On-line Collision and Deadlock Avoidance of PTP Command-Based Industrial Manipulators using Advanced Collision Map

Ahmad Yasser Afaghani

Graduate School of Systems and Information Engineering
Department of Intelligent Interaction Technologies
University of Tsukuba

This dissertation is submitted for the degree of
Doctor of Philosophy in Engineering

March 2015

I would like to dedicate this dissertation to my beloved wife *Dima* and my lovely parents *Hayat and Dr. Jamal* . . .

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 36,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 70 figures.

Ahmad Yasser Afaghani
March 2015

Acknowledgements

This research project would not have been possible without the support of many people. I would like to convey my thanks and gratitude to them throughout this dissertation.

I would like to express my sincere appreciation and gratitude to my supervisor *Prof. Dr. Yasumichi AIYAMA* who was abundantly helpful and offered invaluable assistance, support and guidance. I would like to thank you for encouraging my research and for allowing me to grow as a research scientist.

Deepest thanks are also due to my co-supervisors *Prof. Dr. Takashi TSUBOUCHI*, *Prof. Dr. Yasushi NAKAUCHI*, and *Prof. Dr. Yasuhisa HASEGAWA*. I am appreciate for your helpfulness and kindness during the research.

Special thanks to the committee members *Prof. Dr. Hiromi MOCHIYAMA* and *Prof. Dr. Jun OTA*. Thank you for giving me brilliant comments and suggestions.

I would like also to give special thanks to my previous supervisor during master course *Prof. Dr. Shin' ichi YUTA*.

I would also like to convey special thanks to *Japanese Government* especially *Ministry of Education,Culture,Sports,Science & Technology(MEXT)* who accepted me to do my research in Japan and provided me with the financial means and language learning facilities.

I am thankful to all my colleagues in Manipulation System laboratory. You were nice to me and introduced me more about Japan and its culture.

I would like to say that I am glad and thankful to have friends from different countries who i introduced to them during my study. Thank you for them for spending nice moments together in Japan.

I am very thankful to my dear brother *Salah* and sister *Lama* as well as my mother-in-law *Zainab*, brother-in-law *Abdulkader*, and sisters-in-law *Ruba & Raghad*. Words cannot express how they supported me emotionally during my study. Your prayer for me was what sustained me thus far.

Finally, I wish to express my love and gratitude to my beloved wife *Dima* and my compassionate parents *Hayat and Jamal* who encouraged me emotionally and spiritually to pursue the PhD degree. They were always my support in the moments when there was no one to answer my queries.

Abstract

Since many industrial applications include multiple robot manipulators to achieve several tasks in same workspace, the collision avoidance becomes a significant matter to be considered. Off-line motion planning is the common method which is widely used in factory automation. However, when the control commands of each robot are unpredictable due to unstructured environment such as bin picking, it is necessary to consider on-line motion planning. Moreover, point-to-point (PTP) command-based control, which is widely used for industrial robots, is implemented using a black box controller. That means no modification is possible in the controller system. Thus, the collision avoidance for such robots in on-line mode becomes a significant issue.

This research aims to build a system that can avoid the collisions and deadlocks of n-robot industrial manipulators that are controlled using PTP commands in on-line mode. The robots share same workspace and they have no prior knowledge of the commands which will be sent after start up the system.

To this end, three essential issues are necessary to be addressed. *Collision Detection*, *Collision Avoidance*, and *Deadlock Avoidance*.

First, an *Advanced Collision Map* method is developed to detect the collisions between whole bodies of the robot manipulators and represent them as collision areas on the map. This map provides simple 2D scheme which relates travelling length to servo time, and thus, the collisions which may occur in 3D-space are converted into 2D-space. That gives simplicity of dealing with the problem of collision avoidance. Collision detection using advanced collision map requires to calculate the robots' motions. But since the industrial robot manipulators have complicated shapes, the mathematical expression of robot body is hard and burdensome. Therefore, to decrease the computational cost, the links of the robots are approximated geometrically using the swept sphere volume which presents tight modelling. Furthermore, it is extremely easy to check for collisions and, therefore, feasible for on-line applications.

Second, a *Time Scheduling Method* is applied in this work to avoid the collisions which are detected using advanced collision map. The method guarantees collision-free path of the robot by modifying the trajectory of that robot without changing the geometric path that is

provided by PTP command. It is successful to apply this method due to restrictions of the proposed system that are controlled using black box controllers.

Third, an *Escaping Technique* is developed to avoid the deadlocks which may occur if any robot becomes an obstacle in front of the other. This technique is based on proposing an escaping goal in six directions for the robot manipulator which causes the deadlock. The technique utilizes advanced collision map method intelligently to detect the collisions for the process of avoiding the deadlocks, in addition to decrease meaningless travelling which may be got by proposing far escaping goal.

The collision and deadlock avoidance system has been built for two robot manipulators at first. Then, the system has been generalized to include n-robot manipulators in the workspace. Finally, to demonstrate the effectiveness of proposed methods, the system has been tested and evaluated using an openGL-based simulator.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

ACM	Advanced Collision Map
CA	Collision Avoidance
CAD	Computer Aided Design
CM	Collision Map
DA	Deadlcok Avoidance
EEF	End-Effector
FA	Factory Automation
MR	Moving Robot
PTP	Point-to-Point
SR	Standby Robot
SR_{cmd}	Standby Robot with command
SSV	Swept Sphere Volume
TL	Travelling Length

Chapter 1

Introduction

1.1 Background

Factory Automation (FA) are widely increasing through the use of industrial manipulators. Most applications rely on many robots operating in the same workspace, rather than having one robot in an ad hoc space. This increases productivity and reduces that amount of space needed in the factory. However, including multiple robot manipulators to achieve the tasks will present the problem of collision between them. Thus, the collision avoidance becomes a significant matter to be considered.

Here, it is worth mentioning that motion planning of robots can be categorized into two types:

- *Off-line motion planning*, in which all commands and motion are decided prior to running the system, This type of planning is common method which is widely used in FA. But it requires highly-structured environments, which are exhaustive and expensive to construct. One example for this environment is automotive assembling application as shown in **Fig. 1.1**.

- *On-line motion planning*, in which unpredictable commands are sent to the system during operation. In contrast to *off-line* planning, this kind of planning is better suited to industrial applications that operate in unstructured environments e.g. bin picking as shown in **Fig. 1.2**.

This dissertation highlights on the second kind of motion planning which are still under development.

Meanwhile, point-to-point (PTP) command-based control is widely used for industrial robot manipulators. These commands are sent from the application module which is responsible for distributing the commands to each robot based on the tasks.

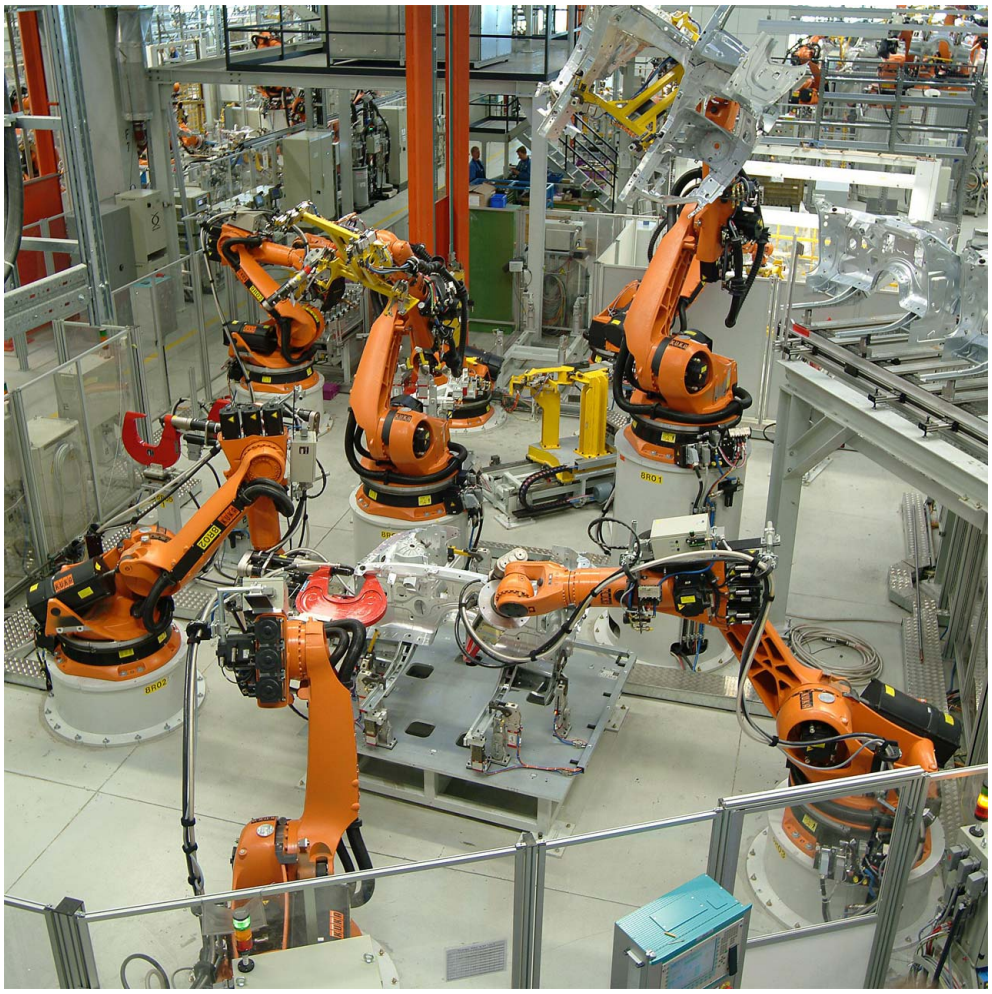


Fig. 1.1 Highly-structured environment: Assembling part of the car in BMW automotive [65]

Fig. 1.2 Unstructured environment: Bin picking application using Fanuc manipulators [66]

However, the controller of industrial robots is a black box i.e. no modification is possible in the controller system. Thus, the collision avoidance for such robots in on-line mode becomes a significant issue. There are a few number of research papers have tackled that issue. Zhou *et al.* [1] proposed a zone-blocking method. When one robot enters into the common workspace, a flag is activated to prevent other robots from entering the workspace. The method is simple and safe but not very efficient.

This work proposes an efficient method for on-line collision avoidance between n-robot industrial manipulators considering the whole body. The robots are controlled using unpredictable PTP commands which are sent after starting the system. Therefore, an advanced collision map method has been developed in this dissertation for detecting collisions between all links of the n-robot body. Furthermore, in some cases and due to unpredictable control commands, the robots may reach to a deadlock situation i.e. the robots become obstacles to each other. Consequently, the new map has been used intelligently to avoid the deadlocks.

1.2 Research Motivation and Purpose

The diversity of industrial applications increases the demand on industrial manipulators. That promoting different companies to compete for building end-user robot manipulators. The controllers of those robots are unique for each company and they have been designed to be a black box, i.e. no modification is possible in the controller system. Some examples of those controller are shown in **Fig. 1.3**.

On the other hand, some applications require to acquire the commands on the fly with no previous knowledge. That is because the robots are operating in an unstructured environment e.g. bin picking. However, the existence of multiple manipulators in the same workspace will increase the productivity but introduce the problem of collision between them. Those problems are the key motivation to start the research.

Therefore, the research aims to build a system with on-line planner for detecting and avoiding any collisions between n-robot industrial manipulators with independent controllers. In addition, the research develops a method to avoid the deadlock situation which is common issue in on-line planning. I believe that the contributions of this work will enhance the advancement of on-line motion planning in industrial applications where the robot manipulators are operated in an unstructured environment with independent controllers.

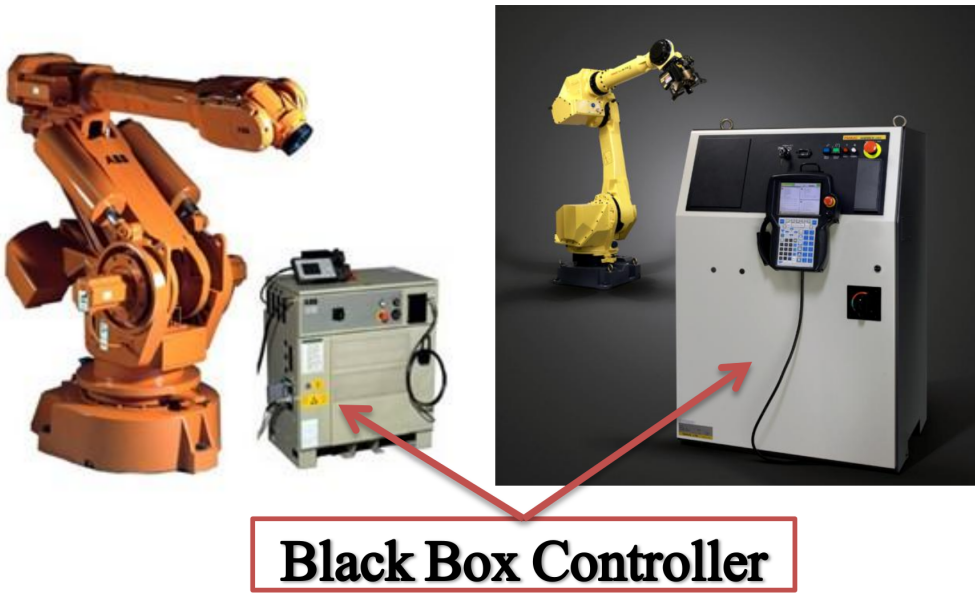


Fig. 1.3 Controllers of different industrial robots. Right picture from [67]. Left Picture from [68]

1.3 Related Works

Collision and deadlock avoidance issue has been studied for decades. Many researchers have addressed that issue in different field of robotics such as mobile robotic field [2][3][4][5] [6] [7], humanoid robotic field [8][9], as well as industrial robotic field.

