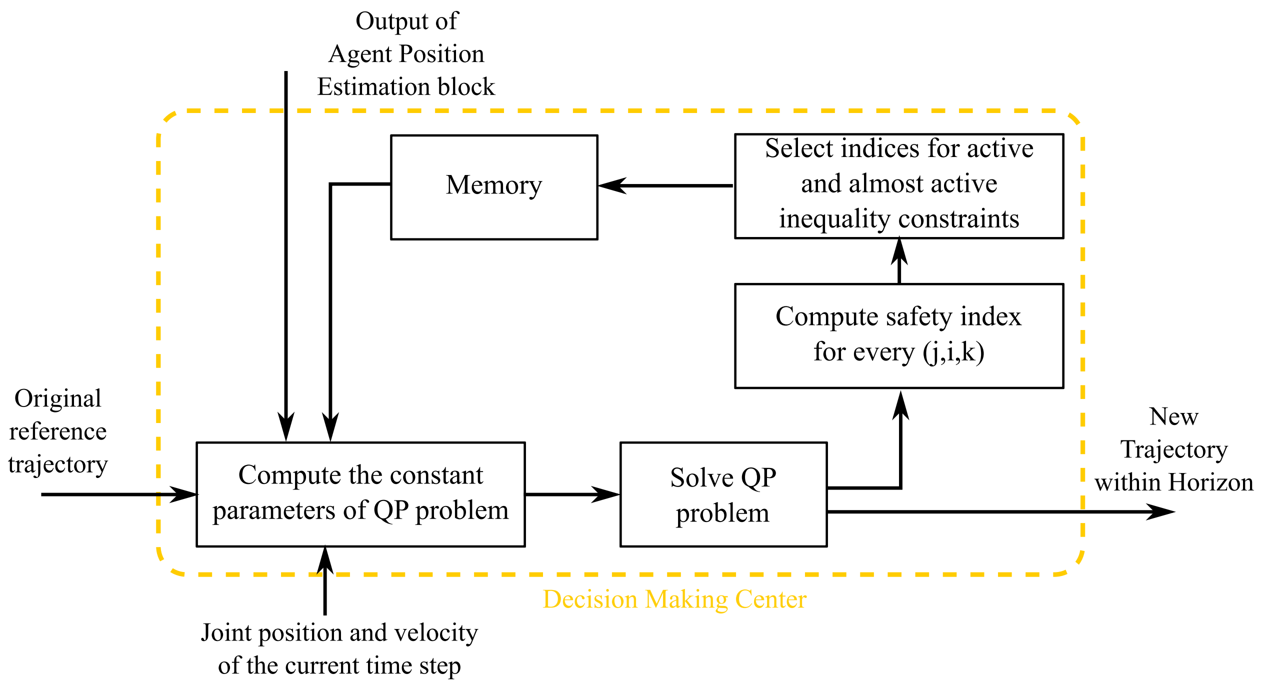


Figure 4.3: The flow chart of the selection method of inequality constraints.

and the length of the planning time horizon, N. Furthermore, there are two safety inequality constraints for every pair of (j, i) in every time step.

From the perspective of the physical meaning of the safety index, it is unlikely that a solution of (4.37) activates all of the inequality constraints of the same time step. Thus, it would further reduce the computational load if only those inequality constraints which are more likely to become active are selected and used in the quadratic problem. The computational load can be reduced in two aspects. First, the quadratic problem could be solved faster because it has fewer inequality constraints. Second, we do not need to compute all of the gradient vectors in the Taylor expansion, i.e., m_{akji} and m_{bkji}. This may greatly save the computational load since m_{akji} and m_{bkji} are computed numerically in simulation validation.

A heuristic, yet intuitive, method to select the inequality constraints is proposed. The idea is to keep track of those indices of (j, i, k) for the inequality constraints which are active or almost active in the generated trajectory of the previous iteration. Then only the inequality constraints corresponding to these indices are selected and used in the optimization problem of the current iteration. An active or almost active inequality constraint implies that the slack variable of this inequality constraint is zero or very close to zero. For example, the equality equation transformed from the h-th inequality constraints of (4.37c) by introducing the slack variable can be written as

$$a_{h,ineq}\, s + y_h = b_{h,ineq}, \tag{4.41}$$

where $a_{h,ineq}$ is the h-th row of A_{ineq} in (4.37c), $b_{h,ineq}$ is the h-th element of b_{ineq}, and y_h is the slack variable. The slack variable is always non-negative. When y_h is zero or very small, the h-th inequality constraints, $a_{h,ineq}\, s \leq b_{h,ineq}$ is active or almost active.

This method is reasonable since the situation of the current iteration should not change much from the situation of the previous iteration when T_c is not very large. Figure 4.3 shows the flow chart of the inequality constraints selection strategy. After solving the quadratic problem (4.37), the safety index for every pair of (j, i) in every time step within the planning horizon is computed. Then the indices of (j, i, k) for the inequality constraints which are active or almost active are selected for the next iteration. In simulation validation, when the slack variable of an inequality constraint is less than or equal to 0.1, it is counted as an almost active inequality constraint.

4.7 Summary

In this chapter, the proposed online trajectory generation algorithm is stated in detail. This algorithm is developed under the optimization-based path planning framework with the use of receding horizon strategy, discretized kinematics model, and the safety index designed in Section 3. By solving the optimization problem, the solution is a trajectory within a short time horizon which achieves the functions of leading the robot manipulator toward the goal position and avoiding any collisions with agents simultaneously.

The formulated optimization problem, however, is highly nonlinear and non-convex because of the transformation of variables between joint space and task space. In order to achieve online trajectory generation, the formulated optimization problem is approximated by a quadratic formulation. Furthermore, a heuristic method is proposed for the selection of the active or almost active inequality constraints to further reduce the computational load.

Chapter 5

Simulation Studies

5.1 Introduction

In this chapter, the proposed online trajectory generation algorithm along with the robot safety system proposed in Section 2.2 are implemented and simulated. Both the nonlinear and non-convex optimization problem in (4.10) and the quadratic problem in (4.37) with/without the selection strategy are simulated for comparison. The simulation environment and the motion of agents were introduced in Chapter 2. The nonlinear and non-convex optimization problem in (4.10) is solved by the `fmincon` solver with the `SQP` algorithm of the MATLAB optimization toolbox, and the quadratic problem is solved by the `Gurobi` solver from Gurobi Optimization Inc.[20]. The first candidate of the overall safety index described in (3.14) is used in simulations in this chapter. For simplicity, NLP and QP are used to stand for the nonlinear and non-convex optimization problem in (4.10) and the quadratic problem in (4.37) in the following simulation results. The quadratic problem with the use of the selection strategy of inequality constraints is denoted by QPS.

5.2 Two-Link Planar Robot Manipulator

The parameters used in the optimization problem in the case of the two-link planar robot are listed in Table 5.1. The weighting factor k_{dsi} is chosen by setting the maximum acceptable DSI, i.e., S_{ad} in (3.15), as 1, and k_{msi} is chosen by allowing the maximum linear momentum toward the agents, S_{am}, to be 0.2 m/s when DSI is 1. The factor k_{po} in k_p is the squared distance from S to E, which is 0.36 m^2 in the case of the two-link planar robot. The factor k_{po} and the length of the planning horizon N act as a normalization factor for k_p. Because of the simple mechanical structure in the case of the two-link planar robot, only the task space squared distance from the end-effector at the last step of the planning horizon to the final goal position is penalized in simulations. In other words, only the first term in $J_p(z_{(N)})$ is used, and hence there is no k_{pns} in the table.

Table 5.1: Parameters of optimization problem in simulations of the two-link planar robot manipulator.

k_{dsi}	k_{msi}	k_p	k_{po}	k_u	ϵ	S_a	Δt
1	25	$10N/k_{po}$	0.36 (m^2)	0.0025	0.5	1	$0.01(sec)$

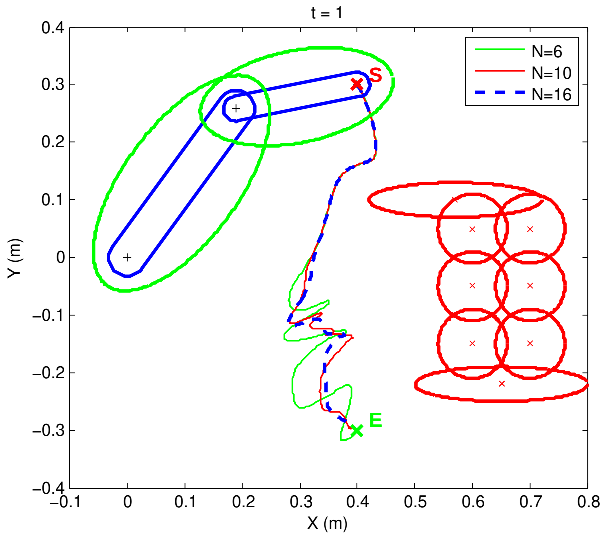
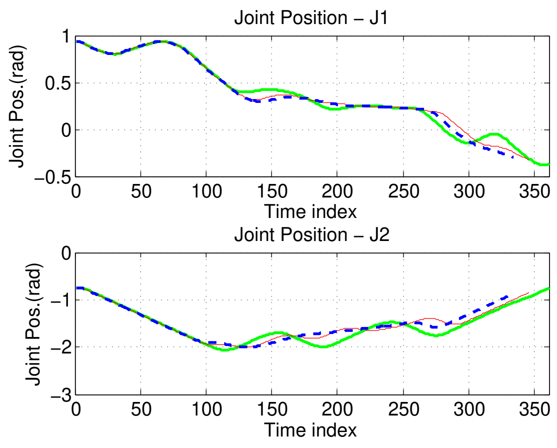


Figure 5.1: An example of different planning horizons in simulations of human–robot collaboration for the case of the two-link planar robot

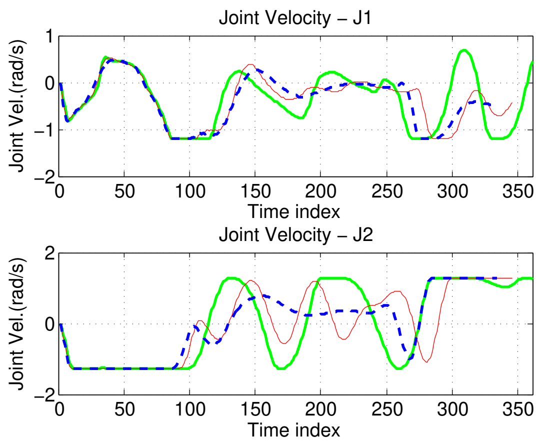
In simulations, the robot controller sampling time is set as 0.01 sec. From the assumption made in Section 2.3, the information of the human worker is set to be available for every three robot controller time steps, and the information of the master robot manipulator is available for every time step. In other words, T_c is equal to three in human–robot collaboration, and is equal to one in decentralized robot–robot collaboration. The planning horizon N is set as 16 time steps in both scenarios of robot collaborations.

5.2.1 The Selection of the Planning Horizon N

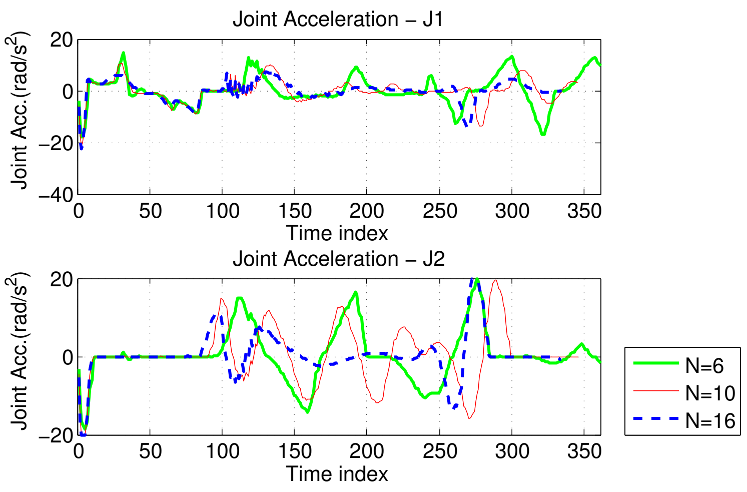
The length of the planning horizon, N, plays an important role in the optimization-based framework since it directly affects the size of the optimization problem. For the proposed trajectory generation algorithm, the numbers of variables and constraints in the optimization problem, as shown in Table 4.1, change with the planning horizon. A long planning horizon provides the advantage of the early detection of a potential collision, but results in a large size



(a)



(b)



(c)

Figure 5.2: The resulting trajectories of simulations with the planning horizon of 6, 10, and 16 in human–robot collaboration for the case of the two-link planar robot using NLP. (a) Joint position, (b) joint velocity, and (c) joint acceleration.

optimization problem and hence being incapable of real-time trajectory generation. Another issue, particularly in the proposed algorithm, is the motion estimation of the agents. The estimated position of an agent ellipsoid in the far future maybe greatly deviate from its real position. On the other hand, an optimization problem with a short planning horizon can be solved more quickly, but the resulting trajectory may contain more oscillations. The reason is that when an agent ellipsoid appears and is detected by the robot system, the robot needs to react and avoid the agent in a short time. This generally results in a rapid and huge change in the joint velocity and joint acceleration. Thus, the selection of the length of the

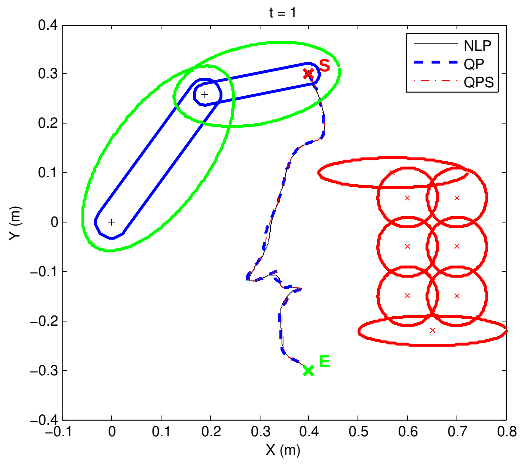


Figure 5.3: The resulting paths in simulations of human–robot collaboration for the case of the two-link planar robot using formulation (4.10), (4.37), and (4.37) with selection strategy for inequality constraints. They are denoted as NLP, QP, and QPS.

planning horizon needs to be considered carefully.

The scenario of human–robot collaboration for the case of the two-link planar robot is used as an example to demonstrate the trajectories generated by the same formulation but with different planning horizons. Three trajectories are generated by NLP with the planning horizon of 6, 10, and 16 time steps, and the corresponding paths are shown in Figure 5.1. In the figure, the agent ellipsoids representing the human worker are in their initial positions. As seen in the figure, when the robot manipulator is close to the left hand of the human worker, its motion in the simulation of the planning horizon of 6 time steps contains more oscillations than the motions in simulations of $N = 10$ and $N = 16$. The oscillations in the motion can be seen as well in the trajectories shown in Figure 5.2. The robot manipulator in three simulations reaches the goal position. However, it takes 361 time steps in the simulation with the planning horizon of $N = 6$ time steps, which is longer than the time in other simulations. It takes 346 and 334 time steps in the simulations with the planning horizon of N =10 and 16, respectively.

The selection of the planning horizon for the case of the two-link planar robot is chosen as 16 time steps, and for the case of the ITRI seven-DOF robot manipulator, it is selected as 12 time steps.

5.2.2 Human–Robot Collaboration

Figure 5.3 shows the resulting paths of the robot end-effector generated by NLP, QP, and QPS in simulations. In the figure, the three paths in the black solid line, the blue dashed line, and the red dash-dot line represent the resulting paths of NLP, QP, and QPS, respectively.

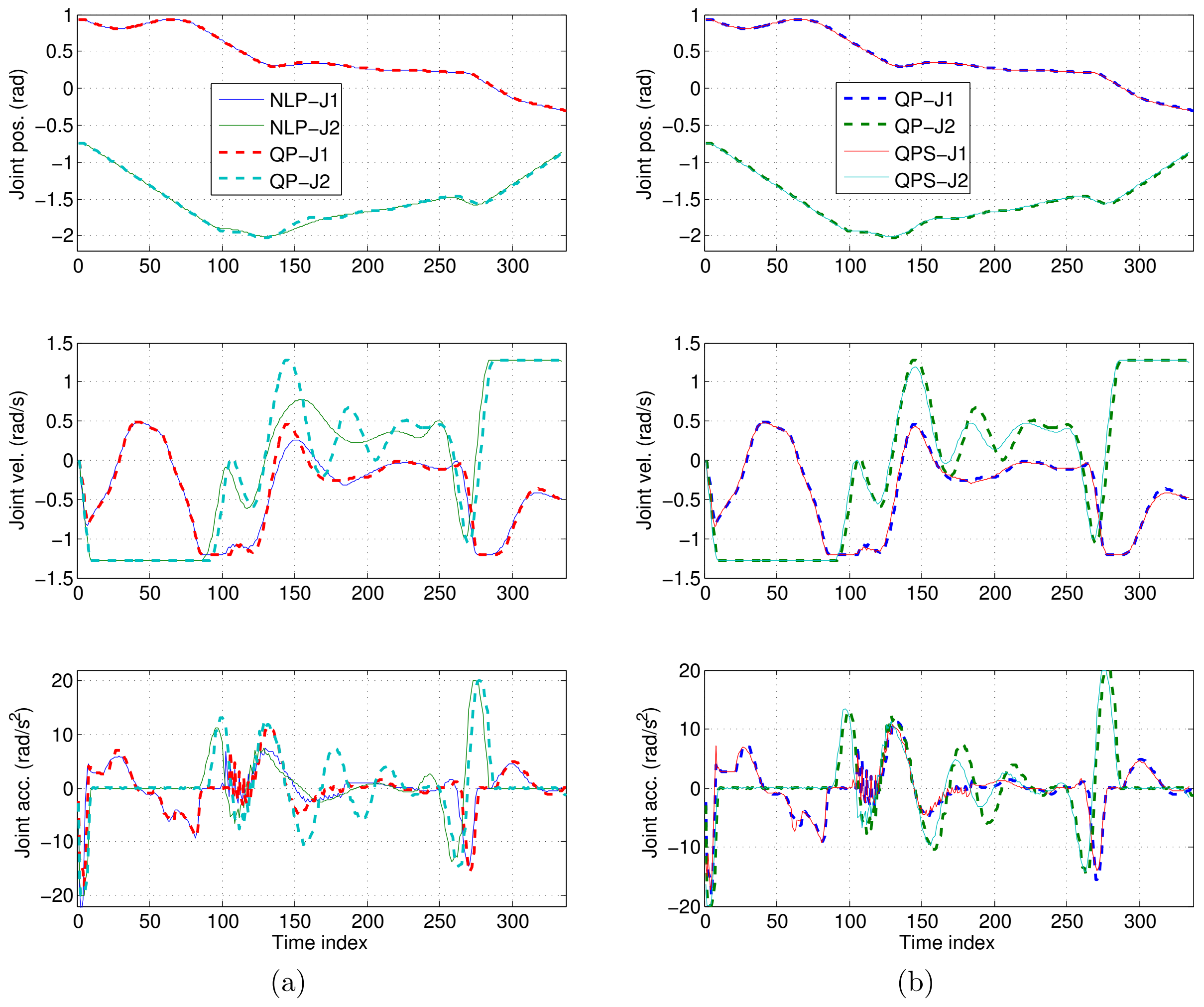


Figure 5.4: The resulting trajectories in simulations of human–robot collaboration for the case of the two-link planar robot using NLP, QP, and QPS. (a) The comparison of trajectories between NLP and QP, and (b) the comparison of trajectories between QP and QPS.

The two green ellipses enveloping the two links of the robot are the unit ellipses of the ellipsoid coordinates attached to the robot links. The agent ellipsoids representing the human worker are in their initial positions. It is seen that the robot manipulator successfully reaches the goal configuration in all simulations.

The end-effector paths generated by the three formulations show a high similarity. The resulting trajectories are shown in Figure 5.4, where Figure 5.4a is the comparison between the trajectories of using NLP and QP, and Figure 5.4b is the comparison between QP and QPS. It can be seen in Figure 5.4a that the trajectory of using NLP is slightly different from the trajectory generated by QP. The difference only shows up in the joint velocity and joint acceleration at the time step of around 150 to 250. This shows that the quadratic

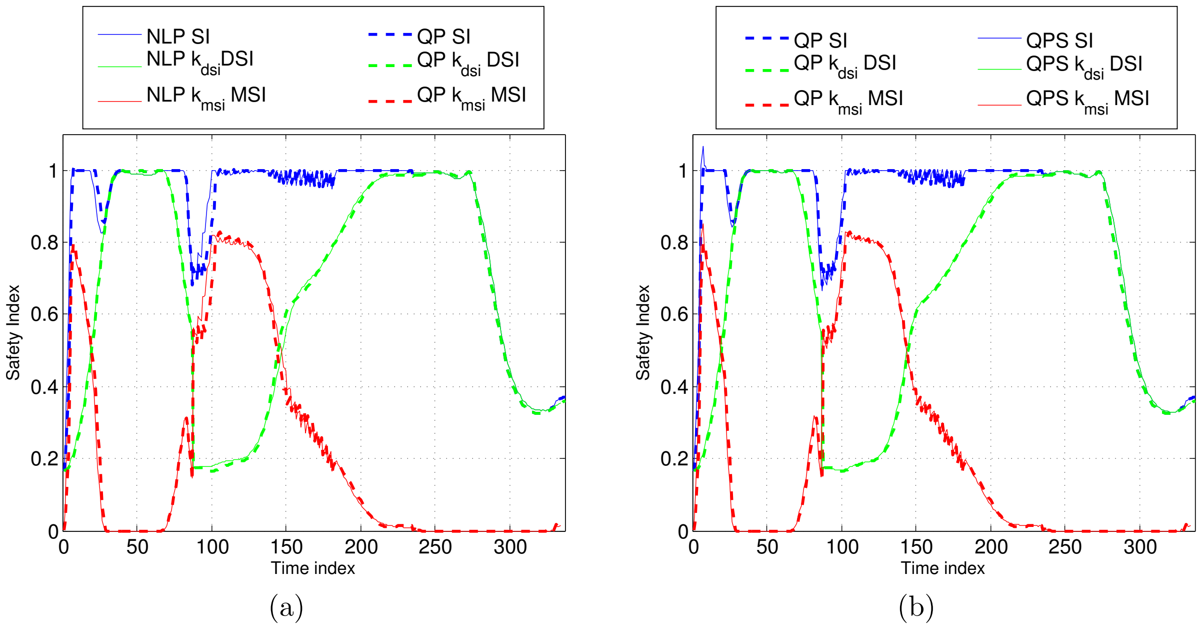


Figure 5.5: The resulting safety indices in simulations of human–robot collaboration for the case of the two-link planar robot using NLP, QP, and QPS. All the blue, green, and red lines represent the overall safety index, the DSI, and the MSI. (a) The comparison between NLP and QP; (b) The comparison between QP and QPS.

formulation can effectively approximate the nonlinear and non-convex formulation in the case of the two-link planar robot. In Figure 5.4b, the trajectories generated by QP and QPS are almost identical to each other, which implies that the selection strategy of the inequality constraints does select the proper inequality constraints.

The safety indices of simulations of NLP, QP, and QPS are shown in Figure 5.5. In the figure, the overall safety index, denoted by SI, is represented by blue lines, either solid or dashed. The pair of $k_{dsi}DSI_{ji}$ and $k_{msi}MSI_{ji}$ which imposes the maximum SI are shown in the figure in green and red lines. A similar profile of the safety index in the three simulations can be seen in the figure. This result is expected because of the similarity in their trajectories. The SI in the three simulations are always less than or equal to the predefined threshold, S_a, which indicates that the human worker is safe and has no collision with the robot arm during his motion.

Figure 5.6 shows the snapshots of the simulation of using NLP. Figure 5.6a and Figure 5.6b show that the robot manipulator starts its motion, makes a turn, and then avoids the right hand of the human worker. The turn is caused by the safety index reaching the threshold, as seen in Figure 5.5a. In the period that the safety index is reaching the threshold, the unit ellipse of the ellipsoid coordinate on the second robot link contacts the red ellipse representing the right hand of the human worker, as shown in Figure 5.6b. From Figure 5.6b to Figure 5.6c, the motion of the robot arm tends to move toward the goal position and

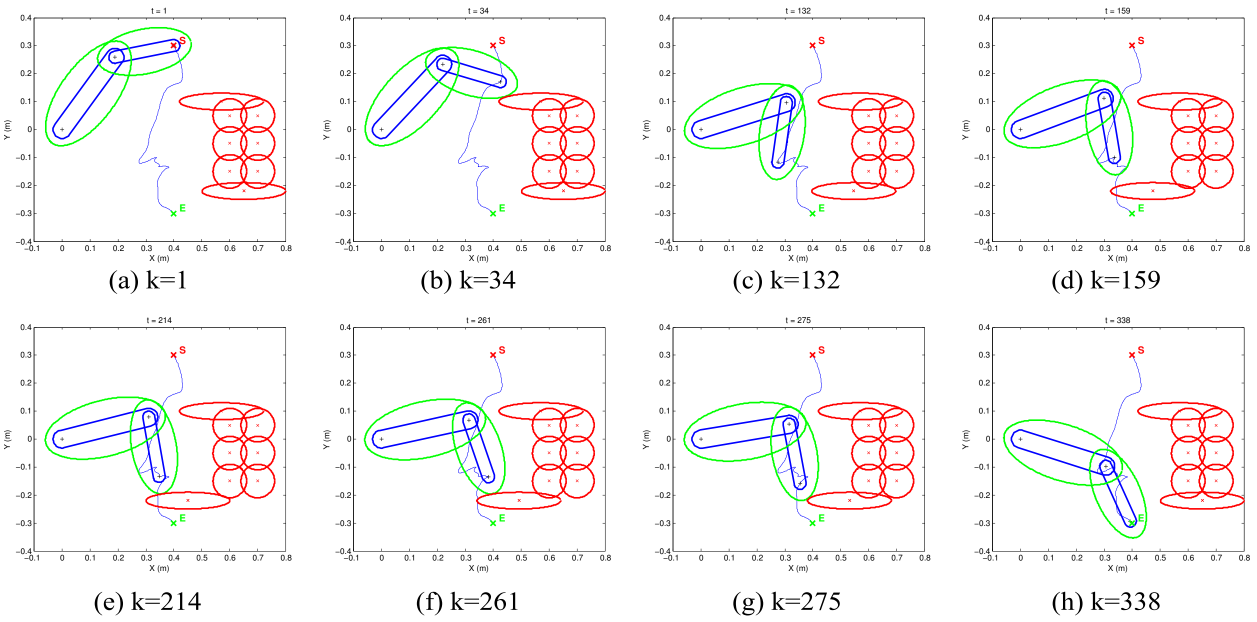


Figure 5.6: Snapshots of the simulation of human–robot collaboration for the case of two-link robot manipulator using NLP.

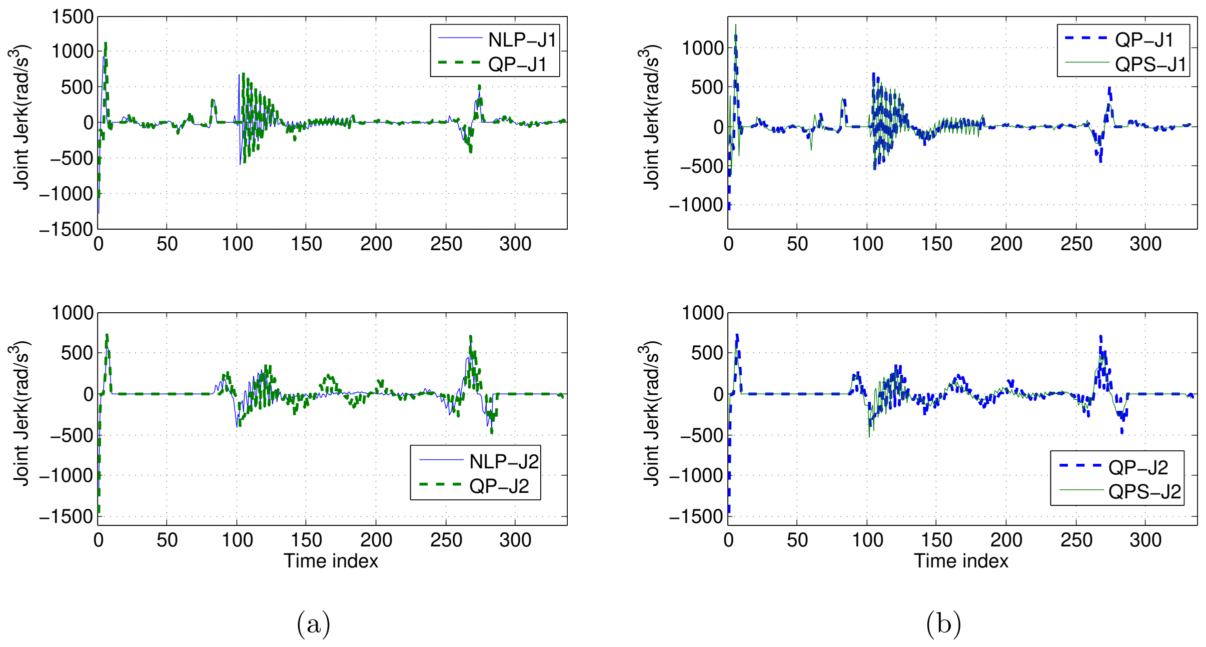


Figure 5.7: The joint jerks (derived from joint accelerations numerically) in simulations of human–robot collaboration for the case of the two-link planar robot. (a) The comparison between NLP and QP, and (b) the comparison between QP and QPS.

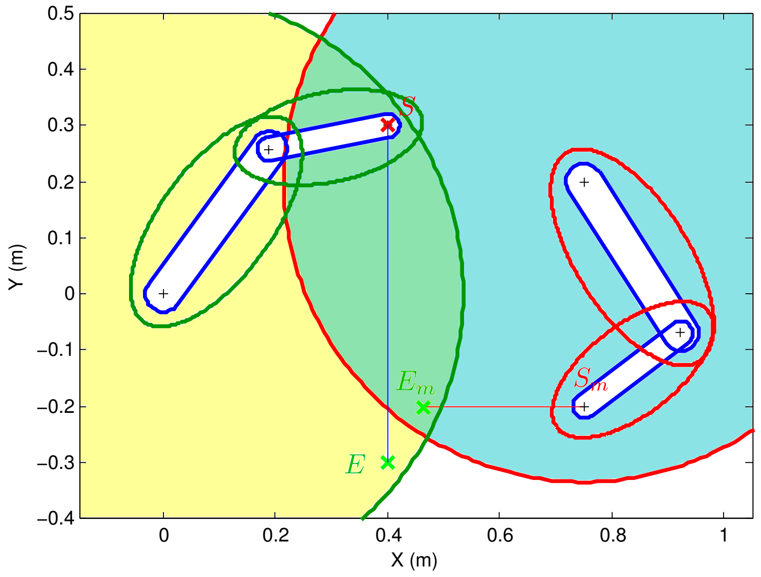


Figure 5.8: The workspace of two robot manipulators in simulations of decentralized robot–robot collaboration.

meanwhile avoid the human worker. In about the same period of time, the human worker stretches out his/her left hand. The motion of the left hand ellipsoid causes the increase of the momentum safety index, and as a result, the robot manipulator moves roughly parallel to the X-axis but in the opposite direction in order to reduce the momentum safety index. This can be shown in Figure 5.6c to Figure 5.6e. After a short period of pause, the human worker moves back his/her hands, and the robot manipulator also detours and then reaches its goal position, as shown from Figure 5.6f to Figure 5.6h.

The joint jerks of the resulting trajectories are derived numerically from the joint accelerations and shown in Figure 5.7. As seen in the figure the joint jerks in the three simulations are all within the range from -1500 rad/s^3 to 1500 rad/s^3, and in most of the time the joint jerks even stay within 1000 rad/s^3 to 1000 rad/s^3. This indicates that three trajectories generated by the three formulations have a bounded jerk and are smooth. However, there exists small chattering in the joint jerk and the joint acceleration.

5.2.3 Decentralized Robot–Robot Collaboration

The simulation results of using NLP, QP, and QPS in decentralized robot–robot collaboration are shown in Figure 5.9 through Figure 5.12. The resulting end-effector paths of the slave robot manipulator are illustrated in Figure 5.9. The two red ellipses are the agent ellipsoids used to represent the master robot manipulator, and the red horizontal line is the path of its end-effector in simulations. The resulting paths of all simulations are very close to each other, and it can also be observed from the trajectories generated by these three formulations

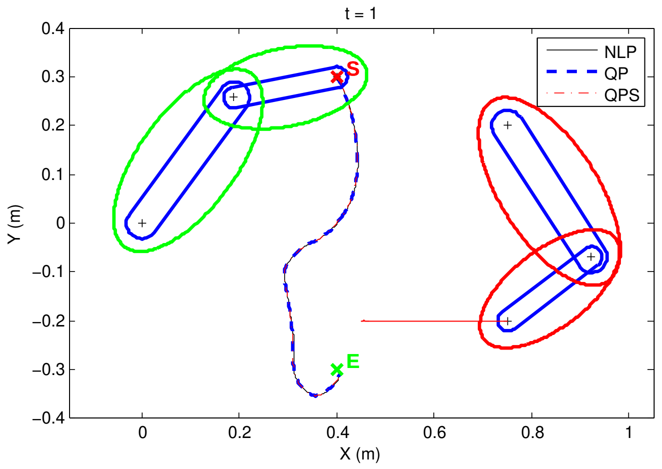


Figure 5.9: The resulting paths in simulations of decentralized robot–robot collaboration for the case of the two-link planar robot using NLP, QP, and QPS.

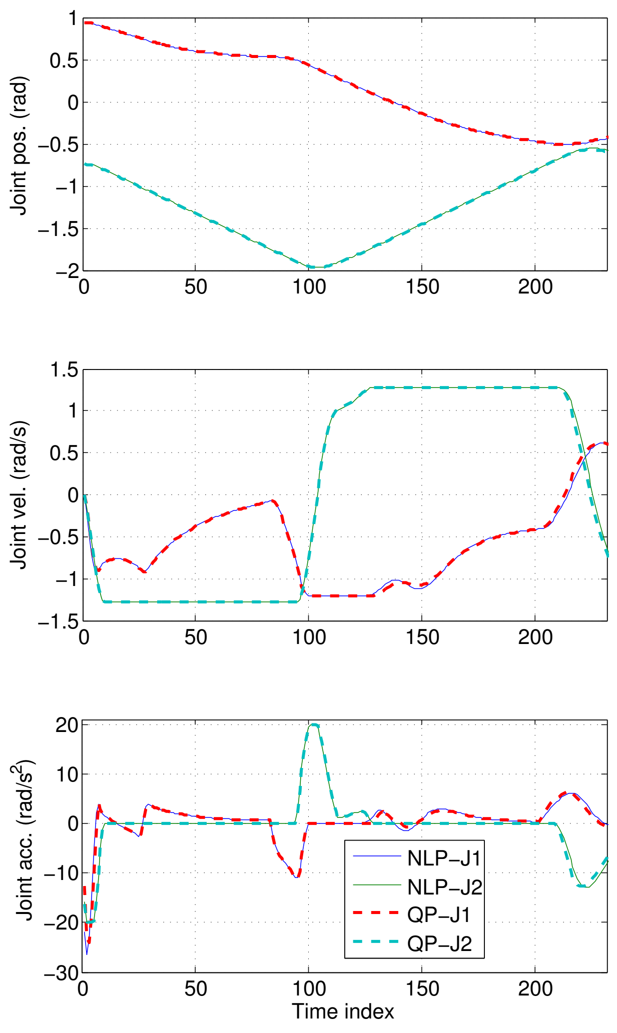
in Figure 5.10. This indicates that QP can effectively approximate NLP in this case, and moreover, the selection strategy of inequality constraints works successfully.