Regarding industrial robotic field, which is core of this research, many concepts have been suggested for solving the problem of collisions and deadlocks, either using off-line or on-line motion planning. In the following a brief introduction to those works: Lee *et al.* [10] and Chang *et al.* [11] suggested a collision map scheme to detect potential collisions between two robots. To avoid the collisions, a time scheduling method of travelling speed along a planned trajectory has been proposed. Roach *et al.* [12] reported the coordination of robot actions in sparse environment not in tight space. The collision avoidance is performed by delaying or altering the path with restriction of EEF's path to ellipsoid. They interpreted the task-level command for avoiding the collisions. Bien *et al.* [13] proposed a method which plans the time-optimal trajectory of each robot independently under the constraints on actuator torques and velocities, and so, the collision region defined in the coordination space is transformed into time-versus-travelled space with the preplanned velocity of one robot. Thereafter, Lee [14] improved the work of Bien to be more general cases of more than one collision region in coordination space. Shiller *et al.* [15] presented a method for planning the motion of the robot manipulator consid-

ering the presence of obstacles. His method relies on the dynamics, actuator constraint, joint limits, and the obstacles. A small set of near-optimal paths are efficiently selected from a grid. Onda *et al.* [16][17] proposed a general approach to planning collision-free path for a multi-DOF manipulator with a large payload. The method is to find a path practically by searching the most promising portion of the space. Zurawski *et al.* [18] proposed a method for avoiding the collision between two robots by assuming that the workspace is divided into discrete 2D regions, and thus these regions are engaged as the robots move along their paths. If the regions of both robots are intersected, the collision is existed. The collision are avoided by altering the path segments for slave robot. Seshadri *et al.* [19] [20] presented optimum path planning algorithms for robot manipulators. They addressed two different problem of path planning, the minimum-time path planning and the minimum-energy path planning. The problems have been solved based on method of local variations. However, collision check is performed for only the wrist and third link of robot. Mohri *et al.* [21] proposed a method of planning collision-free trajectories of two robots with specified joint paths using virtual coordination space[13]. The minimum time trajectories for two manipulators are searched under joint torques/forces constraints and collision-free condition by dynamic programming.
Lee *et al.* [22] describes the safety arc concept for avoiding collision between two degrees of freedom in planner manipulators. The safety arc is an arc-shaped set composed of possible configuration points avoiding the given constraints in configuration space. Shwarzer *et al.* [23] described a new dynamic collision checker to test path segments in C-space or collections of such segments. The checker is designed to be general, but particularly suited for probabilistic roadmap planners applied to manipulator arms. However, the method is dealing with a continuous path control for collision-free checking. Asakawa *et al.* [24] [25] studied the automation of welding for large structure considering the collision between tool and workpiece. The collision-free path is generated based on basis of CAD workpiece data and the concept of potential field.

On the other hand, many methods and concepts have been proposed regarding on-line collision and deadlock avoidance issue. Khatib [26] presented a unique method for obstacle avoidance by manipulators and mobile robots using artificial potential field concept. Then, Kalaycioglu *et al.* [27] suggested a modified impedance control concept for solving the collision problem. Collision-free robot trajectories are generated incorporating the robot dynamics, the robot constraints, and avoidance of singular configurations. Thus, the problem of local singular configurations as resulted in Khatib's approach has been solved by this technique. Freund *et al.* [28] presented an approach for solving find-path problem in

multi-robot systems based on suitable structuring of hierarchical overall system. The system does a change of the basic control dynamics, the introduction of useful couplings as well as the change of the input gains of the robots. Chien *et al.* [29] proposed an on-line trajectory planning of multiple robot manipulators in a loosely coordinated mode, which considers the tasks of the robot manipulators are independent. A hierarchical control by means of PD-controllers are described. The system is able to avoid the collisions but not the deadlocks. Donnell *et al.* [30] described a method for coordinating the trajectories of two manipulators for avoiding collisions and deadlocks between them. They assumed that the path of the robots are planned off-line, and thus, the trajectories can modified on-line for avoidance. Sun *et al.* [31] worked on non-heuristic motion planning of robot arms in unknown 3D environments. The robot is 3-link type with sensor-based system for detecting the approaching of the obstacles. They suggested to use free configuration space to perform the motion planning. Czarnecki *et al.* [32] proposed a collision-free motion planning for two robots operating in a common workspace. Initial time-optimal trajectories are constructed without attention to potential collision. Collision free trajectories are obtained either by time scheduling or alternatively, utilising a simple search technique to derive a new collision-free path. Hoyer *et al.* [33] presented an online collision avoidance strategy for two robots with six degree of freedom using reduced actual robot and virtual hinderance robot concepts, which is proposed by Borgolte *et al.* [34], to compute the danger of collision. Cheng [35] presented a new approach for on-line collision-free and deadlock-free path planning of two-arm robots for service and assembly tasks, integrated in an on-line task-level planner. He proposed a heuristic algorithm for collision-free path of a robot based on 2D geometric model of swept region of the robot. Li *et al.* [36] described an on-line planner for two cooperating arms whose task is to grab parts on a conveyor belt and transfer them to their respective goals, while avoiding collision with obstacles. The approach is based on breaking overall planning problem into subproblems, each involving a low-dimensional configuration or configuration $\times$ time space. Cellier *et al.* [37] proposed the reflex action theory. For example, when the second arm happens to collide with the first arm, the shape of the protection zones is modified. Sugie *et al.* [38] presented new control method for achieving autonomous obstacle avoidance for manipulator with rate constraints. The work is based on Fujimoto's method [39], and it is exploring the freedom of coordinate transformation in feedback linearization and cope with the rate constraints simultaneously by adopting the state-dependent time scale transformation. Hwang *et al.* [40] [41] proposed a collision-free trajectory planning method based on a speed alternation strategy where on-going paths of the robots are fixed. In the first paper, he applied

the method on two-planar robot then it is generalized to multi-joint manipulators. Furthermore, collision-trend index has been used to find optimal resolution in the proposed method. Freund *et al.* [42] presented a new method CARE (Collision Avoidance in Real-time Environment), as an extension of [43], to avoid collisions by changing the predetermined path for endangered robots using the redundant kinematics of a robot. Juang [44] presented collision avoidance method of multiple-manipulator by integrating generic algorithm and repulsive force concept. Where, a repulsive force is artificially created using the distances between the robot links and obstacles, which are generated by a distance computation algorithm. Chuang *et al.* [45] proposed a potential-based path planning algorithm of articulated robots with 2-DOF joints. The proposed approach computes repulsive force and torque between charged objects by using generalized potential model. A collision-free path can be obtained by locally adjusting the robot configuration to search for minimum potential configurations using these force and torque. Leonard *et al.* [46] presented collision-free and occlusion-free algorithm for industrial robot manipulator with specified pixellated regions of an eye-to-hand camera. The algorithm uses probabilistic roadmap [47] with dynamic occlusion algorithm to determine which pixels of the camera are occluded by the robot during motion. Cascio *et al.* [48] presented a novel approach for robot manipulators trajectory planning in the presence of static and dynamic obstacles based on Karush-Kuhn-Tucker conditions to compute the proximity between line-swept spheres volumes used to model colliding objects. Bosscher *et al.* [49] proposed an algorithm for collision avoidance with two robot manipulators. The method is based on creating a set of linear inequality constraints on the commanded velocity of the robot by formulating joint limits and potential collisions imposed by nearby objects or joint constraints. Chung *et al.* [50] proposed collision-free trajectory generation using receding horizon strategy under dynamic environment. They employed the elliptical set to represent the no-collision zone around each link, and explicitly formulate collision avoidance constraints using state-space inequalities.

In tele-operated robot manipulators which normally required to operate within an unstructured environment under the directions of human operator, e.g. in medical and military applications, on-line collision avoidance has been considered and studied extensively. Beaumont *et al.* [51] proposed collision avoidance solution which depends on the individual links of the manipulators being modelled as a number of spheres. Shaffer *et al.* [52] presented a safety system with real-time collision avoidance based on data structure and update algorithm. The structure is N-objects octree. The goal

was not to perform robot path planning, but rather to support real-time safety system to warn of imminent collisions between two robot arms. Lemelsky *et al.* [53] proposed to use whole-sensitive arm manipulators whose whole bodies are covered with a sensitive skin sensors to detect the nearby objects. And thus, collision avoidance of robot manipulators' tasks can be handled automatically. Bon *et al.* [54] described an on-line approach to whole-arm collision avoidance for Ranger Tele-robotic. That was based on virtual repulsive forces of obstacle to avoid collisions. Spencer *et al.* [55] described the cooperative control of both tele-operated and autonomous redundant manipulators with overlapping workspace.

All aforementioned methods and more which have proposed different methods of collision and deadlock avoidance problem can be categorized into general methods e.g C-space and workspace methods e.g potential field. In addition, most of previous methods have addressed the problem for two robot manipulators only.

However, the restrictions of industrial robot manipulators in FA which is black box characteristic of the controllers as well as PTP command-based control have rarely tackled in previous literature papers. Some methods have been proposed e.g [1] and [56] which addressed the problem for only two robot manipulators.

Therefore, In this dissertation, we are going to consider those restrictions to build an efficient on-line collision and deadlock avoidance system of n-robot industrial manipulators with independent controllers.

1.4 Outline of Dissertation

The structure of the dissertation is going to be as follows:

- In Chapter 2, the entire on-line collision and deadlock avoidance system of n-robot manipulators is presented. The system structure, conditions and assumptions, and strategies are explained in detail.
- In Chapter 3, an advanced collision map method which is described. It is one of the most important contributions which have been developed is this work . The method of advanced collision map is explained at first for two robot manipulators, and then, it is generalized in the case of n-robot manipulators.
- In Chapter 4, the entire algorithm of on-line collision avoidance system including two robot manipulators are described. Here, the algorithm is designed regardless the

deadlock situations. This algorithm is consists of two main parts, collision detection and time scheduling which are going to be explained. Afterwards, the algorithm is tested using openGL-based simulator to demonstrate the effectiveness.

- In Chapter 5, the deadlock avoidance method is proposed to improve the system which is described in Chapter 4. The chapter starts with explaining the background of how do the deadlock situation happens. Afterwards, the methodology of deadlock avoidance is explained, followed by several simulations which prove the successfulness of the proposed system.

- In Chapter 6, the proposed system which is explained in Chapter 4 and 5 are generalized. The algorithm of n-robot on-line collision and deadlock avoidance system is described in detail. This is done by explaining the necessary modifications which are done to previous system: the modification to collision avoidance algorithm, and the modification to deadlock avoidance system. Finally, the simulations including three and four robot manipulators are presented.

Chapter 2

On-line System Description

Before starting to explain the methods and main algorithm of collision avoidance system, it is important to understand the structure of proposed system. This chapter is described the overall system for on-line collision and deadlock avoidance. Section 2.1 describes what does on-line system mean in this work. In Section 2.2, the conditions and assumptions of the system are explained. Those information are the key for building the main framework of the algorithm. Then, Section 2.3 describes the structure of overall system. Afterwards, Section 2.4 presents some necessary strategies which are relied on for implementing the system in on-line mode. Finally, Section 2.5 is presenting some conclusions.

2.1 Introduction

Industrial applications which use robot manipulators are often controlled with point-to-point (PTP) commands. Some applications are designed to acquire the commands on the fly with no previous knowledge. That is because the robots are operating in an unstructured environment e.g. bin picking. However, having more than one robot operating in the same workspace will increase the productivity but introduce the problem of collision between these robots. Therefore, there should be an on-line planner.

The main tasks of the on-line planner is to detect and avoid collisions and deadlocks between the robots. The input of the planner are user or application commands to each robot. Those commands are processed inside the planner before output them to robot controllers. The on-line term in this scope doesn't mean that the planner can deal with unforeseen events as in real time systems but it can process the acquired commands, which are not predictable before start up the system, on the fly to avoid any collisions and deadlocks.

Here, if the commands from user are all acquired at the same cycle time to process them on-line, the algorithm is going to fall into very long and undesirable waste time of calculation.

To this end, the planner is set to acquire the command one by one from the user and process it at this moment. Thus, the collision and deadlock avoidance is performed at each acquirement of a command for one of the robot manipulators.

2.2 System Conditions and Assumptions

To built on-line collision and deadlock avoidance system, some conditions for that system should be considered. These conditions are defined as follows:

1. The robots are controlled using PTP commands. These commands have information to move the robots in a straight line in the workspace.

2. The controllers that are responsible to control the robots are all black box. That means the controller design is not modifiable.

3. The controllers of the robots are independently operated. Thus, there is possibility to have more than one kind of controller to control various robots in the workspace.

In addition to above mentioned conditions, the control restrictions of industrial robot systems have considered as conditions to design the collision avoidance system as well.

1. When the robot executes a PTP command, the motion of the robot becomes restricted and unchangeable until reaching the target.

2. Paths of the PTP commands which are sent from application module are not modifiable. Thus, they are regarded as high priority in collision avoidance process.

This system has been built with some assumptions which are necessary to be considered in the design as well:

1. The black box characteristic of the controller refers to impossibility to modify the design of the controller according to the system condition. But that doesn't mean the impossibility of acquiring some information from the controller. In this work, it is assumed that the controllers of the robots should have following specifications to suit the system algorithm:

   a The ability to provide some information of the robot to be used in collision and deadlock avoidance process. These information are mainly robot posture which include the position of each joint, in addition to acceleration and maximum velocity of robot's EEF.

b The controller should provide the user with the inverse kinematics which is used to move the robots' EEFs point-to-point. Thus, it can be used to do collision detection.

c The correspondence with robot controllers usually have some delay time, in addition, the time needed to execute the command may take some delay. In this work, we assume that those time delays are neglected because the experiments are held using simulator only. However, in case of testing that system using real robot manipulators, the time delays should be considered as important parameter.

2. The system is operated in an environment which has no obstacles in the workspace. That refers to both moving and static obstacles.

3. Paths of the robots regarding the PTP commands are straight lines. That means, when the robot acquires a PTP command to move from initial to target positions, it will move in a straight line. Note that deciding the path will have effect on designing the collision and deadlock avoidance algorithm.

4. The system is assumed to work with the applications that robots' tasks are independent i.e. time constrained is not existed. Thus, There is no priorities are assigned to the tasks.

The above mentioned conditions and assumptions draw the framework of the on-line system. They help to design the algorithm for avoiding the collisions and deadlocks within that framework.

2.3 System Structure

The ordinary system for controlling the robots industrially consists of two main modules. One module is for sending specific tasks to the robots as commands and the other module is for acquiring that commands, and executing them to accomplish the tasks.

To solve the problem of collisions and deadlocks between the robots, a new module are proposed to be included between above mentioned modules. Thus, the entire system structure is composed of three substantial modules as follows:

1. *Application Module*, which provides PTP commands for n-robot. These commands are accumulated in n-message queues. Each message queue has the commands for one of the robots. Then these commands are acquired by the next module.

2. *Collision & Deadlock Avoidance Module*, which is responsible to acquire the commands which are sent by application module one after the other. This module is the

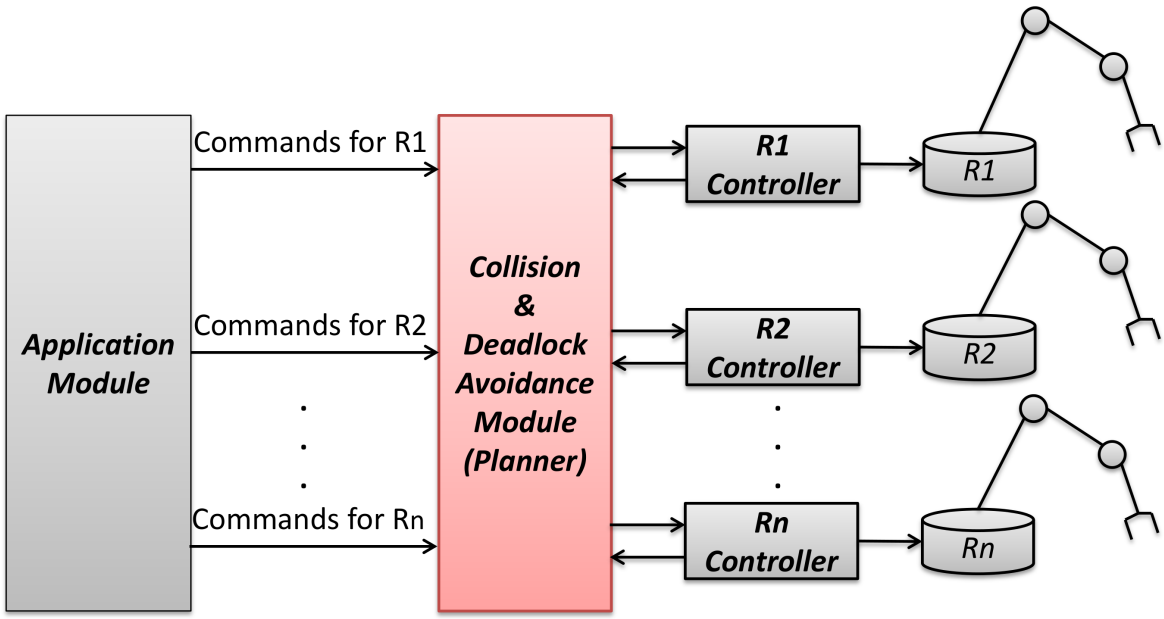


Fig. 2.1 Overview of collision and deadlock avoidance system.

planner which is responsible for detecting collisions between n-robot on the basis of received commands as well as the current data acquired from the robots. Subsequently, the scheduling of the commands' execution timing for avoiding any collisions and deadlocks is performed before sending the commands to the robot.

3. *Robot Controllers Module* are responsible for controlling the robots and providing the planner with the necessary information from the robots such as posture, speed, and acceleration in order to perform collision detection.

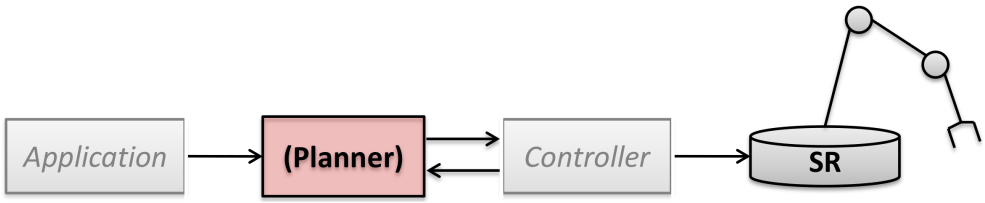
The system modules are illustrated in **Fig. 2.1**.

2.4 System Strategies

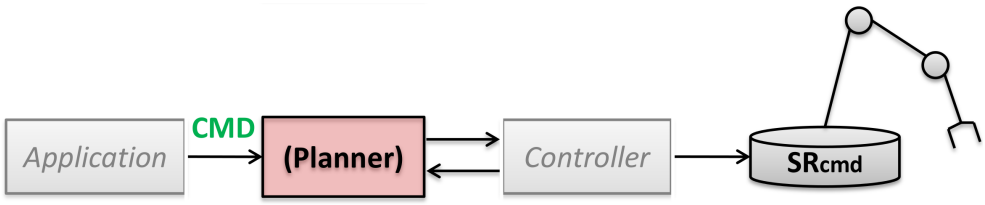
On-line motion planning requires the collision and deadlock avoidance process to be completed as fast as possible. That because long time planning is undesirable in on-line mode. Thus, some strategies are proposed to carry out the planning.

2.4.1 Robot Mode Strategy

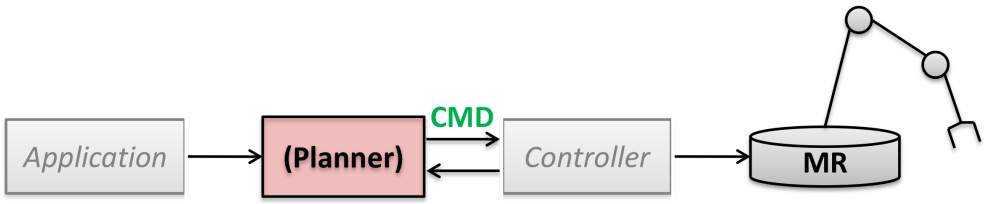
Modes of the robot are defined based on its situation under the system conditions. The main purpose of this strategy is to design designing collision detection algorithm which will be explained in next chapter. Those modes are:



(a) SR mode



(b) SR_{cmd} mode



(c) MR mode

Fig. 2.2 Modes of robot.

1. *Standby Robot* (SR): The robot is denoted as SR if the robot is waiting for new command from application module as shown in **Fig. 2.2(a)**.

2. *Standby-with-Command Robot* (SR_{cmd}): When the planner acquires a command from application module, the robot immediately turns into SR_{cmd} mode as shown in **Fig. 2.2(b)**.

3. *Moving Robot* (MR): After processing the acquired command in the planner, so that the path that provided by the command is free of collision, the command will be sent to the controller for execution. Then, the robot will have the MR mode as shown in **Fig. 2.2(c)**.

2.4.2 Command Acquisition Strategy

The way of acquiring the commands will have effect on the overall algorithm. In this dissertation, the strategy which is proposed to acquire the commands is based on picking up the commands which are sent by application module one by one. In other words, picking up process is performed in sequence for the robots starting from robot number 1 to robot

number *n*. Therefore, the process of collision detection and avoidance will be started as soon as acquiring a new PTP command by the planner for one of the robots.

According to this strategy, the number of SR_{cmd} which can be existed is only one robot among n-robot. Thus, the cycle of the robot modes will have the following flow:

1. *SR*, which is the initial mode of the robot at the start up of the system.
2. SR_{cmd}, as soon as acquiring a command by the planner for one of the robots.
3. *MR*, after processing the command which is received by the planner, the command is sent to the controller for execution. Hence, the robot will turn into MR mode. Then, after accomplishing the command, the robot will turn into SR mode again.

2.4.3 Robot Modelling Strategy

Modelling of the robot arm plays a significant role in detecting possible collisions between robots. The model should be precise while being based on a simple mathematical representation to minimize the calculation cost. For this to be feasible in an on-line system, many researchers have suggested different modelling techniques, as described below.

- In spherical modelling e.g. [58] [63], two parameters are needed, namely, the center and the radius of the sphere. Owing to the rotational invariance of the sphere, the modelling is computationally simple but wasteful of volume.
- Cylindrical modelling involves an elegant shape whereby each link is covered with a tube, but the mathematical expressions are complicated in 3D space.
- Polyhedral modelling [64], which represents each link by a huge number of polygons, provides a very accurate representation but is computationally intensive.
- Discrete-orientation polytopes (DOP) [61], which is one of the bounding box volume (BBV) type, has been utilized for deformed objects but are not useful for rigid bodies such as robot arms.
- Oriented bounding box (OBB) [62] is also suitable for a rigid body and offers very tight modelling and good rotational movement, except that mathematical computation is very intensive.

In this work, the robot arms are modelled using the *Swept Sphere Volume* (SSV)[59], which integrates both tightness and mathematical simplicity. SSV is a volume formed by moving a sphere of a certain radius on a specified primitive such as a point, line, or rectangle

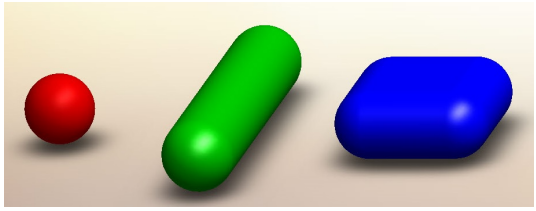


Fig. 2.3 Swept sphere volume using point, line, and rectangular primitives.

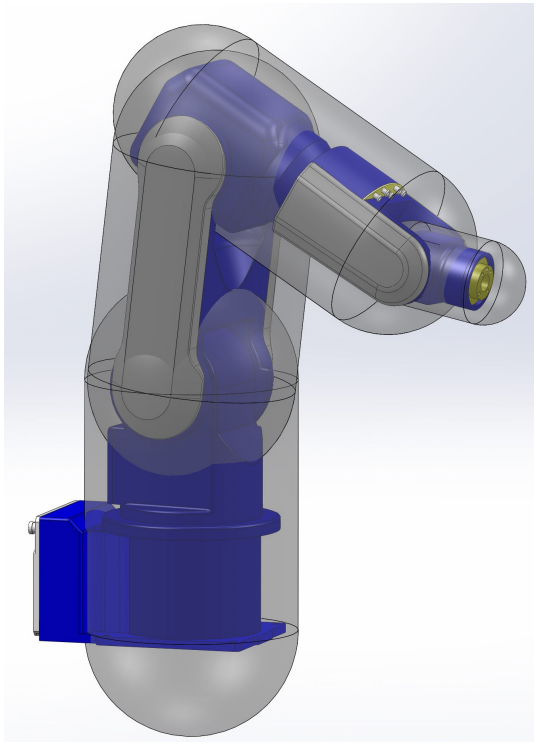


Fig. 2.4 Modelling of robot arm using swept sphere volume.

as shown in **Fig. 2.3**. By changing the primitive dimensions and radius of the sphere, it is possible to model any designated object. The SSV modelling method has two main advantages which are:

- *Tightness*: That because of the possibility of using any number of primitives to model any robot manipulator. Therefore, by selecting appropriate values of sweeping shperes' radii, it is possible to model the object as tight as possible.
- *Mathematical Easiness*: That is because we need to model any object using only two parameters which are primitive dimension and radius of sweeping sphere over the primitive.

The rotational invariant characteristic of the sphere gives simplicity of collision detection using SSV. That characteristic facilitates the finding of the shortest distance between the

primitives and compares it with radii summation of the spheres which sweep on the primitives. Thus, the computational cost for this model is low, which makes it appropriate for on-line applications.