The resulting safety indices are shown in Figure 5.11. There is no distinguishable difference between the safety index of the three simulations, and it is always less than or equal to S_a in simulations. It shows that no collision happens in three simulations. The snapshots of the simulation using NLP are shown in Figure 5.12.

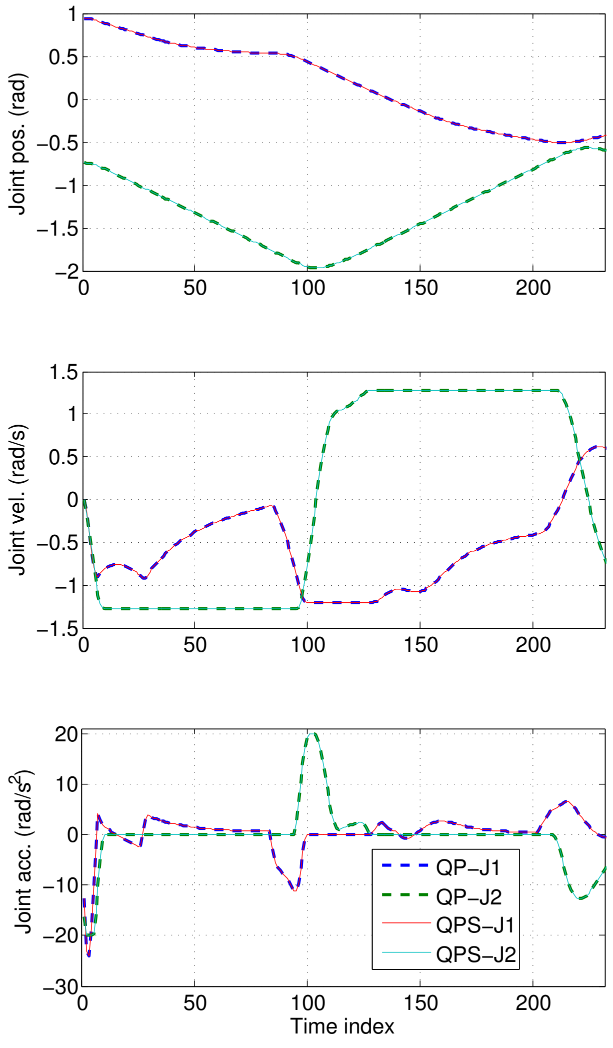
The joint jerks of the resulting trajectories are derived numerically from the joint accelerations and are shown in Figure 5.13. It is seen that the jerks in all three simulations are bounded by -1000 rad/s^3 and 1000 rad/s^3. Furthermore, there is no chattering in the joint jerk. This indicates that the three trajectories generated by the three formulations are smooth.

5.2.4 Computation Time

The sizes of the optimization problems in the three formulations and the average computation time for each iteration in the simulations are summarized in Table 5.2. The first three rows show the size of the optimization problem in the three formulations. Note that because of the use of the selection strategy for the inequality constraints, the number of inequality constraints in the formulation of QPS varies from iteration to iteration. Thus the number of inequality constraints in the column of QPS is not a fixed value. The number of iterations in each simulation indicates how long it takes for the robot manipulator to reach the goal position. It can be seen that in the simulations of the two-link planar robot, the average



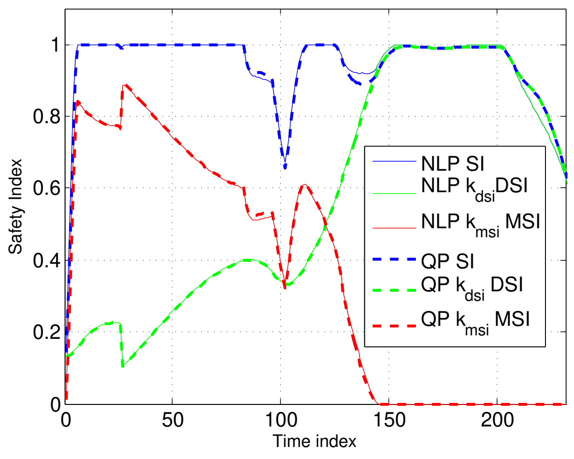
(a)



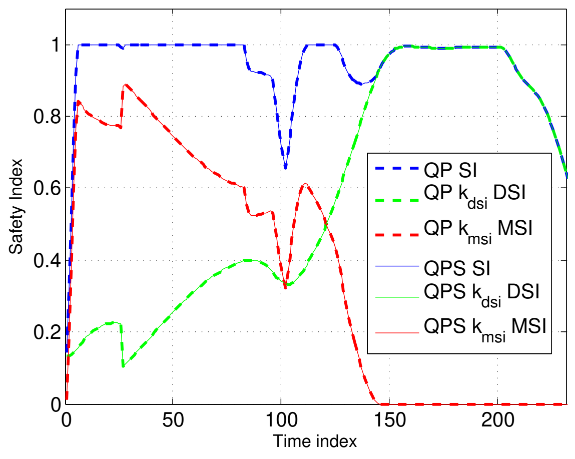
(b)

Figure 5.10: The resulting trajectories in simulations of decentralized robot–robot collaboration for the case of the two-link planar robot using NLP, QP, and QPS. (a) The comparison between NLP and QP, and (b) the comparison between QP and QPS.

computation time of using QP is 0.0096 sec for human–robot collaboration and 0.0082 sec for decentralized robot–robot collaboration. It is about 1/600 of the average time of using NLP, which is 6.2853 sec for human–robot collaboration and 5.2604 sec for decentralized robot–robot collaboration. With the use of selection strategy for the inequality constraints, the average computation time is further reduced to 0.0069 sec in human–robot collaboration, which is a 28% decrease. In the last row of Table 5.2, it lists the time spent on the Taylor series expansion, which is used in approximation of the safety constraints as stated in (4.20) and (4.21). It can be seen that the reduction in the average computation time is mainly

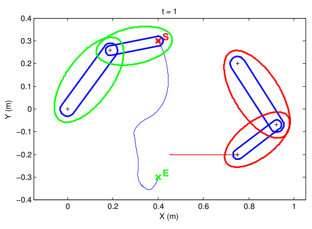


(a)

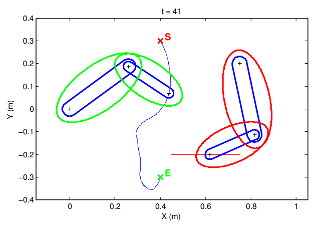


(b)

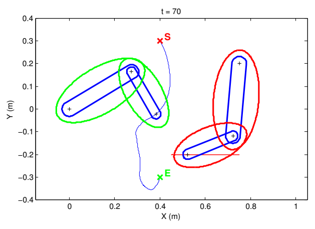
Figure 5.11: The resulting safety indices in simulations of decentralized robot–robot collaboration for the case of the two-link planar robot using NLP, QP, and QPS. The blue, green, and red lines represent the overall safety index, SI, the $k_{dsi}DSI$, and the $k_{msi}MSI$, respectively. (a) The comparison between NLP and QP; (b) The comparison between QP and QPS.



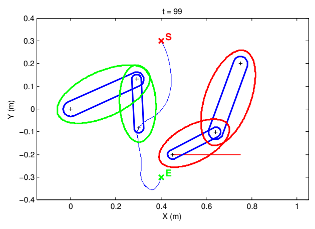
(a) k=1



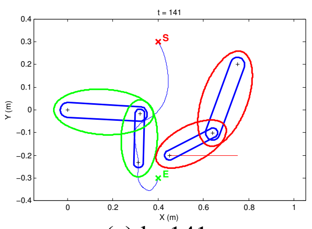
(b) k=41



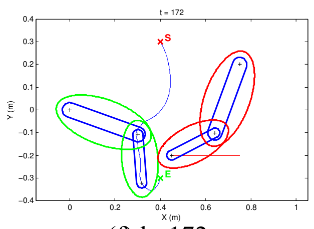
(c) k=70



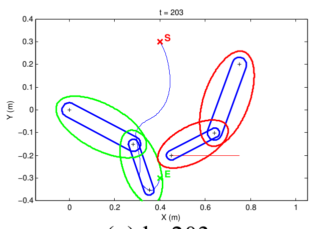
(d) k=99



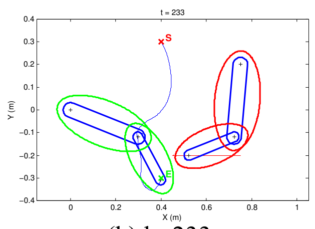
(e) k=141



(f) k=172



(g) k=203



(h) k=233

Figure 5.12: Snapshots of the simulation using NLP in decentralized robot–robot collaboration for the case of the two-link planar robot manipulator.

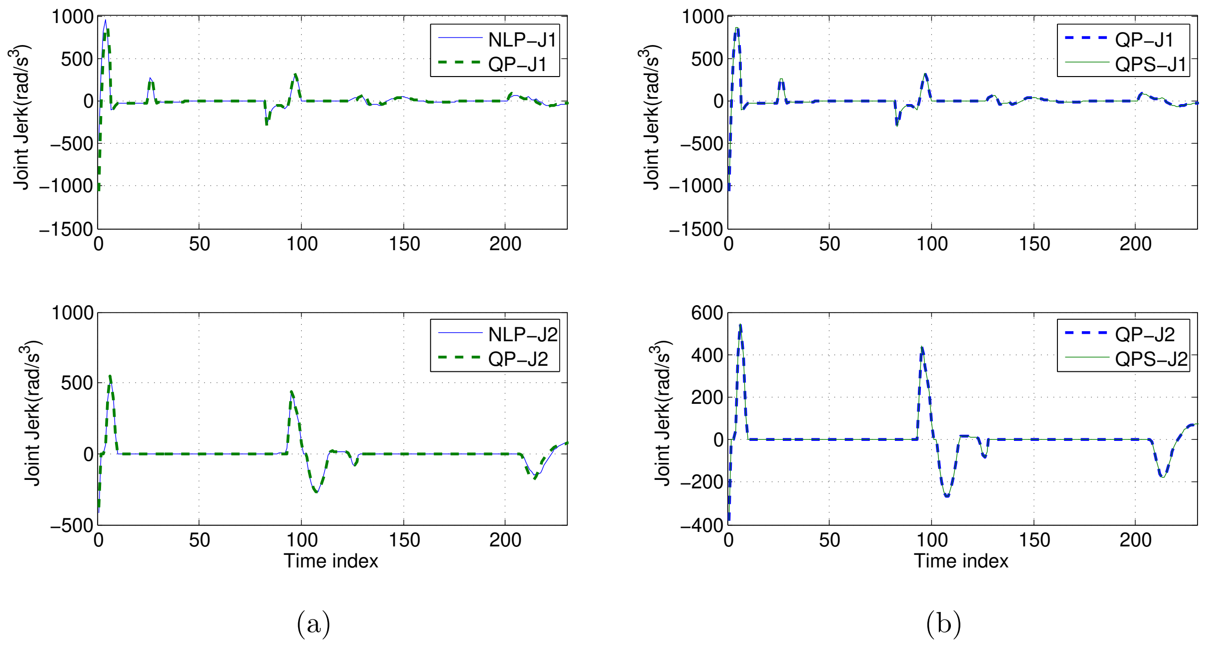


Figure 5.13: The joint jerk (derived from joint accelerations numerically) in simulations of decentralized robot–robot collaboration for the case of the two-link planar robot using NLP, QP, and QPS. (a) The comparison between NLP and QP, and (b) the comparison between QP and QPS.

contributed by the reduction in the average time spent on the Taylor series expansion.

The reduction in the average computation time, however, does not appear in the scenario of decentralized robot–robot collaboration. This is because of the practical implementation issue. In the implementation of (4.21), the forward kinematics and the differential kinematics need to be evaluated a large amount of times in order to obtain the gradient vector around z_k^0. For the formulation of QP, some computational load can be saved, such as the Jacobian matrix in the differential kinematics equation for all pairs of (j, i) in the same time step. In the implementation of QPS, however, such advantages are not present since the index of (j, i, k) is changing in every iteration. As a result, the reduction of the computational load in doing a Taylor series expansion numerically is partially canceled by the increase of the computational load in computing the forward kinematics and the differential kinematics. This negative effect becomes larger when the difference in the number of inequality constraints between QP and QPS is not large.

Table 5.2: Size of the optimization problem in simulations of the two-link planar robot using NLP, QP, and QPS and the computation time in simulations.

	Human–robot collaboration			Decentralized Robot–robot collaboration		
Formulation of optimization problem	NLP	QP	QPS	NLP	QP	QPS
# of variables	96					
# of equality constraints	64					
# of inequality constraints	16	512	10∼28	16	128	20∼26
# of iterations	334	337	334	238	236	236
Average solving time (sec)	6.2853	0.0096	0.0069	5.2604	0.0082	0.0080
Average time of Taylor series expansion (sec)	-	0.0036	0.0013	-	0.0010	0.0011

Table 5.3: Parameters of the optimization problem in simulations of the ITRI seven-DOF robot manipulator.

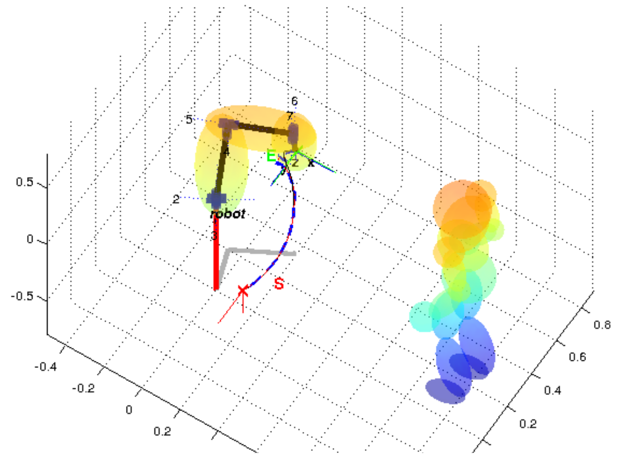
k_{dsi}	k_{msi}	k_p	k_{po}	k_{pns}	k_u	ϵ	S_a	Δt
1	25	$20N/k_{po}$	0.605 (m^2)	0.04	0.0025	0.5	1	$0.01(sec)$

5.3 ITRI Seven-DOF Robot Manipulator

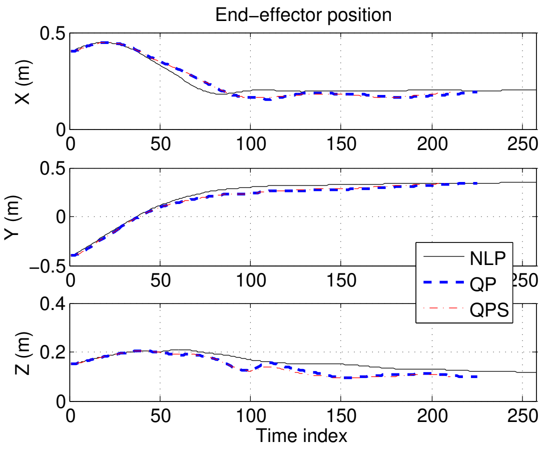
The parameters of the optimization problem used in the simulations of the ITRI seven-DOF robot are listed in Table 5.3. The normalizing factor, k_{po}, is 0.605 m^2 in simulations of the ITRI seven-DOF robot manipulator. The weighting factor on the orientation of the end-effector, k_{pns}, is set to 0.04. Based on the assumption made in Section 2.3, T_c, is selected as three time steps of the robot controller in human–robot collaboration and one time step in decentralized robot–robot collaboration. The planning horizon N is set as 12 time steps in both scenarios of robot collaborations.

5.3.1 Human–Robot Collaboration

The resulting paths of the robot end-effector in simulations using NLP, QP, and QPS are shown in isometric view in Figure 5.14a, and paths along the X-, Y-, and Z-axis in task space are shown in Figure 5.14b. The joint positions of the generated trajectories are shown in Figure 5.15. From Figure 5.14b and Figure 5.15, one can find that the trajectory generated



(a)



(b)

Figure 5.14: The resulting end-effector paths in simulations of human–robot collaboration for the case of the ITRI seven-DOF robot using NLP, QP, and QPS. (a) The path in isometric view, and (b) the paths along the X-, Y-, and Z-axis.

by QPS is very close to the trajectory of using QP. These two trajectories, however, are different from the trajectory generated by NLP, especially when the robot end-effector is getting close to the goal position E. The difference starts around the 60th time step and can also be seen in the generated joint positions in Figure 5.15. A possible reason is the improper trust bound ϵ in the approximation in (4.37) around that joint position and velocity where the trajectory of QP starts to deviate from the trajectory of NLP. As a result, the robot manipulator in simulations of using QP and QPS deviate but still reaches the goal position. It takes 258 time steps for the robot manipulator to reach the goal position in the simulation of using NLP, which is longer than 225 time steps in simulations using the other two formulations.

The snapshots of the simulation using QP are shown in Figure 5.17. From Figure 5.17a to Figure 5.17d, the robot manipulator is moving toward the goal position and meanwhile slightly detouring because of the increasing momentum safety index caused by the human motion. During the time period when the human worker slows down and then pauses his/her motion, as shown from Figure 5.17d to Figure 5.17f, the robot manipulator keeps moving toward the goal position and makes sure the safety index is equal to or smaller than S_a. In this period, the robot manipulator is physically close to the human worker, and hence the safety index is principally determined by the distance safety index. After Figure 5.17f, the human worker moves back to his/her initial gesture, and the robot manipulator reaches the goal position.