Since most industrial robot arms have a near-tube shape, a line primitive is utilized for modelling the whole body of the robot. One example of robot modelling is applied on Yaskawa Motoman arm. This arm is constructed of a base, two links, and an EEF. Thus, four line segments are enough to model the arm with appropriate radii of swept spheres. The robot's CAD design is shown in **Fig. 2.4** using SSV with four line primitives.

2.5 Conclusions

This chapter has described the overview of n-robot collision and deadlock avoidance system that is working in on-line mode. The system has been designed regarding some conditions and assumptions. To make this system feasible in on-line industrial applications, some strategies have been proposed. Those strategies are robot modes, command acquisition, and geometric modelling of the robot.

After understanding the system structure, the next chapters will describe the methods which are proposed to detect and avoid collisions and deadlocks as well as the entire algorithm of the system.

Chapter 3

Advanced Collision Map

This chapter describes *Advanced Collision Map* (ACM) method which is developed in this work. The ACM is improved map based on the concept of *Collision Map* (CM) which has been developed many years ago. The chapter starts with introduction to CM in Section .3.1. In this section, the steps of designing the map is presented as well. Afterwards, in Section 3.2, the ACM design is explained for the case of two robots and followed by generalized case which is n-robot. Then, Section 3.3 is described the time of building ACM. Finally, Section 3.4 is the conclusions of this chapter.

3.1 Collision Map Concept

3.1.1 Introduction of Collision Map

In 1987, Lee *et al* [10] has developed a method for representing the collisions between two robots which are sharing common workspace. The notion is based on representing the collisions which are happened in 3D coordinate as an area on 2D coordinate space, this area called *Collision Area* which can be obtained from the path and trajectory information of the two robots. The 2D coordinates scheme relates travelling lengths to corresponding servo time of the robot that has not yet executed its command. That 2D scheme is called *Collision Map* and it was designed to reveal the potential collisions only between the end-effectors (EEFs) of two robots.

3.1.2 Design of Collision Map

This section presents the way of designing CM according to our proposed system conditions. The description of this design will help to understand the design of advanced collision map which has been developed in this work.

According to our proposed system conditions, let assume there are two robots $R1$ and $R2$ in the workspace which are MR and SR_{cmd} respectively, where robot $R1$'s EEF is moving in a straight line from $S1$ to $E1$ with path length l_{MR}, and the time needed to reach the target is t_{tar}^{MR}. While $R2$'s EEF is supposed to move according to the new PTP command from $S2$ to $E2$ with path length $l_{SR_{cmd}}$, and the requisite travelling time $t_{tar}^{SR_{cmd}}$. A simplified graph of robot paths is shown in **Fig. 3.1**.

The collision between two robot EEFs may occur if the distance between them $d(t)$ is equal or less than the *prohibited distance*. This distance can be determined based on the way of modelling the EEFs. If the EEFs has modelled using spheres with specific radius $r1$ and $r2$ for the robots $R1$ and $R2$ respectively, the prohibited distance will be the radii summation of modelled EEFs $r1+r2$. Thus, there is collision if $d(t) <= r1+r2$. Consequently, if a sphere with radius $r1+r2$ is centred at a point $P_1(t)$ on the $R1$'s path at time t, the collision is existed if that sphere is intersected or contacted with $R2$'s path. Hence, the equation of $R2$'s path is denoted as:

$$P_2(\lambda) = P_2(t_0) + \lambda \cdot (P_2(t_{tar}^{SR_{cmd}}) - P_2(t_0)) \tag{3.1}$$

Where, t_0 is initial time which is assumed to be $t_0 = 0$, $t_{tar}^{SR_{cmd}}$ is the requisite travelling time of $R2$'s trajectory as mentioned before, $P_2(t_0)$ and $P_2(t_{tar}^{SR_{cmd}})$ are initial and target positions of $R2$'s path respectively, and $1 \leq \lambda \leq 0$ is slope of the path. Here, in order to determine

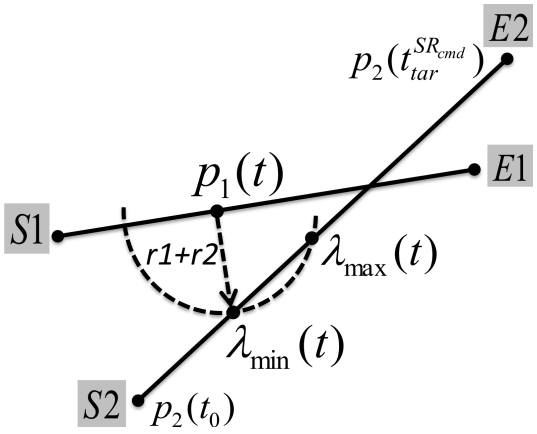


Fig. 3.1 Two Robot Paths($R1$&$R2$)

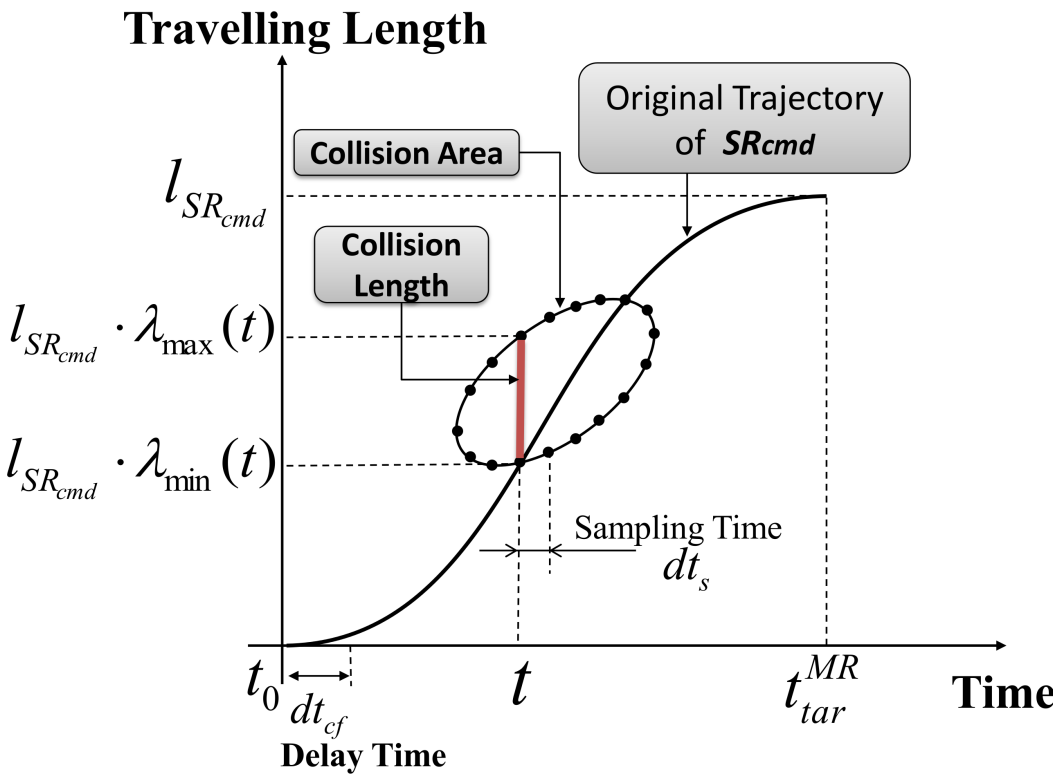


Fig. 3.2 Collision map scheme of two robots

the collisions between two robots along servo time of $R1$ which is t_{tar}^{MR}, it is necessary every fixed sampling time dt_s to calculate the intersecting points between the sphere and $R2$'s path. Assuming that $(P_{2x}(\lambda), P_{2y}(\lambda), P_{2z}(\lambda))$ is the path of $R2$, and $(P_{1x}(t), P_{1y}(t), P_{1z}(t))$ is the position of $R1$ at time t, then, by solving the equation of the sphere which is:

$$(P_{2x}(\lambda) - P_{1x}(t))^2 + (P_{2y}(\lambda) - P_{1y}(t))^2 + (P_{2z}(\lambda) - P_{1z}(t))^2 = (r1 + r2)^2 \tag{3.2}$$

Three possible solutions of λ as function of time can be obtained:

1. *Two Roots* $\lambda_{min}(t)$ and $\lambda_{max}(t)$: means that there is collision.
2. *One Root* $\lambda(t)$: means there is tangential collision.
3. *No Roots*: so there is no collision between $R1$ and $R2$ at time t

Multiplying the roots every dt_s by $R2$'s path length will resulted to have collision lengths. Thus, by drawing these lengths onto collision map, an area called *Collision Area* will be formed as shown in **Fig. 3.2**. As a result, the robots $R1$ and $R2$ are collided if and only if the trajectory of $R2$ passes through that collision area in the map.

3.2 Design of Advanced Collision Map

Since the original CM is not able to detect the collisions between the whole bodies of the robots, it is not feasible in the recent industrial applications. In addition, the original map is able to reveal the collision between only two robots. That is not enough to suit the system requirements which is collision avoidance of n-robot industrial manipulators. To solve these weaknesses in that map, an advanced CM has been developed in this work. The new map has the same scheme of original one with advantage of collision information between whole bodies of n-robot industrial manipulators.

This work has performed the improvement on two steps. At first, the original CM has been improved to be able to detect the collisions between the whole bodies of two robots. Then, the concept of advanced collision map has been generalized for suiting the case of n-robot.

3.2.1 Advanced Collision Map for Two-Robot

Let assume that the workspace is equipped with two robots, namely, $R1$ and $R2$. Assume that a new PTP command has been acquired by the planner to move $R2$'s EEF from $S2$ to $E2$ with path length $l_{SR_{cmd}}$, and requisite travelling time $t_{tar}^{SR_{cmd}}$. At this moment, the robot $R1$ has already adhered to its path so it is MR, where $R1$'s EEF is located at $P1$ and heading toward $E1$. The necessary time to reach the target from $P1$ is t_{tar}^{MR}. A simplified sketch of the workspace with two robots and paths are shown in **Fig. 3.3**.

The danger between the robots can be estimated by measuring the minimum distance between their links along traveling time of MR. If the distance has exceeded a prohibited distance, it will be judged as a collision between the robots. The prohibited distance can be

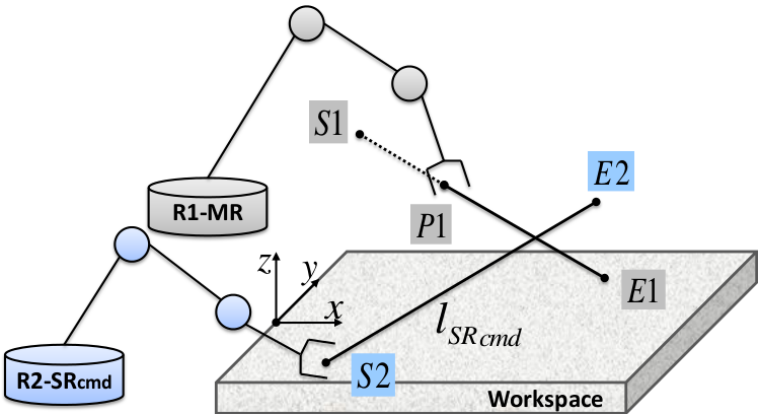


Fig. 3.3 Paths of robots in shared workspace.

determined with the radii summation of pair of links according to SSV modelling of that links. Let us denote the SSV line segments that model $R1$ and $R2$ by seg_j^{MR} and $seg_i^{SR_{cmd}}$, respectively. Where, $i \in [1,m]$ and $j \in [1,n]$ are indicated by segment number, and n and m are the maximum number of segments for $R1$ and $R2$, respectively. To detect potential collisions between the whole bodies of both robots, it is necessary to check each segment pair of the MR and SR_{cmd} under their original trajectory information. To detect collisions between a pair of segments, first, a minimum distance between both segments is calculated as $d_{min}(seg_i^{SR_{cmd}}, seg_j^{MR})$ [Appendix A]. Then, the minimum distance is compared with the summation of the swept spheres radii for both segments. If that distance is equal to or less than the result of the summation, then, there is a collision between two segments.

$$d_{min}\left(seg_i^{SR_{cmd}}, seg_j^{MR}\right) \leq r_i + r_j \tag{3.3}$$

where, r_i and r_j are the radii of the spheres which sweep on $seg_i^{SR_{cmd}}$ and seg_j^{MR}, respectively.

Collision detection of whole bodies of two robots means including the links of that robots. Thus, designing a map for including the collisions information of all links require the path and trajectory information of each link. However, the ordinary map is able to detect the collision only between EEFs of two robots. Consequently, creating the CM of each pair of links of both robots will be resulted to have many CMs with different collision information. Each CM is presented the collision information of pair of links. In order to get one CM with all collision information could be get by combining all resulted CMs. And that is impossible to be performed because the collision area of each resulted CM is created from different trajectories information. Therefore, we define a new collision mapping concept namely *Collision Timing Map* (CTM). CTM relates the servo time of MR with travelling time of SR_{cmd}. It aims to calculate the collision timing between the robots' links. This new map is created between pair of links of both robots MR and SR_{cmd}. As a result, if there are m links of SR_{cmd} and n links of MR, a group of $m \times n$ CTMs are obtained for all pair of links as listed in **Table 3.1**.