The overall safety index in simulations of using NLP, QP, and QPS is kept less than or equal to S_a as shown in Figure 5.16a and Figure 5.16b. It implies that there is no collision between the robot manipulator and the human worker. The safety index in the simulation

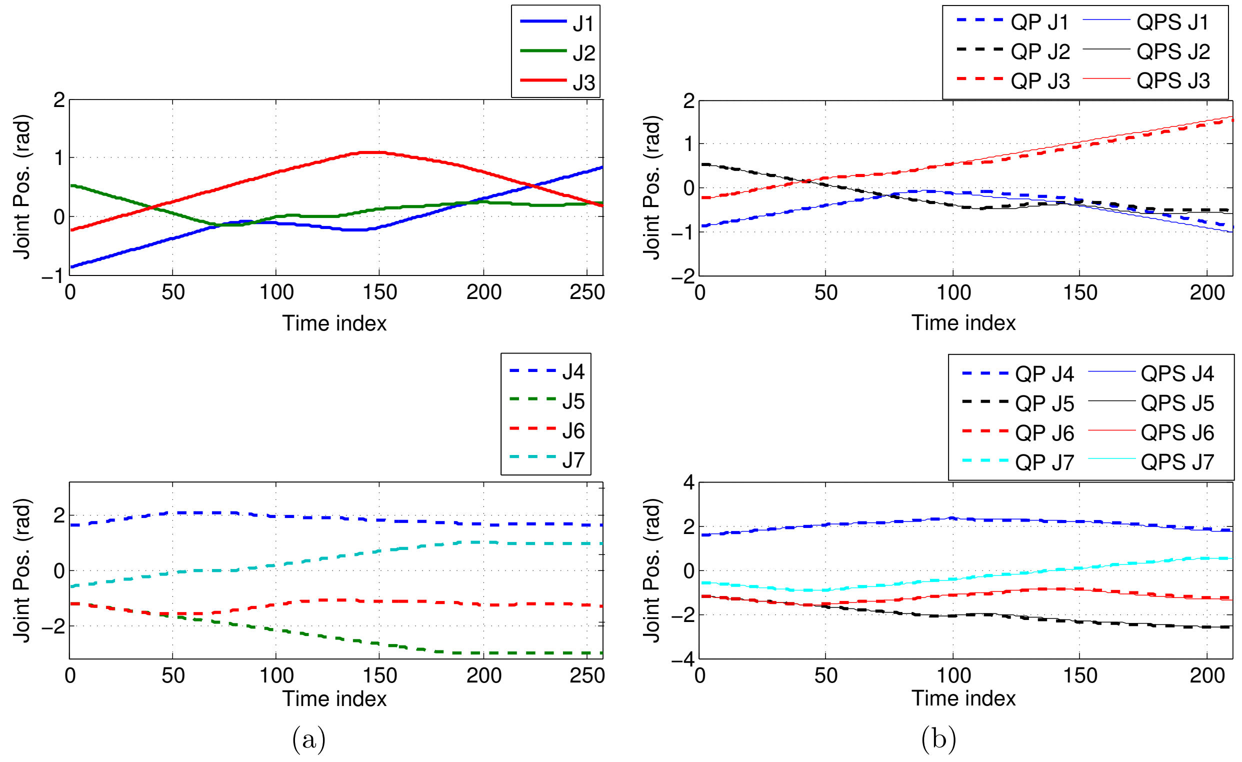


Figure 5.15: The resulting joint positions generated by NLP, QP, and QPS in simulations of human–robot collaboration for the case of the ITRI seven-DOF robot manipulator. (a) NLP, and (b) comparison between QP and QPS.

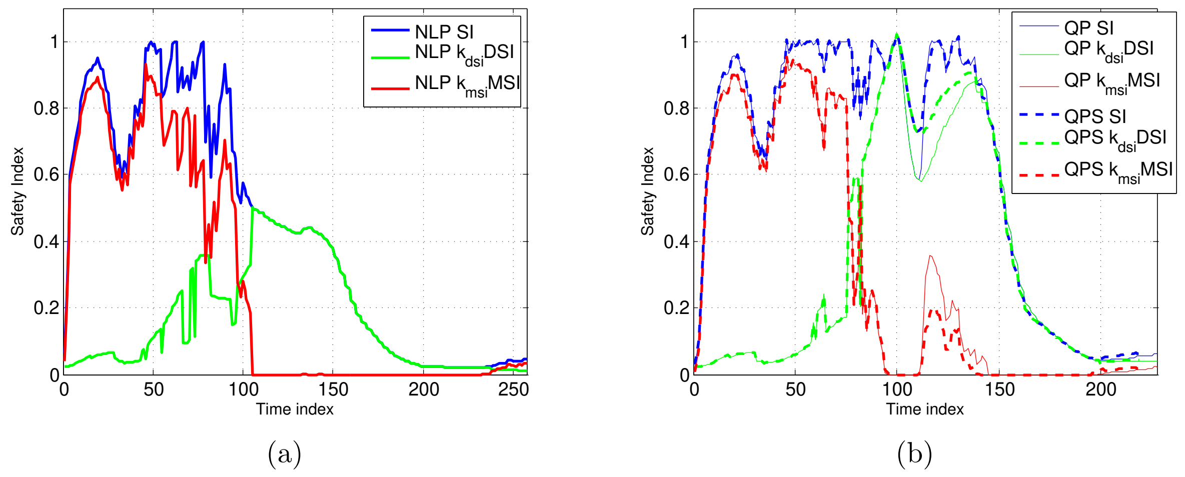
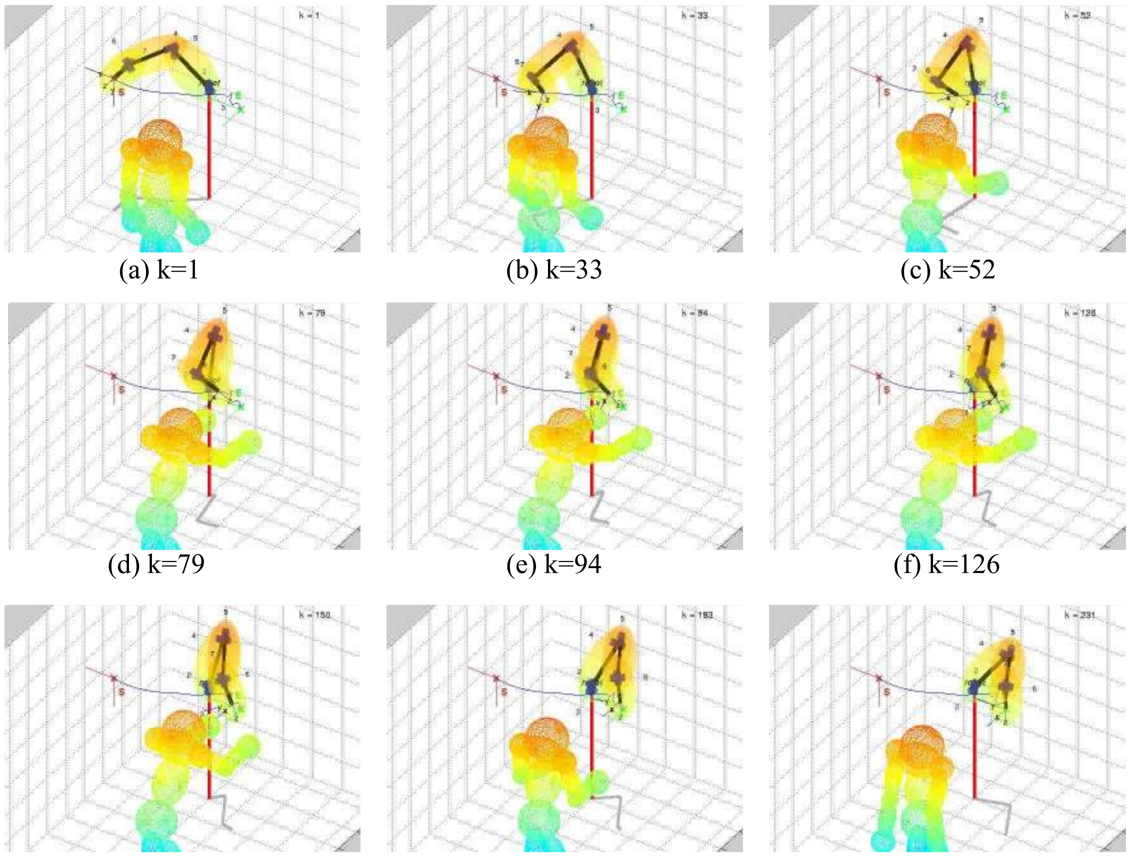


Figure 5.16: The resulting safety indices in simulations of human–robot collaboration for the case of the ITRI seven-DOF robot manipulator using NLP, QP, and QPS. (a) NLP, and (b) the comparison between QP and QPS.


(a) k=1 (b) k=33 (c) k=52

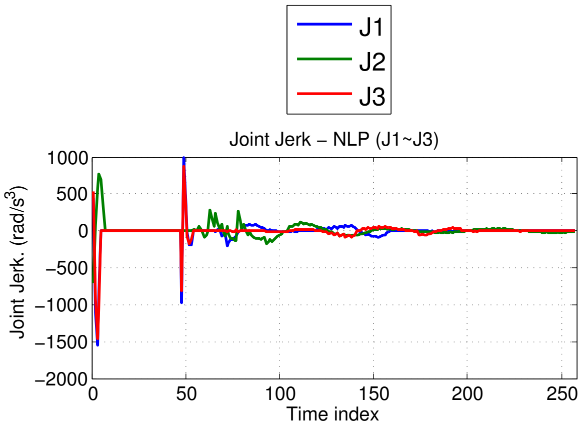
(d) k=79 (e) k=94 (f) k=126

(g) k=150 (h) k=183 (i) k=236

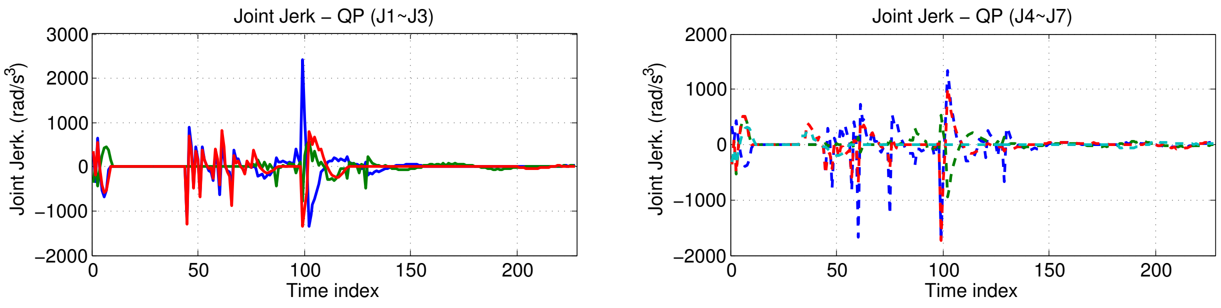
Figure 5.17: Snapshots of the simulation using QP in human–robot collaboration for the case of the ITRI seven-DOF robot manipulator.

using QPS, however, slightly exceeds S_a, at time indices 100 and 101, and the values are 1.0217 and 1.0068, respectively. This may imply that the selection strategy does not perfectly pick up all the active inequality constraints.

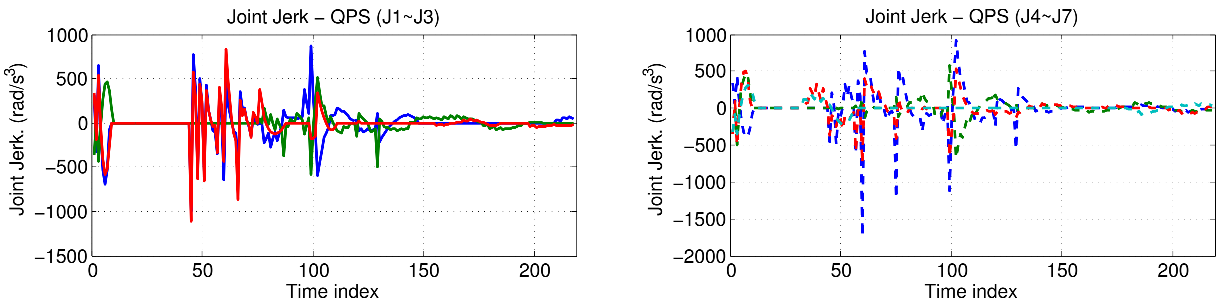
The numerically derived joint jerks from the trajectories generated by NLP, QP, and QPS are shown in Figure 5.18. It is seen that in the simulation of using NLP, there exist several peaks in the joint jerk, but all of the peaks range from 1000 rad/s^3 to -1500 rad/s^3. In simulations of using QP, an undesired peak which is greater than 2000 rad/s^3 appears in the time step of 100. Other than that, the joint jerk is bounded. However, the joint jerk in the QPS simulation in Figure 5.18c does not exhibit this peak. From the figures of the safety index and the derived joint jerk, it can be concluded that the selection of the inequality constraints does not work perfectly in the time step around 100. As a result, the robot manipulator in the simulation using QPS proceeds less safely but with a smaller squared joint jerk direction.



(a)

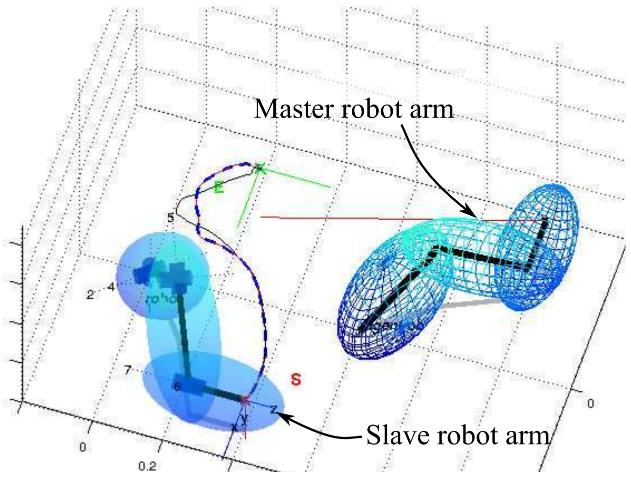


(b)

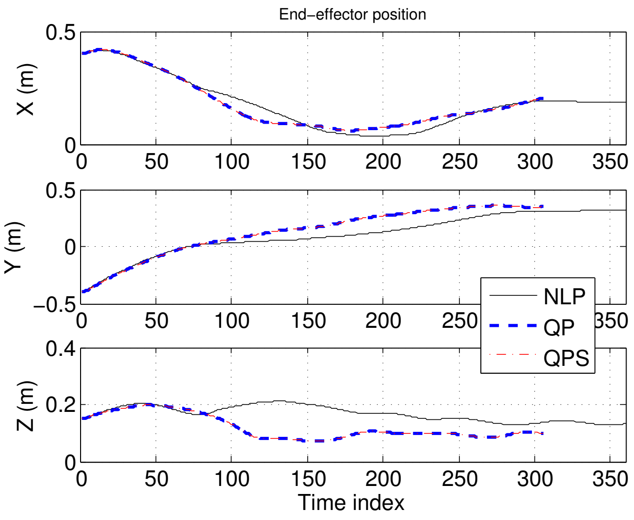


(c)

Figure 5.18: The joint jerks (derived from joint accelerations numerically) in simulations of human–robot collaboration for the case of the ITRI seven-DOF robot using NLP, QP, and QPS. (a) NLP, (b) QP, and (c) QPS.



(a)



(b)

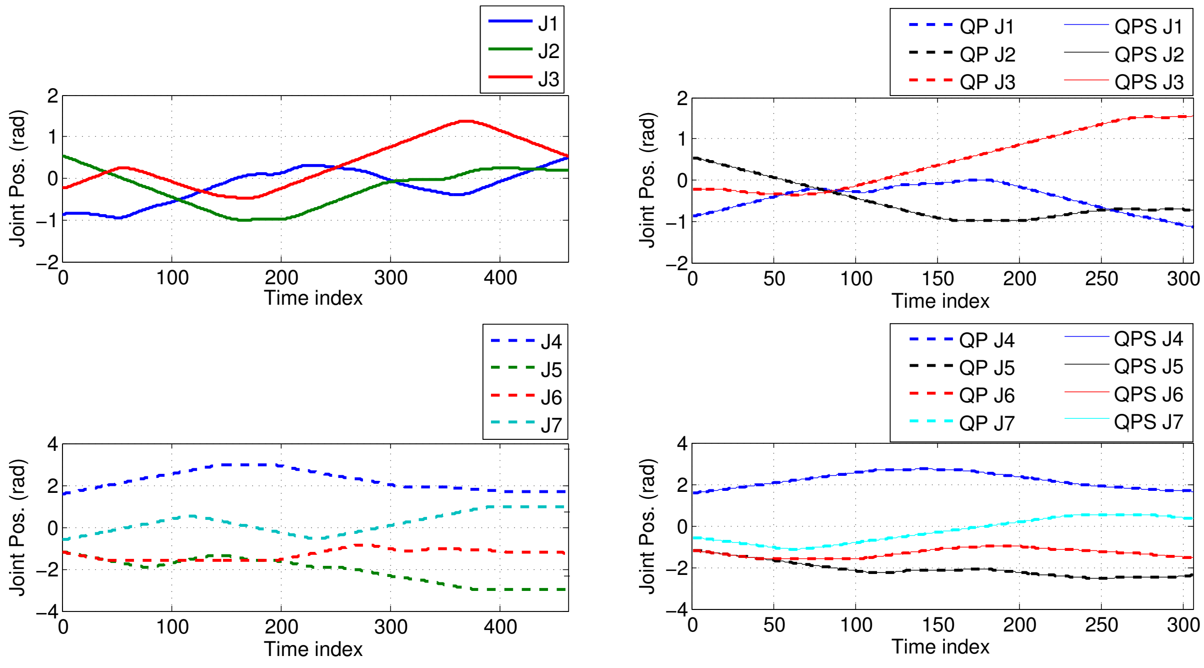
Figure 5.19: The resulting end-effector paths in simulations of decentralized robot–robot collaboration for the case of the ITRI seven-DOF robot. (a) The path in isometric view, and (b) the paths along the X-, Y-, and Z-axis.