To design CTM, it is assumed that the SR_{cmd} starts to move as soon as the planner acquires the command, so, the SR_{cmd} will move from initial position to target position within a time period of $t_{tar}^{SR_{cmd}}$. The collision is tested every sampling time dt_s along servo time of the MR $t \in [0, t_{tar}^{MR}]$. Thus, at time t, the MR's segment seg_j^{MR} has a specific posture which can be acquired by the inverse kinematic. This segment posture is checked with the SR_{cmd}'s segment $seg_i^{SR_{cmd}}$ for all postures every time sample along $\tau \in [0, t_{tar}^{SR_{cmd}}]$. As a result, ranges of collision timings are obtained as $T_{ij}(t,1), T_{ij}(t,2), \ldots, T_{ij}(t, rng_{ij}(t))$. Where, $rng_{ij}(t)$ is the maximum number of ranges at time t for a checked pair (ij). The range at time t for pair

Table 3.1 Collision timing map between each pair of links of MR and SR_{cmd}.

SR_{cmd} \ MR	Base	Link1	$\cdots$	EEF
Base	CTM_{11}	CTM_{12}	$\cdots$	CTM_{1n}
Link	CTM_{21}	CTM_{22}	$\cdots$	CTM_{2n}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
EEF	CTM_{m1}	CTM_{m2}	$\cdots$	CTM_{mn}

(ij) is referred to as a continuous collision and starts from $\tau_{ij}^{str}(t,k)$ and ends at $\tau_{ij}^{end}(t,k)$ whereas $k \in [1, rng_{ij}(t)]$ refers to the range number. After finishing all collision checks for every pair of segments, $m \times n$ collision range results are acquired, as listed in **Table 3.2**.

An illustrative graph of the CTM of pair (ij) is shown in **Fig. 3.4**. Afterwards, for determining the potential collisions between the whole bodies of both robots, a union of the ranges [Appendix B] every time sample within $t \in [0, t_{tar}^{MR}]$ for all pairs is calculated as follows:

$$\bigcup_{i=0}^{i=m} \bigcup_{j=0}^{j=n} \bigcup_{k=0}^{k=rng_{ij}(t)} T_{ij}(t,k) \longrightarrow \begin{bmatrix} T_{Total}(t,1) \\ T_{Total}(t,2) \\ \vdots \\ T_{Total}(t, rng_{Total}(t)) \end{bmatrix}. \tag{3.4}$$

The results is a group of total ranges of collision timings as illustrated in the following:

$$T_{Total}(t,1), T_{Total}(t,2), \ldots, T_{Total}(t, rng_{Total}(t)) \tag{3.5}$$

Where, $rng_{Total}(t)$ is the maximum number of total ranges at time t. In addition, each total range has a boundary which starts from $\tau_{Total}^{str}(t,k_{Total})$ and ends at $\tau_{Total}^{end}(t,k_{Total})$ whereas $k_{Total} \in [1, rng_{Total}(t)]$ refers to the total range number.

Since the original collision map scheme is related to the travelling length of EEF corresponding to the servo time, the collisions between EEFs can only be illustrated. Thus, it is necessary to covert the CTM scheme to have collision map scheme, i.e. converting from travelling time space to travelling length space. To this end, trajectory information of the SR_{cmd} is used. At every time sample along $t \in [0, t_{tar}^{MR}]$, the range $T_{Total}(t,k_{Total})$ is used to get the travelling length of the EEF's trajectory only at start and end time $\tau_{Total}^{str}(t,k_{Total})$ and

Table 3.2 Ranges of collision timings obtained between each pair of segments of MR and SR_{cmd}.

	seg_1^{MR} (Base)	seg_2^{MR} (Link1)	$\cdots$	seg_n^{MR} (EEF)
$seg_1^{SR_{cmd}}$ (Base)	$T_{11}(t,k)$	$T_{12}(t,k)$	$\cdots$	$T_{1n}(t,k)$
$seg_2^{SR_{cmd}}$ (Link1)	$T_{21}(t,k)$	$T_{22}(t,k)$	$\cdots$	$T_{2n}(t,k)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$seg_m^{SR_{cmd}}$ (EEF)	$T_{m1}(t,k)$	$T_{m2}(t,k)$	$\cdots$	$T_{mn}(t,k)$

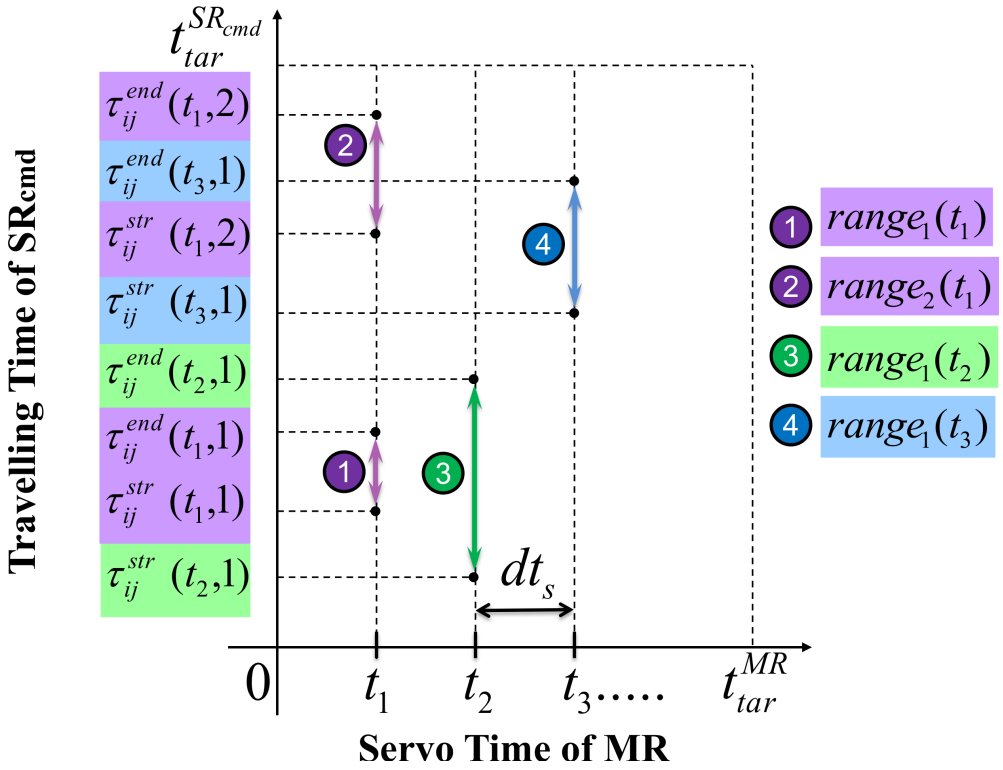


Fig. 3.4 Collision timing map for presenting the ranges of collision timings between segment j of MR and segment i of SR_{cmd}.

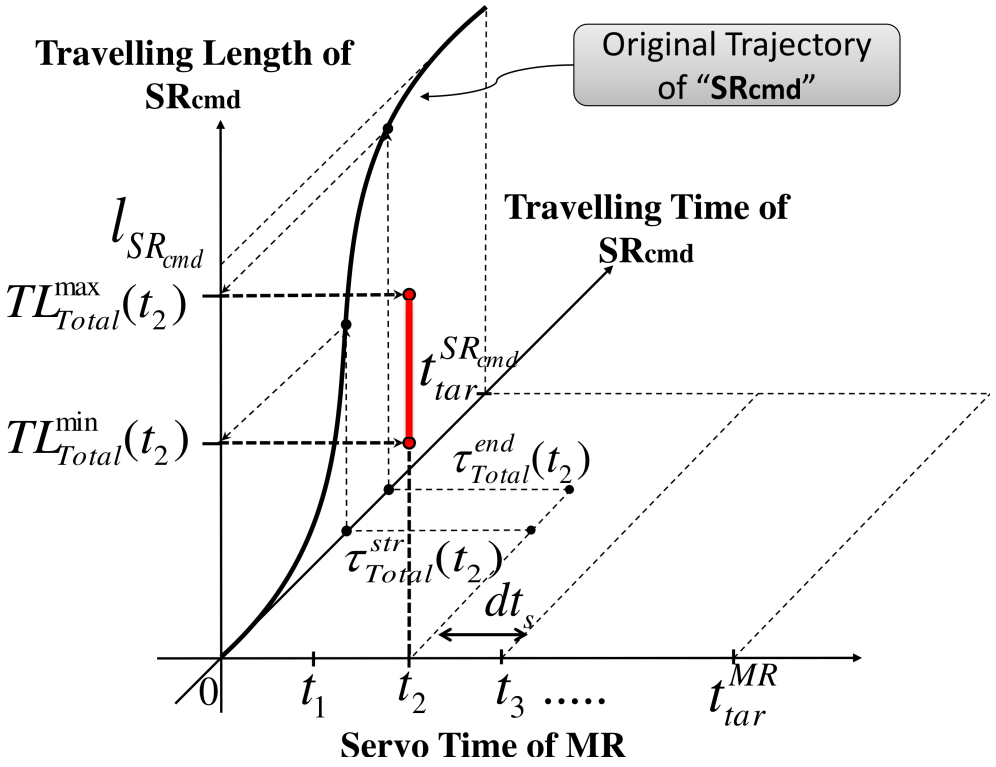


Fig. 3.5 Conversion map for changing from time space to travelling length space using trajectory information of SR_{cmd}'s EEF.

$\tau^{end}_{Total}(t, k_{Total})$, respectively:

$$TL^{min}_{Total}(t, k_{Total}) = f_{TL}(\tau^{str}_{Total}(t, k_{Total})) \tag{3.6}$$

$$TL^{max}_{Total}(t, k_{Total}) = f_{TL}(\tau^{end}_{Total}(t, k_{Total})) \tag{3.7}$$

where, $TL^{min}_{Total}(t, k_{Total})$ and $TL^{max}_{Total}(t, k_{Total})$ are the minimum and maximum travelling lengths of the range at time t, and f_{TL} is the function for calculating the travelling length of the SR_{cmd} at a desired time assuming that $f_{TL}(0) = 0$. A graph which relates the SR_{cmd}'s travelling length to the servo time of the MR by means of travelling time of the SR_{cmd} is illustrated in **Fig. 3.5**. It is shown how to obtain a range of collision lengths at time t, where the SR_{cmd}'s trajectory is shown as a curved line and the range is shown with a straight bold line.

By drawing all the ranges on that graph, an area called the *Collision Area* will be formed as shown in **Fig. 3.6**. This figure is a graph of the final *Advanced Collision Map* which relates the SR_{cmd}'s travelling length to the servo time of the MR. The final arrival time t^{MR}_{tar} of the MR might be smaller, equal to, or larger than the final arrival time $t^{SR_{cmd}}_{tar}$ of the SR_{cmd}.

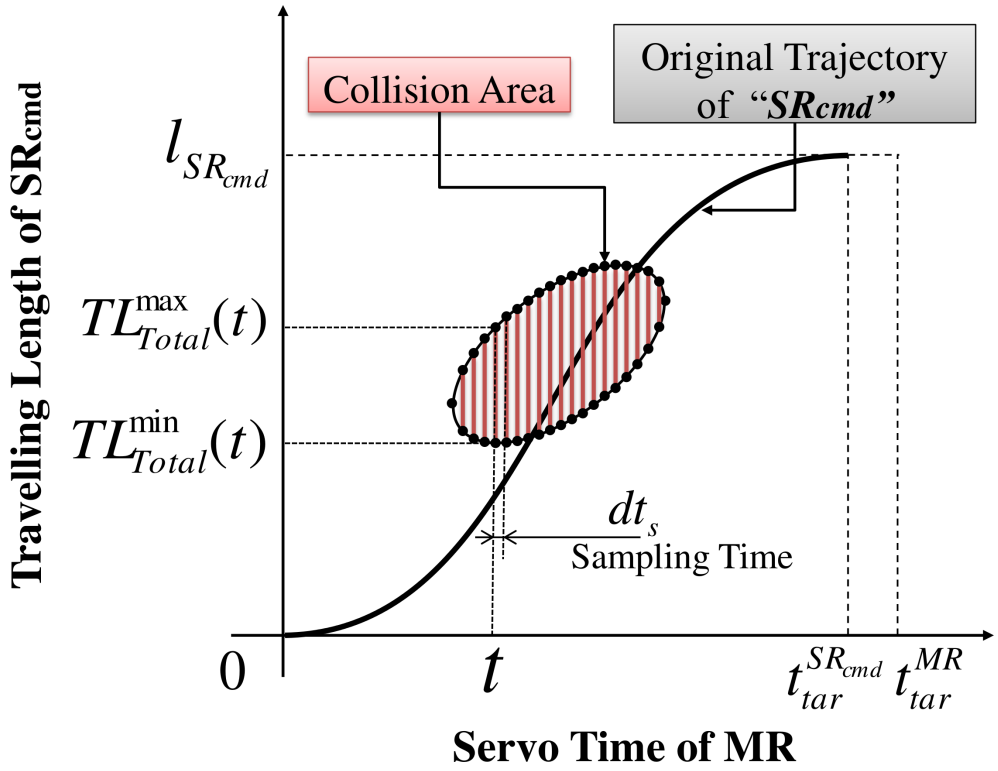


Fig. 3.6 Advanced collision map of two robots showing collision area with original trajectory of SR_{cmd} assuming that $t^{MR}_{tar} > t^{SR_{cmd}}_{tar}$.