5.3.2 Decentralized Robot–Robot Collaboration

The simulation results of decentralized robot–robot collaboration of two ITRI seven-DOF robots using formulations of NLP, QP, and QPS, are shown in Figure 5.19 through Figure 5.23. The resulting paths of the robot end-effector in simulations of NLP, QP, and QPS are shown in isometric view in Figure 5.19a, and paths along the X-, Y-, and Z-axis in task space are shown in Figure 5.19b. In Figure 5.19, the robot manipulator enveloped by the half-transparent ellipsoids is the slave robot manipulator, and the master robot manipulator is enclosed by the meshed ellipsoids. In Figure 5.19b, the path of the end-effector generated by NLP is different from the path generated by QP or QPS, and the paths of using QP and QPS are almost identical to each other. The same observation can be drawn from the joint position in the three simulations in Figure 5.20, in which Figure 5.20a is the joint position generated by NLP and Figure 5.20b shows the comparison between QP and QPS. The considerable similarity between the trajectories of QP and QPS indicates that the selection strategy of the inequality constraints works successfully. Furthermore, all formulations provide the guarantee of a safe motion and the safety index in the three simulations are less than or equal to the threshold S_a, as shown in Figure 5.21.

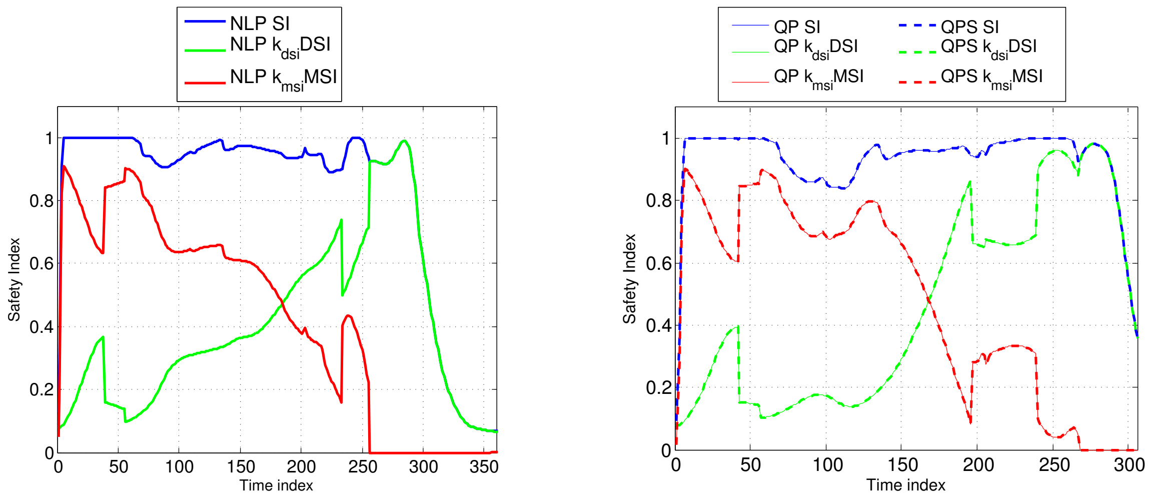
Figure 5.23 is snapshots of the simulation using the formulation of QP. The slave robot manipulator, as shown in the snapshots, is making a curved detour to avoid the master robot manipulator and moving toward the goal position at the same time.

Figure 5.22 shows the joint jerk derived from the joint acceleration generated in the three formulations. The derived joint jerk in three simulations is bounded between 1000 rad/s^3



(a) (b)

Figure 5.20: The resulting joint positions in simulations of decentralized robot–robot collaboration for the case of the ITRI seven-DOF robot manipulator using NLP, QP, and QPS. (a) NLP, and (b) comparison between QP and QPS.



(a) (b)

Figure 5.21: The resulting safety indices in simulations of decentralized robot–robot collaboration for the case of the ITRI seven-DOF robot manipulator using NLP, QP, and QPS. (a) NLP, and (b) the comparison between QP and QPS.

Table 5.4: Size of optimization problem in simulations of the ITRI seven-DOF robot manipulator using NLP, QP, and QPS and the computation time in simulations.

	Human–robot collaboration			Decentralized Robot–robot collaboration		
Formulation of optimization problem	NLP	QP	QPS	NLP	QP	QPS
# of variables	252					
# of equality constraints	168					
# of inequality constraints	12	1440	0∼58	12	216	10∼60
# of iterations	258	228	219	360	306	306
Average solving time (sec)	44.9334	0.0938	0.0298	19.912	0.0318	0.0282
Average time of Taylor series expansion (sec)	-	0.0718	0.0209	-	0.0171	0.0125

to -1000 rad/s^3. Furthermore, the jerk is bounded by 500 rad/s^3 and -500 rad/s^3 except in the beginning period of time. The bounded joint jerk implies that the three trajectories generated by the three formulations are smooth.

5.3.3 Computation Time

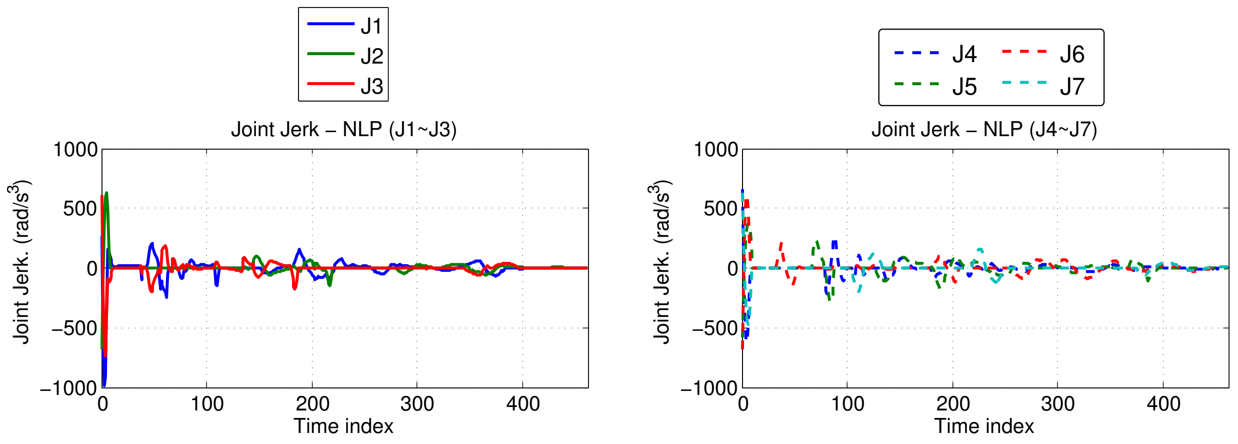
The sizes of the optimization problem in the three formulations and the average computation time for each iteration in simulations of the ITRI seven-DOF robot manipulator are summarized in Table 5.4. The first three rows show the size of the optimization problem in the three formulations, and the fourth row shows the time steps for the robot manipulator to reach the goal position. The average computation time of using QP is 0.0938 sec and 0.0318 sec in human–robot collaboration and decentralized robot–robot collaboration, respectively. They are drastically smaller than the average time in the simulation of using NLP. By using the strategy to select the inequality constraints, the average computation time decreases from 0.0938 sec to 0.0298 sec in the case of human–robot collaboration, which is a 68.2% decrease. As shown in the last row of Table 5.4, the reduction in time is mainly due to the reduction in the average time spent on the Taylor series expansion, which reduces from 0.0718 sec to 0.0209 sec. In the scenario of decentralized robot–robot collaboration with the ITRI seven-DOF robot, the average time for each iteration only drops to 0.0282 sec, which is a 11.3% decrease. This is attributed to the practical issue in implementation as stated in the case of the two-link planar robot manipulator.

5.4 Remarks

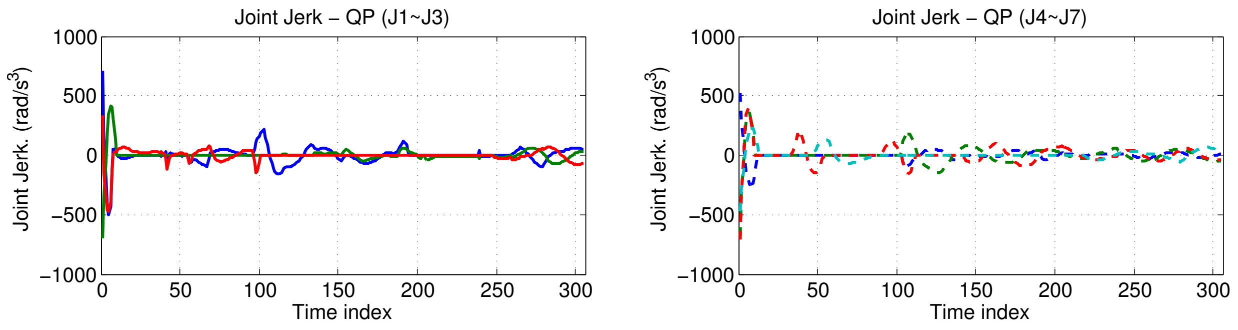
In this chapter, the proposed online trajectory generation algorithm along with the robot safety system proposed in Section 2.2 were implemented and simulated. Three formulations of the optimization problems were implemented in the online trajectory generation algorithm, including the original non-convex and nonlinear optimization problem, represented by NLP, and the quadratic optimization problem with/without the selection strategy to the inequality constraints, represented by QP and QPS. The quadratic optimization problem, QP, is used to approximate the NLP into a quadratic problem in joint space. The selection strategy of the inequality constraints is to further reduce the computational load in QP due to the numerical evaluation of Taylor series expansion.

The resulting generated trajectories in the simulations show that QP and QPS can effectively approximate NLP with much less computational load in the case of the two-link planar robot. In the case of the ITRI seven-DOF robot manipulator, the trajectories generated by using QP and QPS are different from the trajectory generated by using NLP. Yet QP and QPS provide the equivalent performance in terms of ensuring the safety of the agent in the shared workspace and leading the robot manipulator to its goal position. In other words, the three formulations guarantee the online generation of a collision-free trajectory. Because of the J_u term in the optimization problem, the trajectories generated by these three formulations all have a bounded joint jerk and are smooth.

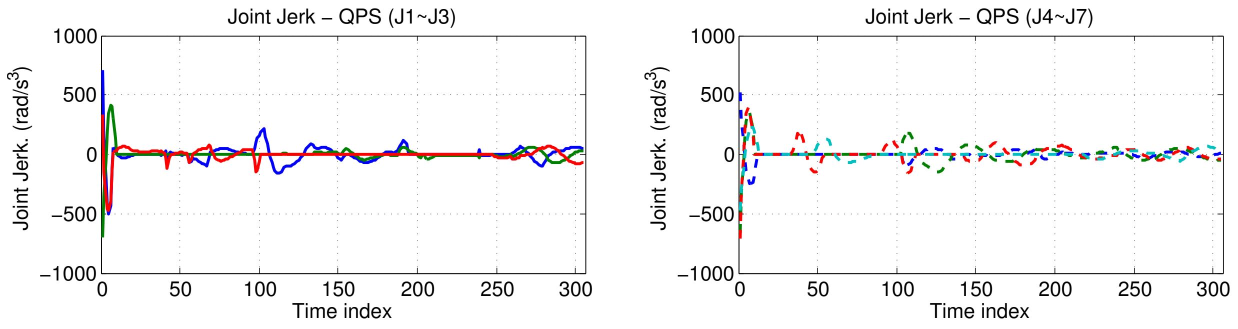
Although this selection strategy is a heuristic method, it works successfully in most of the cases in simulation validation, especially when the number of inequality constraints is large. With the selection strategy, the average computation time was greatly reduced. Regarding the feasibility of the real-time implementation, the simulation results in the case of the two-link planar robot show that it is feasible to achieve real-time trajectory generation. In the case of the ITRI seven-DOF robot manipulator, however, the average computation time is still too long. A big portion of the average computation time is for computing the Taylor series expansion numerically.



(a)

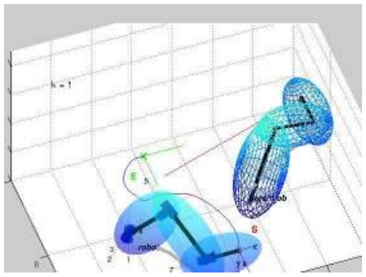


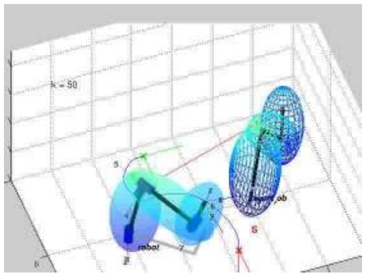
(b)

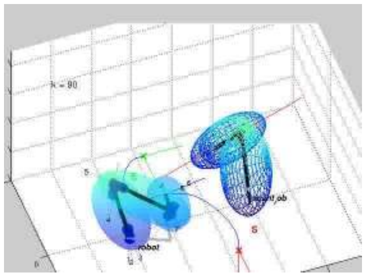


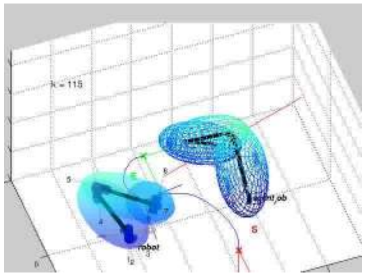
(c)

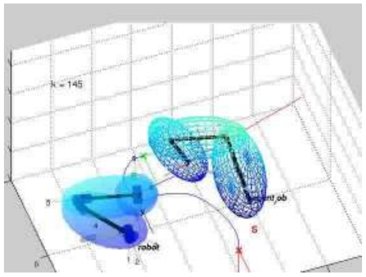
Figure 5.22: The joint jerks (derived from joint accelerations numerically) in simulations of decentralized robot–robot collaboration for the case of the ITRI seven-DOF robot using (a) NLP, (b) QP, and (c) QPS.

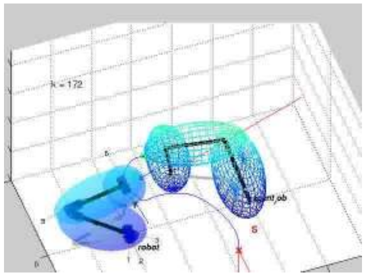

(a) k=1

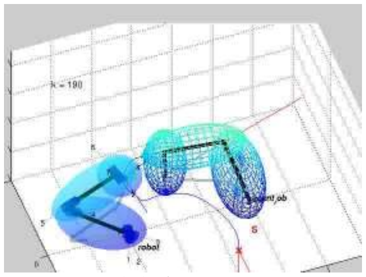

(b) k=50

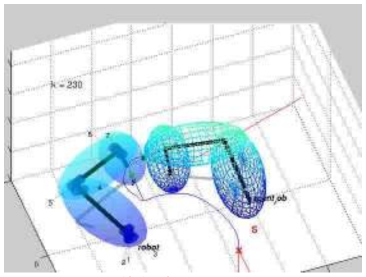

(c) k=90


(d) k=115


(e) k=145


(f) k=172


(g) k=190


(h) k=230

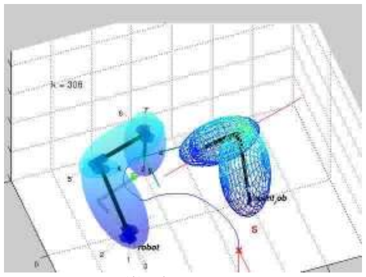

(i) k=306

Figure 5.23: The snapshots of the simulation of decentralized robot–robot collaboration for the case of the ITRI seven-DOF robot using QP.

Chapter 6

Measurement Noise in the Agent Information

6.1 Introduction

In the development of the online trajectory generation algorithm in Chapter 4, a constant velocity model built from the current and previous measurements is used to describe a dynamic model of the agent ellipsoids. However, a noisy measurement of the agent position information is generally inevitable, especially in sensing the human worker in human–robot collaboration. In this chapter, the situation that the position information of agent ellipsoids contains measurement noise is considered. Specifically, a Gaussian random vector is introduced to represent the measurement noise in the velocity level when sensing agents in the shared workspace. Two approaches are proposed in the robot safety system to handle the contaminated agent information. The first one is to consider the measurement noise in the agent position estimation block, and then this block outputs the best estimate of the agent position to the decision making block. The other approach is to handle the measurement noise in the formulation of the optimization problem in the decision making block. The details of the two approaches are stated in the subsequent sections.

6.2 Handling Measurement Noise in Agent Position Estimation Block

In the proposed robot safety system in Figure 2.1, the function of the agent position estimation block is to transform the sensor measurements into a set of ellipsoids which can effectively envelop the space occupied by the agent. Based on the information of the agent ellipsoids, the decision making block generates a collision-free trajectory which leads the robot manipulator toward the goal position. Thus, an intuitive approach is to handle the noisy measurement in the agent position estimation block. This block then outputs the best estimate based on the current observation. The estimated output positions of agent ellipsoids are treated as deterministic, and the online trajectory generation algorithm in the decision making block generates a collision-free trajectory based on the estimated position. The benefit of this approach is that no change is needed for the decision making block, and the online trajectory generation stated in Chapter 4 can be directly used.