Here, the graph depicts the original trajectory of the SR_{cmd} passing through the collision area assuming that $t^{MR}_{tar} > t^{SR_{cmd}}_{tar}$. It is notable that the advanced collision map has devolved to resemble the original design of the map [10], and the collision area in the advanced collision map includes all the information on potential collisions between the whole bodies of the robots while the original map is limited to information on collisions between the EEFs. The avoidance method of that area, is elucidated in the next chapter.

3.2.2 Advanced Collision Map for n-Robot

To generalize the concept of advanced collision map, it is improved to detect the collision of n-robot. The concept is based on treating with all MRs in the workspace as moving segments. Since the SSV modelling, which is utilized in this work, is able to model any robot with suitable number of segments, thus, increasing the number of MRs in the same workspace will increase the number of moving segments. To understand the concept, the steps of creating n-robot ACM are explained as follows:

1. Determining the maximum number of MRs in the workspace at the moment of creating ACM. Let indicates to that number as N.

Table 3.3 Ranges of collision timings obtained between each pair of link of MRs and SR_{cmd}.

SR_{cmd}	MR_1				⋯	MR_N			
	Base	Link1	⋯	EEF	⋯	Base	Link1	⋯	EEF
Base	CTM_{1-11}	CTM_{1-12}	⋯	CTM_{1-1n_1}	⋯	CTM_{N-11}	CTM_{N-12}	⋯	CTM_{N-1n_N}
Link	CTM_{1-21}	CTM_{1-22}	⋯	CTM_{1-2n_1}	⋯	CTM_{N-21}	CTM_{N-22}	⋯	CTM_{N-2n_N}
⋮	⋮	⋮	⋮	⋮	⋯	⋮	⋮	⋮	⋮
EEF	CTM_{1-m1}	CTM_{1-m2}	⋯	CTM_{1-mn_1}	⋯	CTM_{N-m1}	CTM_{N-m2}	⋯	CTM_{N-mn_N}

2. Determining n_i which is the number of segments of the MR_i.

3. Doing collision check between SR_{cmd}'s segments and all segments of every MR, and thus, instead of having $m \times n$ CTMs as listed in **Table 3.1**, there will be N_{CTM} as illustrated in following equation:

$$N_{CTM} = m \times \sum_{i=1}^{N} n_i \tag{3.8}$$

   The CTMs for N number of MRs are listed in **Table 3.3**.

4. The union of aforementioned CTMs is performed. As a result, one single CTM is obtained as the case of two robots ACM.

5. This CTM is converted to have final advanced collision map. This map relates the travelling length of SR_{cmd} with servo time of all MRs.

Here, It is very important to consider that there are many MRs' servo times. Thus, the collisions between SR_{cmd} and all MRs should be checked to the very latest travelling time among MRs. In the case of two robots ACM, the servo time of all CTMs is same, which is servo time of MR, however, in this case, the servo time of all CTMs should be unified to perform the union of those CTMs. Therefore, the servo time of all CTMs is set to be equal which is the maximum travelling time among all MRs:

$$\max_{i=1}^{N}(t_{tar}^{MR_i}) \tag{3.9}$$

As s result, the final ACM scheme after converting final single CTM will be as shown in **Fig. 3.7**.

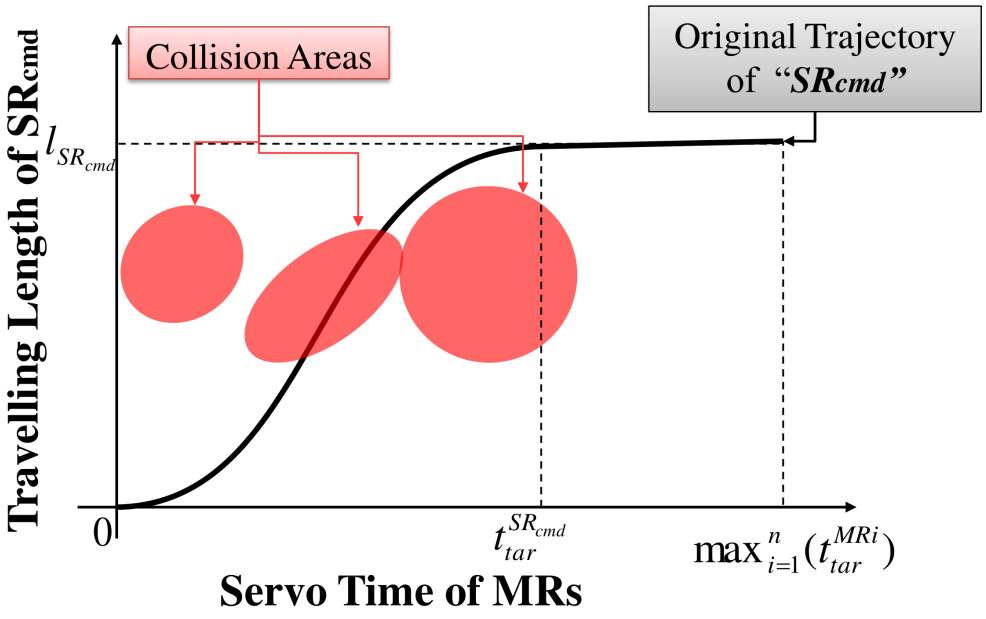


Fig. 3.7 Advanced collision map scheme in the case of n-robot

3.3 Time of Building Advanced Collision Map

Due to working under on-line condition, time of building ACM should be considered well. We refer to this time as dt_m. This time is varying based on several factors as follows:

1. First factor is path length of robots, with which the time of building ACM is varying. Thus, if the paths are long, they will increase the time of calculating dt_m.

2. Second factor is the accuracy of calculating the ACM, meant the sampling time dt_s which is used to build the map. The smaller the sampling time, the longest calculating time dt_m.

3. Third factor is the number of segments which are modelled the robot's whole body. Also, increasing the number of segments will affect to increase the time dt_m because the number of checking pair will increase.

4. Fourth factor is very important that is the PC power. This factor affects on the time of performing one cycle which is $t_{process}$. Relying on recent development of PC power, the time of calculating ACM could be decreased much. Even though it can not be neglected in overall collision avoidance algorithm.

The above factors are arranged in an equation to calculate the approximate time of building ACM. This equation is:

$$dt_m = t_{process} \times (n_{seg}^{SR_{cmd}} \times \sum_{i=1}^{n} n_{seg}^{MR_i} \times max(n_{sample}^{MR_i}) \times n_{sample}^{SR_{cmd}}) \tag{3.10}$$

Where, $n_{seg}^{SR_{cmd}}$ refers to the number of segments which models SR_{cmd}, $n_{seg}^{MR_i}$ is the number of segments which models MR_i, $n_{sample}^{MR_i}$ is the number of sampling time of MR_i robot, and $n_{sample}^{SR_{cmd}}$ is the number of sampling time of SR_{cmd}.

Although the time dt_m is very small but it is going to be included in the design of collision avoidance algorithm which is going to be explained in next chapter.

3.4 Conclusions

This chapter has presented a novel concept of collision detection which is advanced collision map. The design procedure of advanced collision map in described in the case of two robots in the workspace, and then the concept is generalized to include unlimited number of robots in the workspace. This method is feasible in on-line applications which utilized industrial robot manipulators because of computational simplicity and ability to be applied for any shape of robot manipulator. In addition, and due to recent high power PCs, the time which is required to create ACM is short enough to be applied in industry.

Next chapters will utilize the ACM method in collision detection procedure of the system. That is done in the combination case SR_{cmd}-MR.

Chapter 4

Two-Robot Collision Avoidance System

In chapter 2, the on-line system of collision avoidance has been described. The core of that system is the planner which is responsible to do collision avoidance process. This chapter describes the overall algorithm for detecting and avoiding the collisions between two industrial robot manipulators. It is necessary to start building the system for two-robot for establishing the main base of collision avoidance algorithm. That will help for generalizing the current algorithm to suit n-robot case. The system overview of two-robot system is illustrated in **Fig. 4.1**. The system structure itself is same as n-robot system that is composed of three main modules: application module, planner, and controller.

This chapter is divided into five main sections. Section 4.1 is presenting the collision detection algorithm. Section 4.2 describes the concept of avoiding the collisions using time scheduling method. Afterwards, Section 4.3 explains the entire algorithm of the two robots collision avoidance system. Section 4.4 demonstrates the simulation results of two-robot collision avoidance system using openGL-based simulator. Finally, Section 4.5 is discussing the results and evaluating them comparing with previous methods.

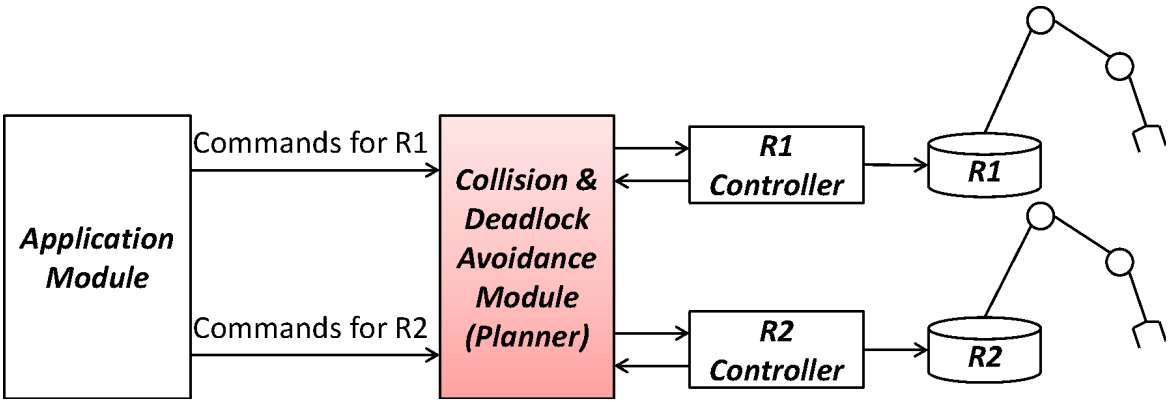


Fig. 4.1 Overview of collision avoidance system with two-robot.

4.1 Collision Detection

The collision check between the robots is key for avoidance process. The detection should be precise to avoid any undesirable or sudden collision between the robots.

According to system strategies which are introduced in Section 2.4, the collision detection and avoidance should start as soon as acquiring a command by the planner for one of the robots. Thus, that robot will turn into SR_{cmd}. Then, this robot will check the other robot mode. Here, three different combinations could be obtained:

1. SR_{cmd}-SR
2. SR_{cmd}-MR
3. SR_{cmd}-SR_{cmd}

But regarding the system strategy of command acquisition method, the third combination can not exist. That because the commands are processed one by one before sending it to the controller for execution. In other words, only one SR_{cmd} will detect and avoid the collisions before turning into MR.

In this dissertation, the collision detection is divided into two methods based on the above mentioned combinations. The methods are as follows:

1. *Advanced Collision Map*, which is already described and explained in detail in previous chapter. This method is applied with case of SR_{cmd}-MR combination.
2. *Simple Collision Detection*, which is applied with case of SR_{cmd}-SR combination.

The second method is based on simple distance check between a pair of links of both robots and then compare the distance with prohibited distance of those links.

To understand the second method, let assume that the workspace has two robots $R1$ and $R2$. Both robots are SR mode initially. If the planner acquires a PTP command for one of the robots; impose it is $R1$, meant that $R1$ is SR_{cmd}, then, the other robot $R2$ becomes a static object in front of $R1$. Hence, SR_{cmd}-SR is obtained. Here, the detection is only performed between the current posture of $R2$ and each posture of $R1$ along entire travelling time of $R1$. Every sampling time dt_s the minimum distance [Appendix A] between each pair of links of both robots is calculated under the original trajectory of $R1$. The method is sufficed to detect at least one collision between any pair of links at any time to judge that there is an *Unavoidable Collision* which cannot be avoided i.e. the newly acquired command can not be executed. It can be concluded that the method is starting as soon as obtaining SR_{cmd}-SR, and the output of the method could be *No Collision* or *Unavoidable Collision*.

4.2 Time Scheduling Concept

4.2.1 Methodology

Collision avoidance methods have been studied over years in literature (See Section 1.3). Those methods can be divided into two categories:

- *Time Scheduling Method*: modifying the trajectory while fixing the geometric path.
- *Path Modification Method*: modifying the geometric path of the robot.

The condition of this work, as mentioned in Section 2.2, is to deal with the PTP commands which are sent by application module as they are. This means no change is allowed to the geometric path which are given by the PTP command. Therefore, time scheduling method has been applied due to suitability of this method regarding proposed system conditions. Time scheduling method is depends on modifying the command's execution time while fixing the geometric path which is given by PTP command.