To estimate the centers of the agent ellipsoids, kinematic kalman filter (KKF) can be utilized in the agent position estimation block. The dynamic model of the agent ellipsoids can be written as,

$$\begin{bmatrix} x_{oi(n)} \\ v_{oi(n)} \end{bmatrix} = \begin{bmatrix} I_{N_x} & (\Delta t)T_c \\ 0 & I_{N_x} \end{bmatrix} \begin{bmatrix} x_{oi(n-1)} \\ v_{oi(n-1)} \end{bmatrix} + B_\nu \nu_{(n-1)}, \tag{6.1a}$$

$$y_{oi(n)} = \begin{bmatrix} I_{N_x} & \mathbf{0} \end{bmatrix} \begin{bmatrix} x_{oi(n)} \\ v_{oi(n)} \end{bmatrix} + \omega_{(n)}, \tag{6.1b}$$

where $x_{oi(\bullet)}$ and $v_{oi(\bullet)}$ are the task space position and velocity of the i-th agent ellipsoid center, $y_{oi(\bullet)}$ is the measurement received from the sensor block, I_{N_x} is an identity matrix with size of N_x by N_x, N_x is the dimension of the task space, and Δt is the sampling time for the controller. Measurement is available every T_c time step, and n represents the n-th measurement instance. The n-th measurement instance corresponds to the (kT_c)-th robot controller time step, and the $(n-1)$-th to the $((k-1)T_c)$-th robot controller time step. The vector $\nu_{(\bullet)} \in \mathbb{R}^{N_x}$ is a Gaussian random vector, and $B_\nu \nu_{(\bullet)}$ accounts for the dynamics which could not be represented by the constant velocity model. In (6.1b), the vector $\omega_{(\bullet)}$ is the measurement noise in the Gaussian distribution, which is the part we want to handle. The Kalman Filter method is applied to (6.1) to estimate the task space position and velocity of the centers of the agent ellipsoids.

6.3 Online Trajectory Generation with Measurement Noise in Agent Ellipsoids

The second approach is to handle the measurement noise directly in the online trajectory generation algorithm in the decision making block. The information of the agent ellipsoids is contaminated by the measurement noise, which is represented by a Gaussian random vector, and hence the dynamic model of the agent ellipsoids becomes stochastic. As a result, the inequality constraints accounting for the safety of the agents in the optimization problem in (4.10) become stochastic. These constraints need to be handled in a stochastic framework as inequality chance constraints. The details of the chance constraints are stated below.

6.3.1 Stochastic Agent Model in the Formulation of the Optimization Problem

The stochastic dynamic model of the agent ellipsoids is formulated by adding a Gaussian random vector to the deterministic constant-velocity model in Section 4.2 as

$$x_{oi(k+1)} = x_{oi(k)} + (\Delta t)(v_{oi} + w_{i(k)}),\ k = 0, \dots, N-1 \tag{6.2}$$

where

$$w_{i(k)} \sim \mathcal{N}(0, \Sigma).$$

The index k indicates the k-th time step of the planning horizon. The vector $x_{oi(\bullet)}$ is the task space position of the i-th agent ellipsoid center. N is the planning horizon. $w_{i(k)}$ is a Gaussian random vector representing the measurement noise in the velocity of the i-th agent ellipsoid center at the k-th time step. The matrix $\Sigma \in \mathbb{R}^{N_x \times N_x}$ is the covariance matrix of the Gaussian random vector. v_{oi} is computed from the previous and current position measurements as,

$$v_{oi} = \frac{1}{T_c \Delta t}(x_{oi(0)} - x_{oi(-T_c)}), \tag{6.3}$$

where $x_{oi(0)}$ and $x_{oi(-T_c)}$ are the current and previous position measurements. The vector $x_{oi(k)}$ is a random variable with normal distribution and can be expressed as,

$$x_{oi(k)} \sim \mathcal{N}\Big(x_{oi(0)} + (k\Delta t)v_{oi},\ k(\Delta t)^2\Sigma\Big),\ \forall k = 1, \dots, N. \tag{6.4}$$

With this stochastic agent model, the inequality constraints accounting for the safety of the agents in the optimization problem in (4.10c) becomes non-deterministic. Thus, the inequality constraints need to be handled in a stochastic framework as inequality chance constraints as:

$$P\big(SI(z_{(k)}) \leq S_a\big) \geq 1 - \alpha, \forall k = 1, \dots, N. \tag{6.5}$$

Equation (6.5) indicates that the probability of being safe, i.e., $SI(z_{(k)}) \leq S_a$, is greater than or equal to a predefined value, $1-\alpha$. The value of α should be very small, such as 0.05 or 0.01.

In the remainder, candidate 2 defined in (3.16) is used as the overall safety index. In Section 4.4, the safety inequality constraint built from candidate 2 is shown to be equivalent to the inequality constraints in (4.19). Hence the inequality chance constraint in (6.5) becomes

$$P\big(k_{dsi}DSI_{ji}(z_{(k)}) \leq S_a\big) \geq 1 - \alpha,\ \forall j, i, k = 1, \dots, N, \tag{6.6a}$$

$$P\big(k_{msi}\overline{MSI}_{ji}(z_{(k)}) \leq S_a\big) \geq 1 - \alpha,\ \forall j, i, k = 1, \dots, N, \tag{6.6b}$$

where $\overline{MSI}_{ji}(z_{(k)})$ is defined in (4.17).

In order to implement the inequality chance constraints, these chance constraints are transformed to deterministic inequality constraints by utilizing the mathematical similarity between the probability density function (PDF) of a Gaussian random vector and the agent ellipsoid. In the next section, a simple one-dimension case is used to illustrate the transformation between the PDF of a Gaussian random vector and an ellipsoid.

6.3.2 Geometric Interpretation of a Gaussian Random Variable with Confidence Interval

One-Dimensional Space

From statistics we know that the sum of squared independent random variables with standard normal distributions is a random variable with a chi-squared distribution. The degree-of-freedom of the chi-squared random variable is the only parameter of this distribution and is determined by the number of the standard normal random variables used to construct it. For instance, $r \in \mathbb{R}$ is a random variable with normal distribution, $\mathcal{N}(\mu, \sigma^2)$, and a chi-squared random variable ρ with the degree-of-freedom of one can be constructed as

$$\frac{(r-\mu)^2}{\sigma^2} = \rho \sim \chi_1^2. \tag{6.7}$$

The value of ρ is always positive. From the cumulative distribution function (CDF) of χ_1^2, we can compute the probability for any given positive number ρ_s,

$$F_{\chi_1^2}(\rho_s) = P\big(\rho \le \rho_s\big) = P\Big(\frac{(r-\mu)^2}{\sigma^2} \le \rho_s\Big) = p_s, \tag{6.8}$$

where $F_{\chi_1^2}$ is the CDF of χ_1^2. Since that CDF is always a monotonic function, for any positive ρ_s, there exists a unique p_s, and vice versa. With some algebraic manipulations of the inequality, $(r-\mu)^2/\sigma^2 \le \rho_s$, (6.8) is equivalent to,

$$P\Big(\mu - \sqrt{\rho_s}\sigma \le r \le \mu + \sqrt{\rho_s}\sigma\Big) = p_s. \tag{6.9}$$

Equation (6.9) can be graphically illustrated by the PDF of r as shown in Figure 6.1. In the figure, the shadow area represents not only the probability of the random variable r staying between $(\mu - \sqrt{\rho_s}\sigma)$ and $(\mu + \sqrt{\rho_s}\sigma)$, but also the probability of the random variable ρ less than or equal to ρ_s. In other words, the area of the shadow zone is p_s. This shadow zone extends over a closed interval, which is symmetric to the mean of the normal distribution, μ, and the width of the closed interval is $2\sqrt{\rho_s}\sigma$. Since it is known that p_s has a one-to-one mapping with ρ_s from the CDF of χ_1^2, the area of the shadow zone, p_s, directly depends on ρ_s. The mean and the variance of the random variable r only determine the shape and the width of the shadow area without changing the area of the shadow zone.

In conclusion, for any one-dimensional closed interval, a random variable with a normal distribution can be found such that this random variable stays inside this interval with a certain probability. For instance, given a desired probability p_d and a closed interval, $[r_a, r_b]$,

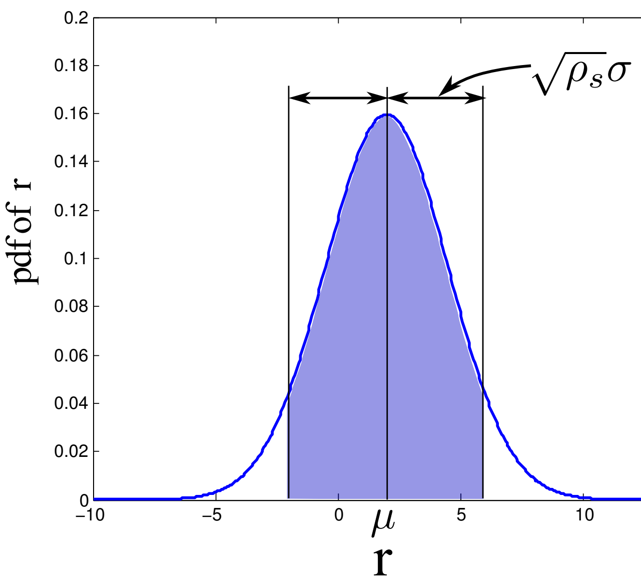


Figure 6.1: An illustration of $\rho \sim \chi_1^2$ in the probability density function (PDF) of a normal random variable, $r \sim \mathcal{N}(\mu, \sigma^2)$.

where $r_b > r_a$, the mean μ and the variance σ^2 of the normal random variable can be obtained by

$$\mu = (r_a + r_b)/2, \tag{6.10a}$$
$$\sigma^2 = (r_b - r_a)^2/(4\rho_s), \tag{6.10b}$$
$$\rho_s = F_{\chi_1^2}^{-1}(p_d). \tag{6.10c}$$

On the contrary, given a Gaussian random variable with mean μ and variance σ^2 of a Gaussian random variable and a desired probability p_d, the corresponding closed interval, $[r_a, r_b]$, can also be computed by

$$r_b = \mu + \sqrt{\rho_s}\sigma \tag{6.11a}$$
$$r_a = \mu - \sqrt{\rho_s}\sigma \tag{6.11b}$$
$$\rho_s = F_{\chi_1^2}^{-1}(p_d) \tag{6.11c}$$

Higher Dimension Space

The same concept can be easily extended to the case of a higher dimensional space. Assume $r_i \in \mathbb{R}^{N_x}$ is a random vector with a normal distribution, $\mathcal{N}(\mu_i, \Sigma_i)$, in task space. Then a chi-squared random variable ρ_i can be constructed from r_i as

$$(r_i - \mu_i)^T \Sigma_i^{-1} (r_i - \mu_i) = \rho_i \sim \chi_{N_x}^2. \tag{6.12}$$

Given a positive number ρ_s, we know from the CDF of $\chi^2_{N_x}$ that

$$\begin{aligned} F_{\chi^2_{N_x}}(\rho_s) &= P\big(\rho_i \le \rho_s\big) \\ &= P\big((r_i-\mu_i)^T\Sigma_i^{-1}(r_i-\mu_i) \le \rho_s\big) = p_s. \end{aligned} \tag{6.13}$$

Equations (6.12) and (6.13) are equivalent to (6.7) and (6.8) for a higher dimensional space. It can be observed that the part inside the probability in (6.13) is written in the same form as an ellipsoid. Hence (6.13) also represents the probability of r_i staying inside an ellipsoid which is defined as

$$\mathcal{E}_i = \{x \in \mathbb{R}^{N_x} \mid (x-\mu_i)^T(\rho_s\Sigma_i)^{-1}(x-\mu_i) \le 1\}, \tag{6.14}$$

where μ_i and $(\rho_s\Sigma_i)^{-1}$ are the center and the matrix of the ellipsoid $\mathcal{E}_i$. It implies that a closed interval in a higher dimensional space, such as the space of dimension N_x, is an ellipsoid.

Hence, for an agent ellipsoid with matrix P_i and centered at x_{oi}, we can find a normal random vector r_i such that the probability of r_i staying inside the agent ellipsoid is p_d. The mean vector and the covariance matrix of the random vector r_i can be obtained by

$$\mu_x = x_{oi}, \tag{6.15a}$$
$$\Sigma_i = (\rho_s P_i)^{-1}, \tag{6.15b}$$

where

$$\rho_s = F^{-1}_{\chi^2_{N_x}}(p_d). \tag{6.16}$$

6.3.3 Chance Constraints of DSI

The chance constraint accounting for the distance safety index in (6.6a) can be transformed into deterministic inequality constraints by applying the concept introduced in the previous section. By utilizing the geometric interpretation of a Gaussian random vector given a probability p_d, each agent ellipsoid can be interpreted as a Gaussian random vector. For instance, given a probability p_d, a Gaussian random vector, r_i, can be built for the i-th agent ellipsoid. The mean and the covariance matrix of the random vector r_i can be computed by (6.15). The mean is at the agent ellipsoid center, and the covariance matrix is the inverse of the product of the i-th agent ellipsoid matrix and ρ_s. ρ_s is computed from the inverse of the CDF of $\chi^2_{N_x}$ evaluated at p_d. The random vector r_i is guaranteed to stay within the i-th agent ellipsoid with the probability of p_d. Furthermore, the position of the center of the i-th agent ellipsoid in every time step within the planning horizon is another random vector $x_{oi(k)}$ as stated in (6.4). By combining r_i and $x_{oi(k)}$, the i-th agent ellipsoid in each time step within the planning horizon can be represented as,

$$x_{i(k)} \sim \mathcal{N}\big(\mu_{ik(k)},\ \Sigma_{ik(k)}\big),\ \forall k = 1,\dots,N, \tag{6.17}$$

where

$$\mu_{ik(k)} = x_{oi(0)} + (k\Delta t)v_{oi} \tag{6.18a}$$
$$\Sigma_{ik(k)} = k(\Delta t)^2\Sigma + \Sigma_i. \tag{6.18b}$$

The mean of $x_{i(k)}$ is just the estimated position of the i-th agent ellipsoid center at the k-th time step of the planning horizon when no measurement noise is considered. The random vector $x_{i(\bullet)}$ in every time step of the planning horizon is guaranteed to stay within the i-th agent ellipsoid with the probability of p_d, i.e.,

$$P\Big((x_{i(k)} - \mu_{ik(k)})^T P_i (x_{i(k)} - \mu_{ik(k)}) \leq 1\Big) \geq p_d,\ \forall k = 1, \dots, N. \tag{6.19}$$

Note that $x_{i(k)}$ is a point in task space. In Section 3.3, we have defined the squared distance in ellipsoid coordinates from a point to the ellipsoid coordinate origin and the distance safety index for the point. Hence, the squared distance in the j-th ellipsoid coordinate from $x_{i(k)}$ to the ellipsoid coordinate origin can be computed as,

$$d^2_{ji(k)} = \big(x_{i(k)} - x_{cj}(z_{(k)})\big)^T Q_j(z_{(k)})\big(x_{i(k)} - x_{cj}(z_{(k)})\big). \tag{6.20}$$

And the DSI_{ji} for the k-th time step of the planning horizon is,

$$DSI_{ji}(k) = \frac{1}{d^2_{ji(k)}},\ \forall k = 1, \dots, N. \tag{6.21}$$

The ellipsoid matrix $Q_j(z_{(k)})$ in (6.20) is composed of two matrices as

$$Q_j(z_{(k)}) = R_j^T(z_{(k)}) A_j R_j(z_{(k)}) \tag{6.22}$$

where $R_j(z_{(k)})$ is the rotation matrix of the ellipsoid coordinate with respect to the world coordinate. The matrix A_j is a diagonal matrix as

$$A_j = \begin{bmatrix} \lambda_{j1}^2 & 0 & \cdots & 0 \\ 0 & \lambda_{j2}^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_{jN_x}^2 \end{bmatrix}. \tag{6.23}$$

The diagonal elements, $\lambda_{j1}, \dots, \lambda_{jN_x}$, are the radii of the j-th ellipsoid coordinate and are all positive. Substituting $Q_j(z_{(k)})$ in (6.22) into (6.20) yields,

$$\begin{aligned} d^2_{ji(k)} &= \big(x_{i(k)} - x_{cj}(z_{(k)})\big)^T R_j(z_{(k)})^T A_j R_j(z_{(k)})\big(x_{i(k)} - x_{cj}(z_{(k)})\big) \\ &= \Big(A_j^{1/2} R_j(z_{(k)})\big(x_{i(k)} - x_{cj}(z_{(k)})\big)\Big)^T \Big(A_j^{1/2} R_j(z_{(k)})\big(x_{i(k)} - x_{cj}(z_{(k)})\big)\Big) \\ &:= y^T_{ji(k)} y_{ji(k)} \end{aligned} \tag{6.24}$$

The matrix $R_j(z_{(k)})$ and the vector $x_{cj}(z_{(k)})$ are the variables used to describe the spatial relationship between the j-th ellipsoid coordinate and the world coordinate at the k-th time step in the planning horizon. They are deterministic and depend on the joint position and joint velocity, i.e., $z_{(k)}$. The matrix A_j is a constant matrix. Hence the vector $y_{ji(k)}$ is a random vector generated by the random vector $x_{i(k)}$ with a rotation and a translation. $y_{ji(k)}$ is also normally distributed and can be written as

$$y_{ji(k)} \sim \mathcal{N}\big(\mu_{yji(k)},\ \Sigma_{yji(k)}\big),\ \forall k = 1, \ldots, N \tag{6.25}$$

where

$$\mu_{yji(k)} = A_j^{1/2} R_j(z_{(k)})\big(\mu_{ik(k)} - x_{cj}(z_{(k)})\big) \tag{6.26a}$$

$$\Sigma_{yji(k)} = A_j^{1/2} R_j(z_{(k)})\, \Sigma_{ik(k)}\, R_j^T(z_{(k)}) A_j^{1/2}. \tag{6.26b}$$

Plugging the definition of DSI_{ji} in (6.21) into the chance constraint in (6.6a) and replacing $d^2_{ji(k)}$ by $y^T_{ji(k)} y_{ji(k)}$ yields,

$$\begin{aligned}
&P\Big(k_{dsi} DSI_{ji}(z_{(k)}) \leq S_a\Big) \\
&= P\Big(\frac{k_{dsi}}{d^2_{ji(k)}} \leq S_a\Big) \\
&= P\Big(d^2_{ji(k)} \geq \frac{k_{dsi}}{S_a}\Big) \\
&= P\Big(y^T_{ji(k)} y_{ji(k)} \geq \frac{k_{dsi}}{S_a}\Big) \geq 1 - \alpha,\ \forall j, i, k = 1, \ldots, N.
\end{aligned} \tag{6.27}$$

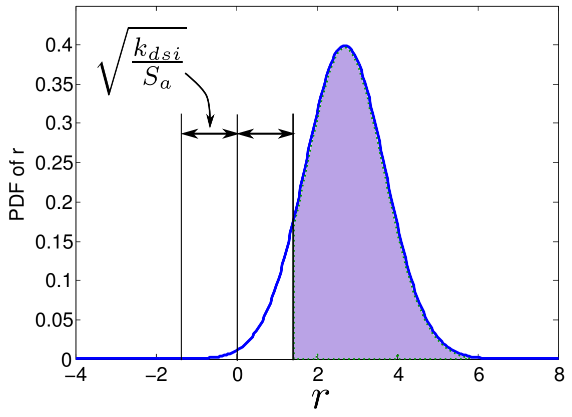
Since k_{dsi} and S_a are all positive, (6.27) is equivalent to

$$P\left(\|y_{ji(k)}\|_2 \geq \sqrt{\frac{k_{dsi}}{S_a}}\right) \geq 1 - \alpha,\ \forall j, i, k = 1, \ldots, N \tag{6.28}$$

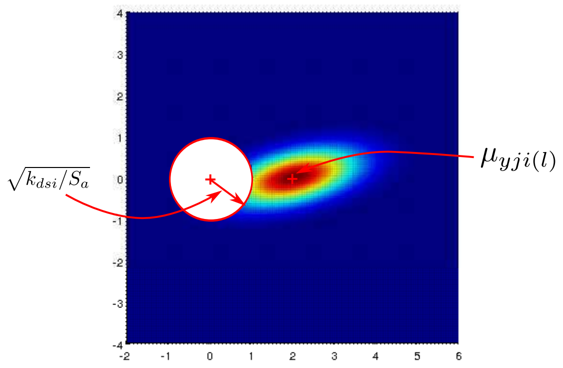
Equation (6.28) implies that the probability for the length of the random vector $y_{ji(k)}$ being greater than or equal to $\sqrt{k_{dsi}/S_a}$ is desired to be greater than or equal to $1 - \alpha$. This is illustrated in cases of a one-dimension random variable and a two-dimension random vector as shown in Figure 6.2. In the two examples, the probability is illustrated by the colored area and is desired to be greater than or equal to $1 - \alpha$.