With the SR_{cmd}-MR combination, to avoid the collision area on ACM, the original trajectory of the SR_{cmd} is shifted by dt_{cf}, which is the necessary delay time required to produce a collision-free path for the SR_{cmd}. Thus, the delay time is calculated as follows:

Reference to the previous assumption where $R1$ is MR which adheres to its original trajectory, and $R2$ is SR_{cmd} which must modify its trajectory to avoid the collision area, assuming an initial value of delay time $dt_{cf} = 0$. the method starts by getting the start and end times of the collision area which are t_{ca}^{str} and t_{ca}^{end}, respectively. To check whether the trajectory of $R2$ passes through the area within that period, the travelling length of $R2$'s trajectory by means of the function $f_{TL}(t)$ is calculated at every time sample dt_s:

$$TL(t - dt_{cf}) = f_{TL}(t - dt_{cf}) : t \in [t_{ca}^{str}, t_{ca}^{end}], t_{ca}^{str} \geq dt_{cf} \tag{4.1}$$

where, f_{TL}, as mentioned in Section 3.2, is the function for calculating the travelling length of the SR_{cmd} at a desired time assuming that $f_{TL}(0) = 0$. Subsequently, $TL(t - dt_{cf})$ is compared with all available ranges of the collision lengths at time t. The range is determined with the minimum and maximum lengths, which are $TL_{Total}^{min}(t, k_{Total})$ and $TL_{Total}^{max}(t, k_{Total})$ as described in Eqs. (3.6) and (3.7). Therefore, a collision exists if $TL(t - dt_{cf})$ is within the range defined as follow:

$$TL_{Total}^{min}(t, k_{Total}) \leq TL(t - dt_{cf}) \leq TL_{Total}^{max}(t, k_{Total}) \tag{4.2}$$

If any collision is detected, the original trajectory of $R2$ will be shifted by one time sample $1 \times dt_s$, and the delay time is increased by $dt_{cf} = dt_{cf} + dt_s$. The next step is a repetition

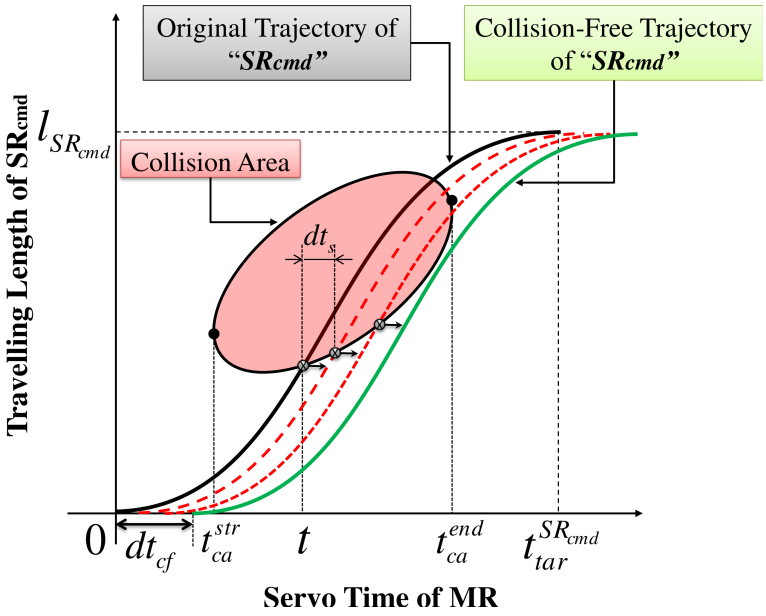


Fig. 4.2 Description of calculating necessary delay time to avoid collision area.

of the last steps but this time the check is done with a new position of travelling length considering that, after each shift, if $t_{ca}^{str} < dt_{cf}$, the time range check is reduced to become $t \in [t_{ca}^{str} + dt_s, t_{ca}^{end}]$. Thus, by continuing to shift the trajectory every time a collision is detected, we get a final collision-free trajectory of $R2$ as illustrated by a curved line in the rightmost of **Fig. 4.2**. The algorithm which presents the time scheduling method for calculating the necessary delay time dt_{cf} is illustrated in **Algorithm 1**.

Algorithm 1 Calculating the delay time dt_{cf} to produce collision-free trajectory

Require: Start and end timing of Collision area t_{ca}^{str} and t_{ca}^{end}

Initializing: $dt_{cf} = 0$;$collision = true$
while $(collision)$ **do**
  for $(t = t_{ca}^{str}$ to $t_{ca}^{end}])$ **do**
    $collision \leftarrow false$; $TL \leftarrow f_{TL}(t - dt_{cf})$
    if ($TL_{Total}^{max}(t, k_{Total}) \geq TL \geq TL_{Total}^{min}(t, k_{Total})$) **then**
      Increase dt_{cf} by one sampling time $1 \times dt_s$
      $collision \leftarrow true$
      break
    end if
  end for
end while

4.2.2 Collision Types

Advanced collision map could have either collision area or not. In the case of existence of collision area, **Algorithm 1** can be applied to have the necessary delay time dt_{cf}. This delay could get three possible cases:

1. $dt_{cf} = 0$: means that there is collision area on the map with no collision between the robots.

2. $dt_{cf} > 0$: means that there is collision, and the time of command execution should be postponed.

3. $dt_{cf} = \infty$: means there will be an unavoidable collision if the command is executed immediately.

The third case may happen in two situations:

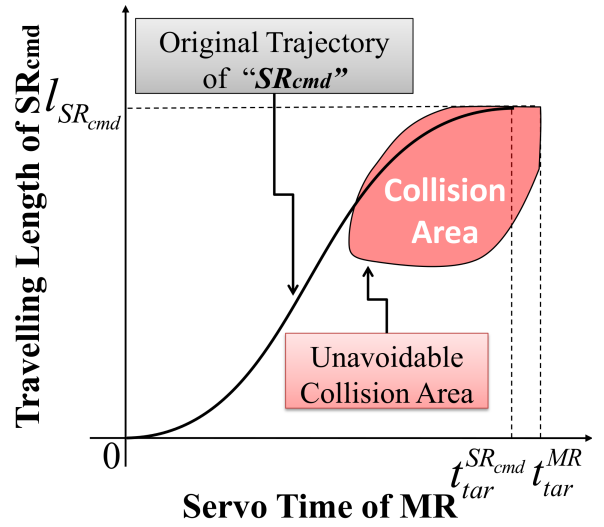
1. If the original trajectory of SR_{cmd} passes through the collision area which has an end time $t_{ca}^{end} = t_{tar}^{MR}$ i.e. the collision still exists even after the end travelling of MR. We refer to this area as *Unavoidable Collision Area*. Some examples for unavoidable collision areas are shown in **Fig. 4.3(a)** and **Fig. 4.3(b)**.

2. If the original trajectory of SR_{cmd} passes through the collision area which is located at the bottom of ACM as shown in **Fig. 4.3(c)**. This means SR_{cmd} will collide with other robot even if it doesn't move. Thus, delaying the execution time of SR_{cmd} is not effective. However, this situation is not possible with SR_{cmd}-MR combination because MR is executed after make sure that its path is free of collision. But this situation will exist in the deadlock avoidance process which will be discussed in detail in next chapter.

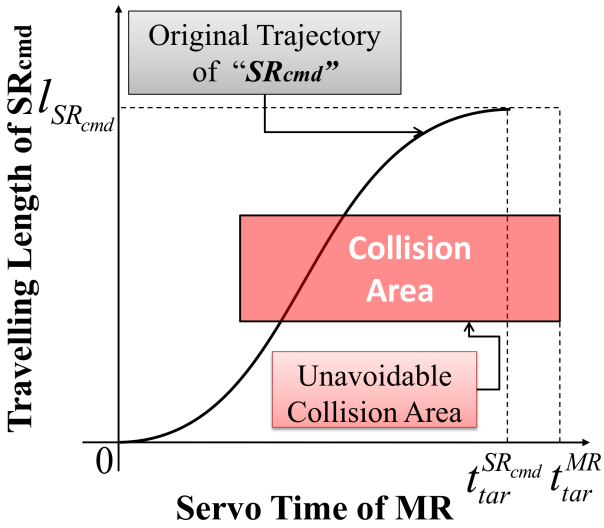
Consequently, in the first situation, SR_{cmd} will be prevented to move, so, we get a *deadlock* situation which will be discussed in the next chapter. The deadlock may happen in the SR_{cmd}-SR combination as well, if the collision is detected between both robots.

Since the collision map is built while the system is operating, it is necessary to consider the time of building the ACM dt_m which has explained in Section 3.3. The time dt_m should be included to calculate the final delay time for obtaining collision-free trajectory of SR_{cmd}. To this end, a software timer has been added. Thus, the final delay time required to avoid the collision area in on-line mode is:

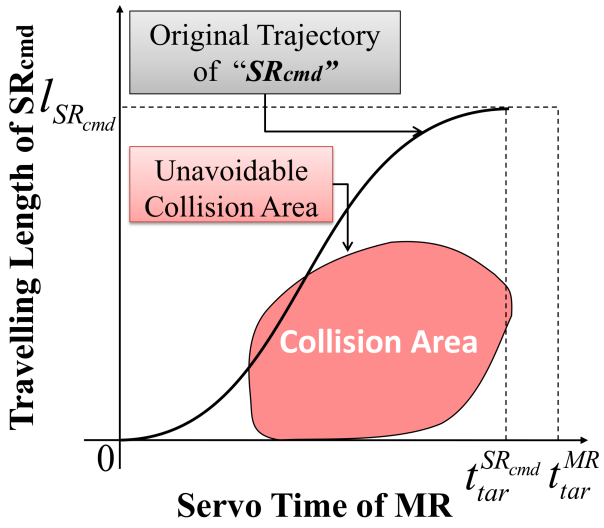
$$dt = dt_{cf} - dt_m \tag{4.3}$$



(a)



(b)



(c)

Fig. 4.3 Illustrative examples of unavoidable collision areas.

4.3 Two-Robot Collision Avoidance Algorithm

The previous sections have described the necessary steps to achieve the main goal. To understand the flow of on-line collision avoidance system for two robots; meant the algorithm of the planner itself, the overall algorithm is described in this section.

As elucidated in Chapter 2, the planner is designed to be centralized structure. Thus, the algorithm of the planner, which is contains two robots in the workspace, is divided into two main parts. Each part is responsible for one of both robots. Both parts have the same structure which consists of four essential stages as follows:

1. *Command Acquisition*, in which the message queue is read for acquiring any new command.

2. *Collision Detection*, in which the collision with other robot is examined.

3. *Command Scheduling*, in which the time scheduling method is applied for avoiding any collisions.

4. *Command Execution*, in which the command is executed by the robot, and then, it verifies the end of command execution time before transferring to the first stage again which is command acquisition.

A brief flow chart of this algorithm is shown in **Fig. 4.4**. Both parts of the algorithm are connected together in complicated structure which is going to be explained in detail in Chapter 6. But briefly say, when each stage task is performed, it will be saved and then the process is transferred to next robot algorithm which is the second part of the algorithm. After finishing a stage task in second robot, it will be saved and then the process is transferred to the first robot stage where it has been left and so on.

The flow of one part of the algorithm is starting with initializing, here, the robot is SR mode. Then, the message queue is read for any command. If a command is acquired, the robot turn into SR_{cmd} and the other robot mode is checked. Therefore, if the combination is:

1. SR_{cmd}-SR: simple collision check is performed. If any collision is detected it will be judged as deadlock. on other side, if there is no collision, the command is executed immediately.

2. SR_{cmd}-MR: advanced collision map is created. Then the collision type is estimated. If it is unavoidable collision, the algorithm will wait other robot to get new command. But if there is collision, the command will be delayed with necessary delay time before execution. on other side, if there is no collision the command will be executed immediately.

After executing the command the robot is turned into MR mode. Thus, the algorithm will wait for accomplished the command before transferring to the beginning.

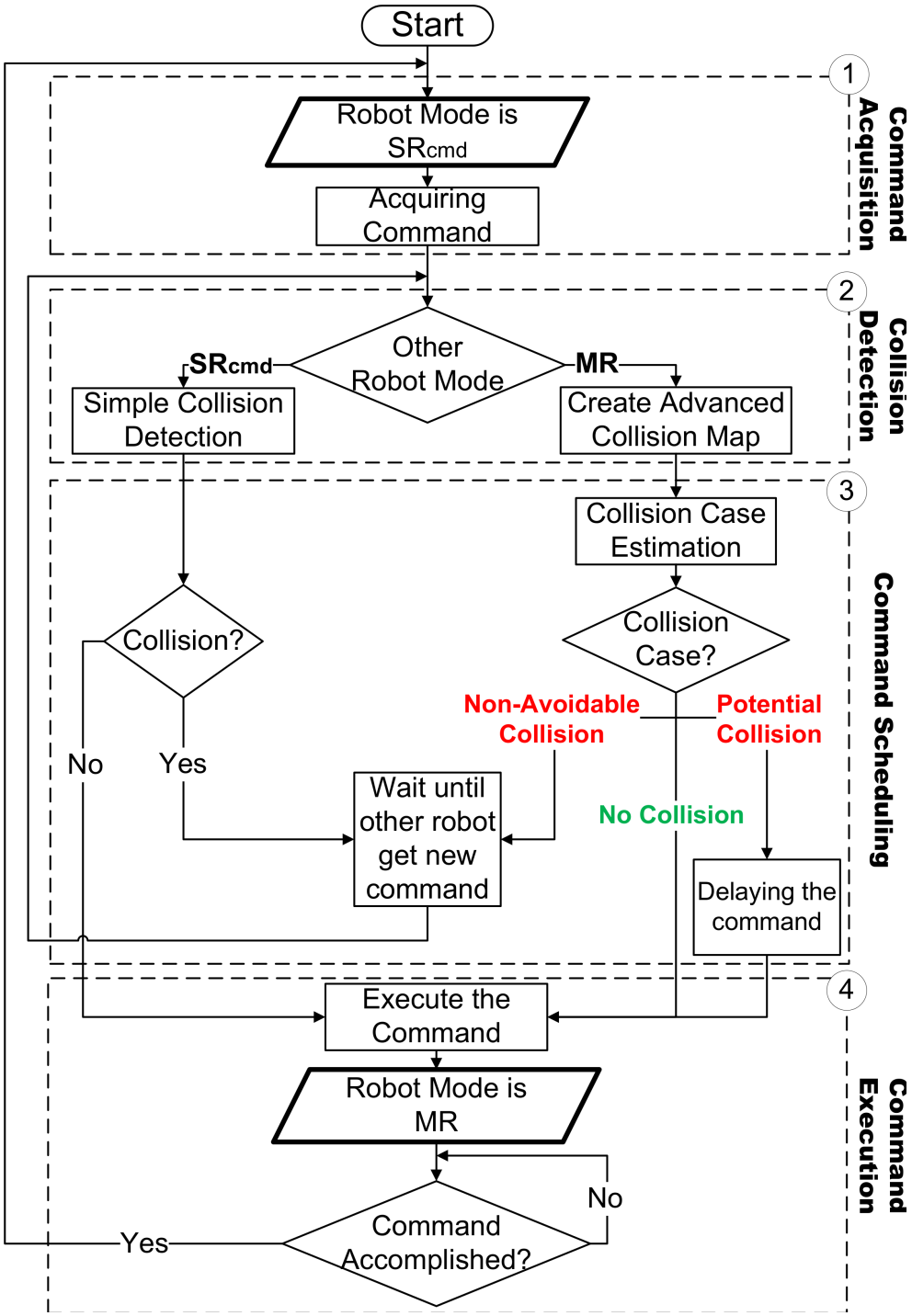


Fig. 4.4 Brief flow chart of collision avoidance algorithm for one robot.