From (6.27) and (6.28), the safety chance constraint accounting for the distance safety index in (6.6a) can be expressed as another chance constraint. Then, the concept in the previous section is used again to conservatively approximate this chance constraint in (6.28). A chi-squared random variable can be constructed from the normal random vector, $y_{ji(k)}$, as,

$$(y_{ji(k)} - \mu_{yji(k)})^T \Sigma^{-1}_{yji(k)} (y_{ji(k)} - \mu_{yji(k)}) = \rho_{ji(k)} \sim \chi^2_{N_x}. \tag{6.29}$$



(a)



(b)

Figure 6.2: Examples of (6.28) in one-dimensional and two-dimensional space. (a) In one-dimensional space, and (b) in two-dimensional space.

From the CDF of $\rho_{ji(k)}$, we can find one $\rho_{\bar{\alpha}}$ for a given probability $1-\alpha$ such that

$$P\left(\rho_{ji(k)} \leq \rho_{\bar{\alpha}}\right) = 1-\alpha \tag{6.30}$$

Equation (6.30) implies that the random vector $y_{ji(k)}$ stays inside an ellipsoid with the probability of $1-\alpha$. The center and the matrix of the ellipsoid are $\mu_{yji(k)}$ and $(\rho_{\bar{\alpha}}\Sigma_{yji(k)})^{-1}$ respectively, and the ellipsoid can be written mathematically as

$$\mathcal{E}_{ji(k)} = \{x \in \mathbb{R}^{N_x} \mid (x-\mu_{yji(k)})^T(\rho_{\bar{\alpha}}\Sigma_{yji(k)})^{-1}(x-\mu_{yji(k)}) \leq 1\}. \tag{6.31}$$

In other words, an ellipsoid $\mathcal{E}_{ji(k)}$ can be found such that $y_{ji(k)}$ is guaranteed to stay inside $\mathcal{E}_{ji(k)}$ with the probability of $1-\alpha$. Therefore, the chance constraint (6.28) is guaranteed to be true if the ellipsoid $\mathcal{E}_{ji(k)}$ does not overlap the sphere of radius $\sqrt{k_{dsi}/S_a}$. This means that if the shortest distance from the surface of the ellipsoid $\mathcal{E}_{ji(k)}$ to the origin of the j-th ellipsoid coordinate is greater than or equal to $(k_{dsi}/S_a)^{1/2}$, then the chance constraint in (6.28) is true. When the chance constraint in (6.28) is true, it implies that the safety index is guaranteed to be less than or equal to S_a with at least the probability of $1-\alpha$. Thus, the chance constraint in (6.28) is equivalent to

$$d^2_{ji(k),ell} \geq \frac{k_{dsi}}{S_a},\ \forall j,i,k = 1,\ldots,N, \tag{6.32}$$

where the shortest squared distance $d^2_{ji(k),ell}$ can be computed using the method introduced in Section 3.3 as,

$$\begin{aligned} d^2_{ji(k),ell} =& \min_{x_t} \left(x_t - x_{cj}(z_{(k)})\right)^T \left(x_t - x_{cj}(z_{(k)})\right) \\ & s.t. \quad (x_t-\mu_{yji(k)})^T(\rho_{\bar{\alpha}}\Sigma_{yji(k)})^{-1}(x_t-\mu_{yji(k)}) \leq 1. \end{aligned} \tag{6.33}$$

Equation (6.32) is deterministic and the chance constraint for the distance safety index in (6.6a) is guaranteed to be valid when (6.32) is valid.

6.3.4 Chance Constraints of MSI

The chance constraint accounting for the momentum safety index for the stochastic agent model is represented in (6.6b). Plugging the definition of $\overline{MSI}_{ji}(z_{(k)})$ in (4.17) yields,

$$P\Big(k_{msi}\,\mathrm{sign}\big(p_{ji}(z_{(k)})\big)\frac{p_{ji}^2(z_{(k)})}{d_{ji}^2(z_{(k)})} \le S_a\Big) \ge 1-\alpha,\ \forall j,i,k=1,\dots,N. \tag{6.34}$$

The scalar value $p_{ji}(z_{(k)})$ is the projection of the linear momentum vector onto the vector from the origin of the ellipsoid coordinate pointing to the center of the agent ellipsoid. It is defined in (3.11) and can be written using $z_{(k)}$ as

$$p_{ji}(z_{(k)}) = M_j(\dot{x}_{cj}(z_{(k)}) - \dot{x}_{oi(k)})^T x_{uji(k)} \tag{6.35}$$

where

$$x_{uji(k)} = \frac{x_{oi(k)} - x_{cj}(z_{(k)})}{\left\|x_{oi(k)} - x_{cj}(z_{(k)})\right\|_2}. \tag{6.36}$$

From the stochastic dynamic model of agent ellipsoids in (6.2), the measurement noise is represented by $\omega_{i(k)}$ in the velocity level. Hence, the velocity of the i-th agent ellipsoid center, $x_{oi(k)}$, is a Gaussian random vector as

$$\dot{x}_{oi(k)} = v_{oi} + \omega_{i(k)} \ \sim \mathcal{N}\left(v_{oi}, \Sigma\right). \tag{6.37}$$

Since the momentum safety index is designed on the velocity level between the agent ellipsoids and ellipsoid coordinates attached on the robot, the unit vector $x_{uji(k)}$ is considered as a deterministic vector. As a result, $p_{ji}(z_{(k)})$ becomes a Gaussian random variable as

$$p_{ji}(z_{(k)}) \sim \mathcal{N}\left(\mu_{pji(k)}, \sigma_{pji(k)}^2\right), \tag{6.38}$$

where

$$\mu_{pji(k)} = M_j\left(\dot{x}_{cj}(z_{(k)}) - v_{oi}\right)^T x_{uji(k)} \tag{6.39a}$$
$$\sigma_{pji(k)}^2 = x_{uji(k)}^T\,\Sigma\, x_{uji(k)}. \tag{6.39b}$$

Knowing that $p_{ji}(z_{(k)})$ is a random variable, (6.34) can be re-ordered as

$$P\left(\mathrm{sign}\big(p_{ji}(z_{(k)})\big)p_{ji}^2(z_{(k)}) \le S_a\frac{d_{ji}^2(z_{(k)})}{k_{msi}}\right) \ge 1-\alpha,\ \forall j,i,k=1,\dots,N. \tag{6.40}$$

The values of S_a, k_{msi}, and $d_{ji}^2(z_{(k)})$ in the chance constraint are all positive, and the above chance constraint is always valid when $p_{ji}(z_{(k)})$ is negative. It is correct from the perspective of the physical meaning of the momentum safety index because a negative $p_{ji}(z_{(k)})$ implies a safe situation for this pair of (j,i). Thus, the probability in (6.40) is equivalent to

$$P\left(p_{ji}(z_{(k)}) \leq c_{msi}\ \right) \geq 1-\alpha,\ \forall j,i,k=1,\ldots,N, \tag{6.41}$$

where

$$c_{msi} = \sqrt{S_a \frac{d_{ji}^2(z_{(k)})}{k_{msi}}}. \tag{6.42}$$

Since $p_{ji}(z_{(k)})$ is a Gaussian random variable, the probability in the preceding equation can be computed as

$$\begin{aligned} & P\left(p_{ji}(z_{(k)}) \leq c_{msi}\right) \\ & = P\left(\frac{p_{ji}(z_{(k)}) - \mu_{pji(k)}}{\sigma_{pji(k)}} = Z \leq \frac{c_{msi} - \mu_{pji(k)}}{\sigma_{pji(k)}}\right) \\ & = \Phi\left(\frac{c_{msi} - \mu_{pji(k)}}{\sigma_{pji(k)}}\right) \end{aligned} \tag{6.43}$$

where Z is the standard normal random variable and Φ is the CDF of Z. Then (6.41) is equivalent to

$$\frac{c_{msi} - \mu_{pji(k)}}{\sigma_{pji(k)}} \geq \Phi^{-1}\left(1-\alpha\right),\ \forall j,i,k=1,\ldots,N. \tag{6.44}$$

In (6.44), $\Phi^{-1}\left(1-\alpha\right)$ is constant, c_{msi}, $\mu_{pji(k)}$, and $\sigma_{pji(k)}$ are deterministic but depend on $z_{(k)}$. In other words, the chance constraint accounting for the momentum safety index in (6.6b) is interpreted as a deterministic inequality constraint.

6.3.5 Formulation of the Optimization Problem

The chance constraints accounting for the safety index in (6.6) are transformed into the two deterministic constraints, (6.32) and (6.44). The formulation of the optimization problem for the stochastic agent model, thus, becomes

$$\min_{U_k,Z_k} \quad k_p J_p(z_{(N)}) + k_u \sum_{k=0}^{N-1} J_u(u_{(k)}, u_{(k-1)}) \tag{6.45a}$$

$$s.t. \quad z_{(k+1)} = A_d z_{(k)} + B_d u_{(k)},\ k=0,\ldots,(N-1) \tag{6.45b}$$

$$d_{ji(k),ell}^2 \geq \frac{k_{dsi}}{S_a},\ \forall j,i,k=1,\ldots,N \tag{6.45c}$$

$$\frac{c_{msi} - \mu_{pji(k)}}{\sigma_{pji(k)}} \geq \Phi^{-1}\left(1-\alpha\right),\ \forall j,i,k=1,\ldots,N \tag{6.45d}$$

$$\underline{u} \leq u_{(k)} \leq \overline{u},\ k=0,\ldots,(N-1) \tag{6.45e}$$

$$\underline{z} \leq z_{(k)} \leq \overline{z},\ k=1,\ldots,N \tag{6.45f}$$

$$U_k = \{u_{(0)},\ldots,u_{(N-1)}\} \tag{6.45g}$$

$$Z_k = \{z_{(1)},\ldots,z_{(N)}\} \tag{6.45h}$$

Equations (6.45c) and (6.45d) are deterministic and transformed from chance constraints (6.6). Then, the approximation of $J_p(z_{(N)})$ detailed in Section 4.4 and the first order Taylor series expansion can be used to further transform the optimization problem into a quadratic formulation for real-time implementation.

6.4 Summary

This chapter has considered the situation where the position information of the agent contains measurement noise. The measurement noise is represented by a zero-mean Gaussian random vector in the velocity of the agent ellipsoids. To handle the noisy agent information in the robot safety system, two possible approaches were proposed. The first one is to handle the measurement noise in the agent position estimation block, and the best estimation is input to the decision making block. The advantage of this approach is that no change is required for the online trajectory generation algorithm in the decision making block. However, the estimation required significant computational effort in addition to the online trajectory generation. The second approach, on the other hand, is to handle the measurement noise directly in the formulation of an optimization problem in the decision making block. In this approach, the inequality constraints accounting for safety become chance inequality constraints. By utilizing the mathematical similarity between the Gaussian random vector and the agent ellipsoids, the chance constraints were transformed back to deterministic inequality constraints.

Chapter 7

Conclusion and Future Research

7.1 Summary and Conclusion

In this dissertation, a robot safety system with an online trajectory generation algorithm was proposed for robot manipulators in dynamic environments to achieve the goal position and avoid collisions simultaneously.

Two scenarios in robot collaborations were considered in dynamic environments. One is human–robot collaboration and the other is decentralized robot–robot collaboration. In this dissertation, the human worker in human–robot collaboration and the master robot in decentralized robot–robot collaboration are generalized as agents. A set of ellipsoids which envelope the agent is used to represent the agent in the proposed robot safety system. The agent ellipsoids then are used in the online trajectory generation algorithm.

In order to quantitatively measure the safety of the agent ellipsoids, two factors are taken into account. The first factor is the distances between the robot manipulator and the agent ellipsoids, and the second factor is the momentum of the robot links toward the agent ellipsoids. The distance safety index (DSI) and momentum safety index (MSI) were designed based on the two factors. Two candidates of the overall safety index were proposed to combine the distance safety index and the momentum safety index. The advantage of the proposed safety index is the introduction of the ellipsoid coordinates which are constructed on the local coordinates of robot links. The radius parameters of each ellipsoid coordinate are properly selected so that the unit ellipsoid of each ellipsoid coordinate encloses the corresponding robot link. The distance from an agent ellipsoid to a robot link then is represented by the distance in the corresponding ellipsoid coordinate from the surface of the agent ellipsoid to the origin of the ellipsoid coordinate. With the use of ellipsoid coordinates and ellipsoid representation for agents, the computational load does not increase exponentially when the entire robot manipulator is regarded in evaluation of the safety index of the agent ellipsoids.

The online trajectory generation algorithm was developed in the optimization-based framework. A collision-free trajectory within a planning time horizon is generated by solving the optimization problem. In the formulation of the optimization problem, the objective function was designed to penalize the squared distance from the robot end-effector to the goal position and the squared joint jerk. The discretized kinematics model of robot joints and the safety index were used as the equality and inequality constraints. The original formulation of the optimization problem is nonlinear and non-convex because of the use of forward kinematics and differential forward kinematics. In order to achieve online trajectory generation, the formulated optimization problem is approximated by a quadratic problem with linear constraints. A heuristic method was proposed to select the inequality constraints which are likely to become active or almost active for the next iteration so as to further attenuate the computational load.

The robot safety system and the associated online trajectory generation algorithm were validated in simulations. Three formulations of the optimization problems were validated, including the original nonlinear and non-convex formulation, the quadratic problem approximation, and the quadratic problem approximation with the selection strategy of the inequality constraints. From the simulation results, we have found that the three formulations are capable of generating a collision-free trajectory which lead the robot manipulator toward the goal position. The generated trajectories from the three formulations are smooth. The formulation of the quadratic problem can be solved considerably faster than the nonlinear and non-convex formulation. The selection strategy for the inequality constraints further enhances the computational performance of the quadratic problem. As a practical issue, the measurement of the agents from the sensors could contain measurement noise. To handle the noisy measurement of agent information, two approaches were proposed. The first one is to consider the measurement noise in the agent position estimation block, and then this block outputs the best estimate of the agent position to the decision making block. The second approach, on the other hand, is to handle the measurement noise directly in the formulation of the optimization problem in the decision making block. In this approach, the inequality constraints accounting for safety become chance inequality constraints. By utilizing the mathematical similarity between the Gaussian random vector and the agent ellipsoids, the chance constraints were transformed back to deterministic inequality constraints.

7.2 Future Research

Research may be continued to further improve the topics presented in this dissertation. Because of the complexity of the distance safety index and momentum safety index, the gradient vectors in the Taylor series expansion in this dissertation are computed numerically. In order to reduce the computational load in the evaluation of the gradient vector numerically, a heuristic selection strategy for the inequality constraints was proposed. Computational load can be greatly reduced if the analytical form for the gradient vector in the Taylor series expansion is known. Another possible research topic is a better selection strategy for the inequality constraints of the quadratic problem, independent from whether the gradient vectors are computed numerically or analytically. Furthermore, experimental validation should be done to verify the proposed safety system and the online trajectory generation algorithm.

Bibliography

[1] Alin Albu-schäffer, Christian Ott, and Gerd Hirzinger. “A Unified Passivity-based Control Framework for Position, Torque and Impedance Control of Flexible Joint Robots”. In: *The International Journal of Robotics Research* 26.1 (2007), pp. 23–39.

[2] ANSI. *ANSI/RIA R15.06-2012 Industrial Robots Robot Systems - Safety Requirements.* 2012. URL: `http://webstore.ansi.org/RecordDetail.aspx?sku=ANSI%2fRIA+R15.06-2012`.

[3] Lucian Balan and Gary M. Bone. “Real-time 3D Collision Avoidance Method for Safe Human and Robot Coexistence”. In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems.* Oct. 2006, pp. 276 –282.

[4] Oliver Brock and Oussama Khatib. “High-speed navigation using the global dynamic window approach”. In: vol. 1. IEEE, 1999, pp. 341–346.

[5] Arthur E. Bryson. *Applied Optimal Control: Optimization, Estimation and Control.* CRC Press, 1975. 500 pp.

[6] Kuo-Hung Chiang, Shih-Huan Tseng, Yen-Hsun Wu, Guan-Hao Li, Chi-Pang Lam, and Li-Chen Fu. “Multisensor-based outdoor tour guide robot NTU-I”. In: *SICE Annual Conference, 2008.* Aug. 2008, pp. 1425–1430.

[7] Junho Choi, Sunchul Park, Woosub Lee, and Sung-Chul Kang. “Design of a robot joint with variable stiffness”. In: *IEEE International Conference on Robotics and Automation, 2008. ICRA 2008.* May 2008, pp. 1760–1765.

[8] Hoam Chung and Soo Jeon. “Collision-free trajectory generation of robotic manipulators using receding horizon strategy”. In: *American Control Conference (ACC), 2011.* July 29, 2011, pp. 1692–1697.

[9] Peter Corke. “A robotics toolbox for MATLAB”. In: *IEEE Robotics Automation Magazine* 3.1 (Mar. 1996), pp. 24–32.

[10] Alessandro De Luca, Alin Albu-Schäffer, Sami Haddadin, and Gerd Hirzinger. “Collision Detection and Safe Reaction with the DLR-III Lightweight Manipulator Arm”. In: 2006, pp. 1623–1630.

[11] Alessandro De Luca and Raffaella Mattone. “Sensorless Robot Collision Detection and Hybrid Force/Motion Control”. In: *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on.* 2005, pp. 999–1004.

[12] Jacques Denavit and Richard S. Hartenberg. “A kinematic notation for lower-pair mechanisms based on matrices.” In: *Trans. of the ASME. Journal of Applied Mechanics* 22 (1955), pp. 215–221.

[13] Hao Ding, Gunther Reißig, and Olaf Stursberg. “Increasing efficiency of optimization-based path planning for robotic manipulators”. In: *2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*. Dec. 2011, pp. 1399 –1404.

[14] Hao Ding, Mingxiang Zhou, and Olaf Stursberg. “Optimal motion planning for robotic manipulators with dynamic obstacles using mixed-integer linear programming”. In: *17th Mediterranean Conference on Control and Automation, 2009. MED ’09*. June 2009, pp. 934–939.

[15] Markus Fischer and Dominik Henrich. “Surveillance of robots using multiple colour or depth cameras with distributed processing”. In: *Third ACM/IEEE International Conference on Distributed Smart Cameras, 2009. ICDSC 2009*. Aug. 2009, pp. 1–8.

[16] Fabrizio Flacco, Torsten Kröger, Alessandro De Luca, and Oussama Khatib. “Depth space approach to human-robot collision avoidance”. In: *2012 IEEE International Conference on Robotics and Automation (ICRA)*. May 2012, pp. 338–345.

[17] Yahui Gan, Xianzhong Dai, and Jun Li. “Cooperative Path Planning and Constraints Analysis for Master-Slave Constraints Analysis for Master-Slave Industrial Robots Industrial Robots”. In: *International Journal of Advanced Robotic Systems* (2012), pp. 1–13.

[18] Point Grey. *FireWire CCD Stereo Vision Cameras by Point Grey*. 2014. URL: `http://www.ptgrey.com/products/stereo.asp`.

[19] Herwig Guggi and Bernhard Rinner. “Distributed smart cameras for hard real-time obstacle detection in control applications”. In: *2011 Fifth ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*. Aug. 2011, pp. 1–6.

[20] Gurobi Optimization, Inc. *Gurobi Optimizer Reference Manual*. 2013. URL: `http://www.gurobi.com/`.

[21] Sami Haddadin, Alin Albu-Schäffer, and Gerd Hirzinger. “Requirements for Safe Robots: Measurements, Analysis and New Insights”. In: *The International Journal of Robotics Research* 28.11 (2009), pp. 1507–1527.

[22] Sami Haddadin et al. “Real-time reactive motion generation based on variable attractor dynamics and shaped velocities”. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2010, pp. 3109–3116.

[23] Jochen Heinzmann and Alexander Zelinsky. “Quantitative Safety Guarantees for Physical Human-Robot Interaction”. In: *The International Journal of Robotics Research* 22.7 (July 1, 2003), pp. 479–504.

[24] Dominik Henrich and Thorsten Gecks. “Multi-camera collision detection between known and unknown objects”. In: *Second ACM/IEEE International Conference on Distributed Smart Cameras, 2008. ICDSC 2008.* Sept. 2008, pp. 1–10.

[25] Koji Ikuta, Hideki Ishii, and Makoto Nokata. “Safety Evaluation Method of Design and Control for Human-Care Robots”. In: *The International Journal of Robotics Research* 22.5 (May 1, 2003), pp. 281–297.

[26] Koji Ikuta, Makoto Nokata, and Hideki Ishii. “Safety evaluation method of human-care robot control”. In: *Proceedings of 2000 International Symposium on Micromechatronics and Human Science, 2000. MHS 2000.* 2000, pp. 119–127.

[27] ISO. *ISO 10218-1:2011 - Robots and robotic devices – Safety requirements for industrial robots – Part 1: Robots.* 2011. URL: `http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=51330`.

[28] Jae Wook Jeon. “An efficient acceleration for fast motion of industrial robots”. In: *, Proceedings of the 1995 IEEE IECON 21st International Conference on Industrial Electronics, Control, and Instrumentation, 1995.* Vol. 2. Nov. 1995, pp. 1336–1341.

[29] Seong-Hee Jeong and Takayuki Takahashi. “Optimal braking for impact force reduction using the dynamics of redundant manipulators”. In: *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on.* 2006, pp. 1898–1903.

[30] Seong-Hee Jeong and Takayuki Takahashi. “The development of a service manipulator system for simultaneous realization of high safety and dexterity”. In: *ICMIT 2005: Control Systems and Robotics.* Vol. 6042. SPIE, Dec. 22, 2005, pp. 34–36.

[31] kawada. *NEXTAGE - a next-generation robot able to work together with people.* 2013. URL: `http://nextage.kawada.jp/en/`.

[32] Oussama Khatib. “Real-time obstacle avoidance for manipulators and mobile robots”. In: *1985 IEEE International Conference on Robotics and Automation. Proceedings.* Vol. 2. Mar. 1985, pp. 500–505.

[33] Jörg Krüger, Terje K. Lien, and Alexander Verl. “Cooperation of human and machines in assembly lines”. In: *CIRP Annals - Manufacturing Technology* 58.2 (2009), pp. 628–646.

[34] Jörg Krüger, Bertram Nickolay, P. Heyer, and Günther Seliger. “Image based 3D Surveillance for flexible Man-Robot-Cooperation”. In: *CIRP Annals - Manufacturing Technology* 54.1 (2005), pp. 19–22.

[35] Dana Kulić and Elizabeth A. Croft. “Real-time safety for human - robot interaction”. In: *12th International Conference on Advanced Robotics, 2005. ICAR ’05. Proceedings.* July 2005, pp. 719–724.

[36] Dana Kulić and Elizabeth A. Croft. “Safe planning for human-robot interaction”. In: *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA ’04.* Vol. 2. 2004, pp. 1882–1887.

[37] Dana Kulić and Elizabeth A. Croft. “Pre-collision safety strategies for human-robot interaction”. In: *Autonomous Robots* 22.2 (2007), pp. 149–164.

[38] Dana Kulić and Elizabeth A. Croft. “Real-time safety for human–robot interaction”. In: *Robotics and Autonomous Systems* 54.1 (2006), pp. 1–12.

[39] Dana Kulić and Elizabeth A. Croft. “Safe planning for human-robot interaction”. In: *Journal of Robotic Systems* 22.7 (2005), 383–396.

[40] Konstantinos J. Kyriakopoulos and George N. Saridis. “Minimum jerk path generation”. In: *, 1988 IEEE International Conference on Robotics and Automation, 1988. Proceedings*. Apr. 1988, pp. 364–369.

[41] Bakir Lacevic and Paolo Rocco. “Kinetostatic danger field - a novel safety assessment for human-robot interaction”. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2010, pp. 2169 –2174.

[42] Bakir. Lacevic and Paolo Rocco. “Towards a complete safe path planning for robotic manipulators”. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2010, pp. 5366–5371.

[43] Bakir Lacevic, Paolo Rocco, and Andrea Maria Zanchettin. “Safety Assessment and Control of Robotic Manipulators Using Danger Field”. In: *IEEE Transactions on Robotics* 29.5 (2013), pp. 1257–1270.

[44] Neil Lawrence. *MATLAB Motion Capture Toolbox*. 2014. URL: `http://staffwww.dcs.shef.ac.uk/people/N.Lawrence/mocap/`.

[45] Shuai Li, Sanfeng Chen, Bo Liu, Yangming Li, and Yongsheng Liang. “Decentralized kinematic control of a class of collaborative redundant manipulators via recurrent neural networks”. In: *Neurocomputing* 91 (Aug. 15, 2012), pp. 1–10.

[46] Guanfeng Liu, Jijie Xu, Xin Wang, and Zexiang Li. “On quality functions for grasp synthesis, fixture planning, and coordinated manipulation”. In: *IEEE Transactions on Automation Science and Engineering* 1.2 (Oct. 2004), pp. 146–162.

[47] B. Martinez-Salvador, A.P. del Pobil, and M. Perez-Francisco. “A hierarchy of detail for fast collision detection”. In: *2000 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings*. Vol. 1. 2000, pp. 745–750.

[48] Microsoft. *Kinect for Windows Sensor Components and Specifications*. Dec. 2010. URL: `http://msdn.microsoft.com/en-us/library/jj131033.aspx`.

[49] Gustavo Montemayor and John T. Wen. “Decentralized Collaborative Load Transport by Multiple Robots”. In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005. ICRA 2005*. Apr. 2005, pp. 372–377.

[50] Carlos Morato, Krishnanand N. Kaipa, Boxuan Zhao, and Satyandra K. Gupta. “Toward Safe Human Robot Collaboration by Using Multiple Kinects Based Real-Time Human Tracking”. In: *Journal of Computing and Information Science in Engineering* 14.1 (2014), pp. 011006–1–011006–9.

[51] Masashiro Morioka and Shinsuke Sakakibara. “A new cell production assembly system with human–robot cooperation”. In: *CIRP Annals - Manufacturing Technology* 59.1 (2010), pp. 9–12.

[52] Makoto Nokata, Koji Ikuta, and Hideki Ishii. “Safety-optimizing method of human-care robot design and control”. In: *IEEE International Conference on Robotics and Automation, 2002. Proceedings. ICRA ’02.* Vol. 2. 2002, pp. 1991–1996.

[53] Masafumi Okada, Yoshihiko Nakamura, and Shigeki Ban. “Design of programmable passive compliance shoulder mechanism”. In: *IEEE International Conference on Robotics and Automation, 2001. Proceedings 2001 ICRA.* Vol. 1. 2001, pp. 348–353.

[54] Jung-Jun Park, Hwi-Su Kim, and Jae-Bok Song. “Safe robot arm with safe joint mechanism using nonlinear spring system for collision safety”. In: *IEEE International Conference on Robotics and Automation, 2009. ICRA ’09.* 2009, pp. 3371–3376.

[55] Aurelio Piazzi and Antonio Visioli. “Global minimum-jerk trajectory planning of robot manipulators”. In: *IEEE Transactions on Industrial Electronics* 47.1 (Feb. 2000), pp. 140–149.

[56] Gill A. Pratt and Matthew M. Williamson. “Series elastic actuators”. In: *1995 IEEE/RSJ International Conference on Intelligent Robots and Systems 95. ’Human Robot Interaction and Cooperative Robots’, Proceedings.* Vol. 1. Aug. 1995, pp. 399–406.

[57] Panjawee Rakprayoon, Miti Ruchanurucks, and Ada Coundoul. “Kinect-based obstacle detection for manipulator”. In: *2011 IEEE/SICE International Symposium on System Integration (SII).* Dec. 2011, pp. 68–73.

[58] Rethink Robotics. *Baxter and Baxter Research Robot.* 2014. URL: `http://www.rethinkrobotics.com/resources/safety/`.

[59] SafetyEYE. *SafetyEYE.* 2007. URL: `https://www.pilz.com/en-INT/eshop/00014000337042/SafetyEYE-Safe-camera-system`.

[60] Haddadin Sami, Rico Belder, and Alin Albu-Schaeffer. “Dynamic Motion Planning for Robots in Partially Unknown Environments”. In: Aug. 28, 2011, pp. 6842–6850.

[61] R. Schiavi, G. Grioli, S. Sen, and A. Bicchi. “VSA-II: a novel prototype of variable stiffness actuator for safe and performing robots interacting with humans”. In: *IEEE International Conference on Robotics and Automation, 2008. ICRA 2008.* May 2008, pp. 2171–2176.

[62] Dongjun Shin, Irene Sardellitti, Yong-Lae Park, Oussama Khatib, and Mark Cutkosky. “Design and Control of a Bio-inspired Human-Friendly Robot”. In: *Experimental Robotics.* Springer Tracts in Advanced Robotics 54. Springer Berlin Heidelberg, 2009, pp. 43–52.

[63] Leena Singh, Harry Stephanou, and John Wen. “Real-time robot motion control with circulatory fields”. In: *1996 IEEE International Conference on Robotics and Automation, 1996. Proceedings.* Vol. 3. 1996, pp. 2737–2742.

[64] Jefferey Too Chuan Tan, Feng Duan, Ryu Kato, and Tamio Arai. “Safety Strategy for Human–Robot Collaboration: Design and Development in Cellular Manufacturing”. In: *Advanced Robotics* 24.5 (2010), pp. 839–860.

[65] Stefan Thiemermann. *Direkte Mensch-Roboter-Kooperation in der Kleinteilemontage mit einem SCARA-Roboter*. IPA-IAO Forschung und Praxis. Heimsheim: Jost-Jetter Verlag, 2005. 123 pp.

[66] Stefan Thiemermann. “team@work - Mensch-Roboter-Kooperation in der Montage”. In: *Automatisierungstechnische Praxis atp* 45 (Nr.11 2003), pp. 31–35.

[67] Iwan Ulrich and Johann Borenstein. “VFH+: reliable obstacle avoidance for fast mobile robots”. In: vol. 2. IEEE, 1998, pp. 1572–1577.

[68] Carnegie Mellon University. *CMU Graphics Lab - motion capture library*. 2014. URL: `http://mocap.cs.cmu.edu/`.

[69] Volkswagen. *First robot collaborates directly with employees in Volkswagen plant*. 2013. URL: `http://www.universal-robots.com/GB/Universal_Robots/News.aspx?Action=1&NewsId=459&PID=14345`.

[70] L. Wu, K. Cui, and S. B. Chen. “Redundancy coordination of multiple robotic devices for welding through genetic algorithm”. In: *Robotica* 18.6 (2000), pp. 669–676.

[71] Shu-Wen Yu, Chi-Shen Tsai, and Masayoshi Tomizuka. *The Safety of Robot Coming in Contact with Human*. Dec. 2010.

[72] Andrea M. Zanchettin, Bakir Lacevic, and Paolo Rocco. “A novel passivity-based control law for safe human-robot coexistence”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2012, pp. 2276–2281.

[73] Michael Zinn, Oussama Khatib, Bernard Roth, and J. Kenneth Salisbury. “Towards a Human-Centered Intrinsically-Safe Robotic Manipulator”. In: *In IARP Workshop on Technical Challenges for Dependable Robots in Human Environments*. 2002.

[74] Michael Zinn, Bernard Roth, Oussama Khatib, and J. Kenneth Salisbury. “A New Actuation Approach for Human Friendly Robot Design”. In: *The International Journal of Robotics Research* 23.4 (Apr. 1, 2004), pp. 379–398.

Appendix A

Numerical Computation of Gradient Vector

Algorithm 1 Numerical Computation of Gradient Vector

```
 1: procedure COMPUTE_GRADIENT_VECTOR($f, \mathbf{x_0}, \delta$)
 2:     $\mathbf{g} \leftarrow$ Initialize_Zero_Vector($m$)
 3:     for $i = 1$ to $m$ do
 4:         $\mathbf{x_p} \leftarrow$ Initialize_Zero_Vector($m$)
 5:         $\mathbf{x_m} \leftarrow$ Initialize_Zero_Vector($m$)
 6:         $\mathbf{x_p} \leftarrow \mathbf{x_0} + \delta \mathbf{e}_i$
 7:         $f_p \leftarrow f(\mathbf{x_p})$
 8:         $\mathbf{x_m} \leftarrow \mathbf{x_0} - \delta \mathbf{e}_i$
 9:         $f_m \leftarrow f(\mathbf{x_m})$
10:         $\mathbf{g} \leftarrow \mathbf{g} + \big((f_p - f_m)/(2\delta)\big)\mathbf{e}_i$
11:     end for
12:     return $g$
13: end procedure
```

Algorithm 1 shows the algorithm to numerically compute the gradient vector, g, of a function, $f(\bullet)$, at a point, $x_0 \in \mathbb{R}^m$. The function $f : \mathbb{R}^m \rightarrow \mathbb{R}$ takes a vector in an m-dimensional space as the argument and returns a scalar function value of x. The third argument of Algorithm 1, δ, is a small number used in the computation of gradient vector. In the algorithm, $\mathbf{e}_i \in \mathbb{R}^m$, is a vector that has 1 at the i-th element and zeros at the other elements. The function, Initialize_Zero_Vector(m), is for generating a zero vector with length m.

The concept of the algorithm is to evaluate two points finitely close to x_0 in each dimension, and δ is used to determine how close they are. The slope of the line connecting two points in each dimension is the element of the gradient vector.