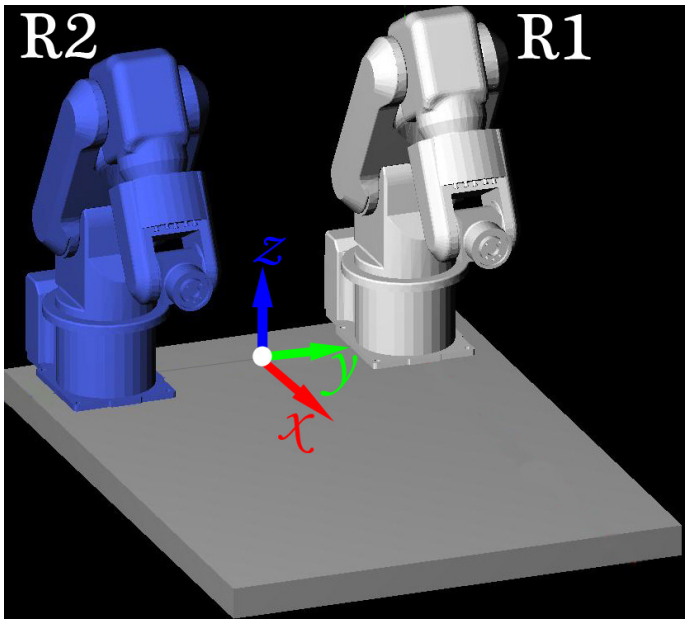


Fig. 4.5 Snapshot of openGL-based simulator that shows two robots in their initial posture.

4.4 Simulation Results

The collision avoidance system has been tested and evaluated with two robots using an OpenGL-based simulator. The simulator can emulate the motion of any robot by providing 3D-CAD model of the robot as well as the joint angles of the robot every time sample of motion. In this work, the simulator is used to imitate two Yaskawa Motoman robots (HP3J and UPJ). The robots are fixed on a board, with a distance between the central axis of the robots' bases being 500 mm. The global coordinate system is located in the middle between both robot bases. A snapshot of the simulator is shown in **Fig. 4.5**.

4.4.1 Simulation Experiment 1

The experiment is divided into two parts. The first part is performed without a collision avoidance module, so the commands are sent directly to the robot controller, after which the motion is monitored. The second part is performed with a collision avoidance module. Let us assume that $R1$ is the gray color robot and $R2$ is the blue robot, and the maximum velocity and acceleration of each robot EEF are set as $V1_{max} = V2_{max} = 100$ mm/s and $a1 = a2 = 100$ mm/s^2, respectively. Both robots have the same length of links which are $l_{base\text{-}joint1} = 290$ mm, $l_{joint1\text{-}joint2} = 260$ mm, $l_{joint2\text{-}joint3} = 270$ mm, and $l_{joint3\text{-}EEF} = 90$ mm. The radii of the spheres which sweep on the line primitives that model the robot are $r_{base} = 112$ mm, $r_{link1} = 117$ mm, $r_{link2} = 100$ mm, and

Table 4.1 PTP commands that are used in Exp1 for $R1$ and $R2$.

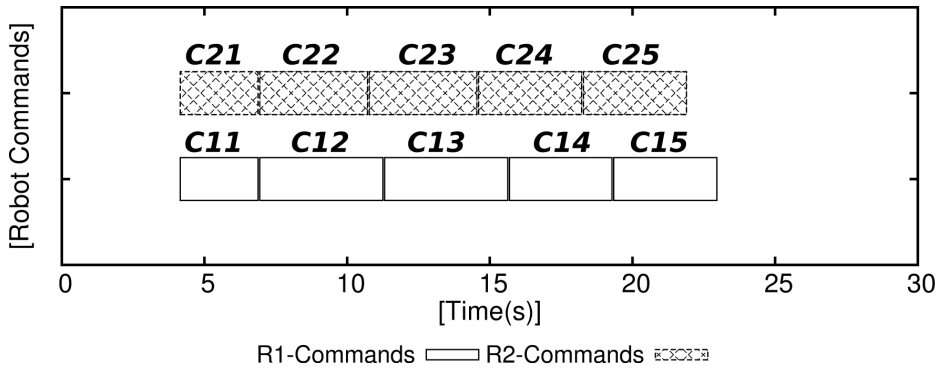
***R*1-Commands** **(*x*, *y*, *z*, roll, pitch, yaw)**	***R*2-Commands** **(*x*, *y*, *z*, roll, pitch, yaw)**
$C11\ (300, 250, 300, 0, 0, 0)$	$C21\ (300, -250, 300, 0, 0, 0)$
$C12\ (500, -50, 360, 0, -20, 0)$	$C22\ (500, 0, 250, 0, -20, 0)$
$C13\ (300, 250, 300, 0, 0, 0)$	$C23\ (300, -250, 300, 0, 0, 0)$
$C14\ (500, 15, 360, 0, 0, 0)$	$C24\ (500, -15, 300, 0, 0, 0)$
$C15\ (300, 250, 300, 0, 0, 0)$	$C25\ (300, -250, 300, 0, 0, 0)$

$r_{EEF} = 48$ mm. Thus, PTP commands are sent to the robots as shown in **Table 4.1**. Each command includes information of EEF posture according to the global coordinate system $[x\ \text{[mm]}, y\ \text{[mm]}, z\ \text{[mm]}, \text{roll [deg]}, \text{pitch [deg]}, \text{yaw [deg]}]$. The initial posture of each robots is assumed to be: $R1_{init}\ (450, 250, 300, 0, 0, 0)$ and $R2_{init}\ (450, -250, 300, 0, 0, 0)$.

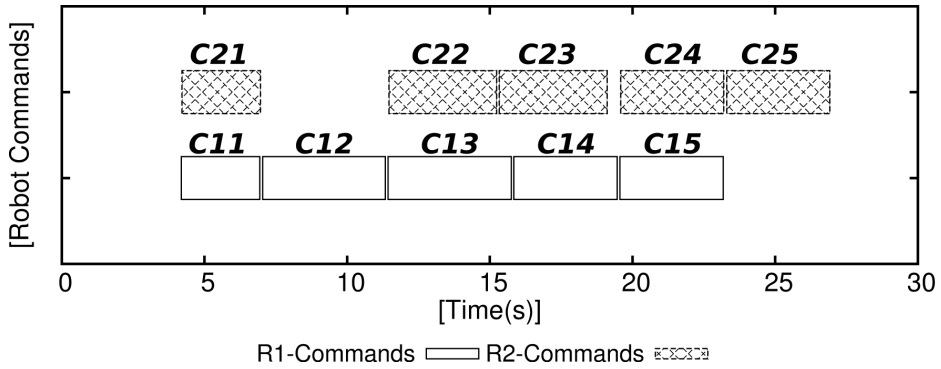
In the first part of the experiment (Exp.1.1), the commands are sent directly to the controller. The execution timing of the commands from the startup of the system are illustrated in **Fig. 4.6(a)**. It is worth mentioning that the period before the execution of the first command is that time required to initialize the system. The minimum distance between each pair of segments of the robots is tracked, and the results for all possibilities of pairs are acquired as shown in **Table 4.2**. The distance curve is shown as a solid line, and the prohibited distance, which is the radii summation of spheres that model the tracked segments, is shown as a dashed line. We can notice that several curves have passed through the prohibited distance, which means that collisions are happened between the robots.

The second part of the experiment (Exp.1.2) has been done after including the collision avoidance module. The execution timing of the commands after applying the collision avoidance method are shown in **Fig. 4.6(b)**. The results of the distance curve of each pair are illustrated in **Table 4.3**. It is observable that all the curves are navigating away from the prohibited distance.

To evaluate the experiment visually, the snapshots of robots' motion are taken in both parts of the experiment. The snapshots are shown in **Fig. 4.7** and **Fig. 4.8** for Exp.1.1 and Exp.1.2 respectively. It is clear that the system can successfully avoid all potential collisions.



(a) Exp1.1: Command chart before applying collision avoidance method.



(b) Exp1.2: Command chart after applying collision avoidance method.

Fig. 4.6 Command execution timing of $R1$ and $R2$ in Exp.1.

4.4.2 Simulation Experiment 2

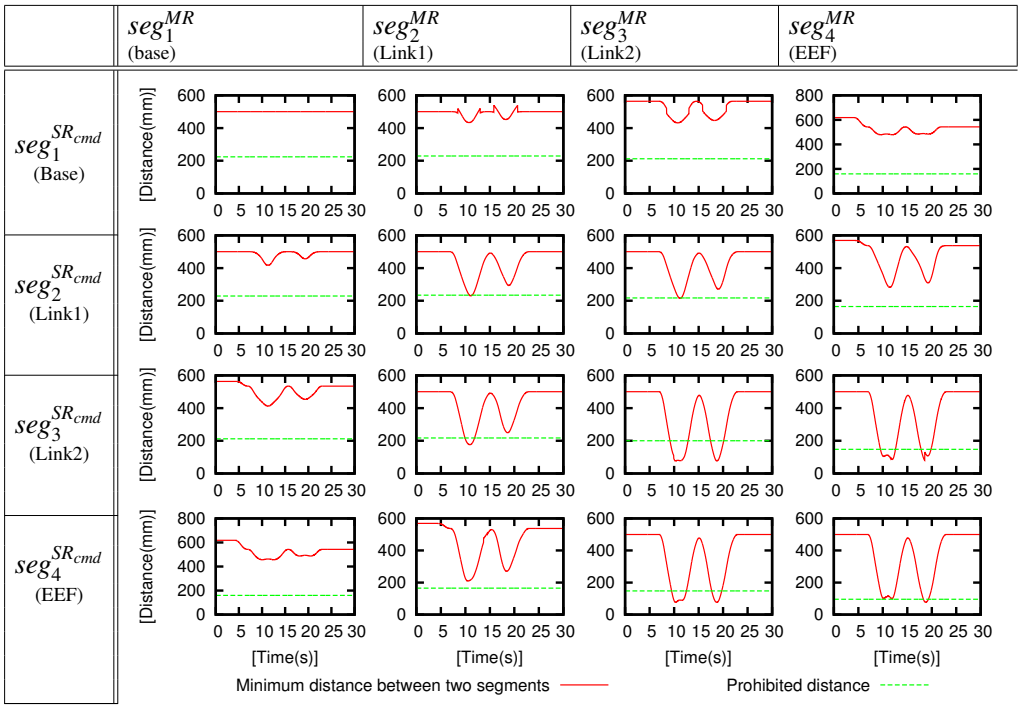
Another simulation experiment is presented. It imitates the same experiment which is performed in [1]. The results will be used for making the final comparison and evaluation between our proposed system and previous system.

Eight PTP commands as illustrated in **Table 4.4** are sent to the system. The maximum velocity, acceleration, and initial position of both robots are set as experiment 1.

In the first part of experiment (Exp.2.1), the commands are sent directly to the controller without applying CA method. The execution timing of the commands are illustrated in **Fig. 4.9(a)**. In addition,the minimum distance between each pair of links are tracked. The distance curves result are shown in **Table 4.5**. It is clear that many distance curves exceed the prohibited distance of pair of links. The snapshot of robots' motion which are illustrated in **Fig. 4.10** showing the visual collision between the robots.

In the second part of experiment (Exp.2.2), the commands are sent to the planner. Thus, the execution timing of the commands are illustrated in **Fig. 4.9(b)**. The tracking result of minimum distance between each pair of links are calculated and they are shown in **Table 4.6**.

Table 4.2 Tracking results of minimum distance between the links of both robots before applying collision avoidance system in Exp. 1.1.



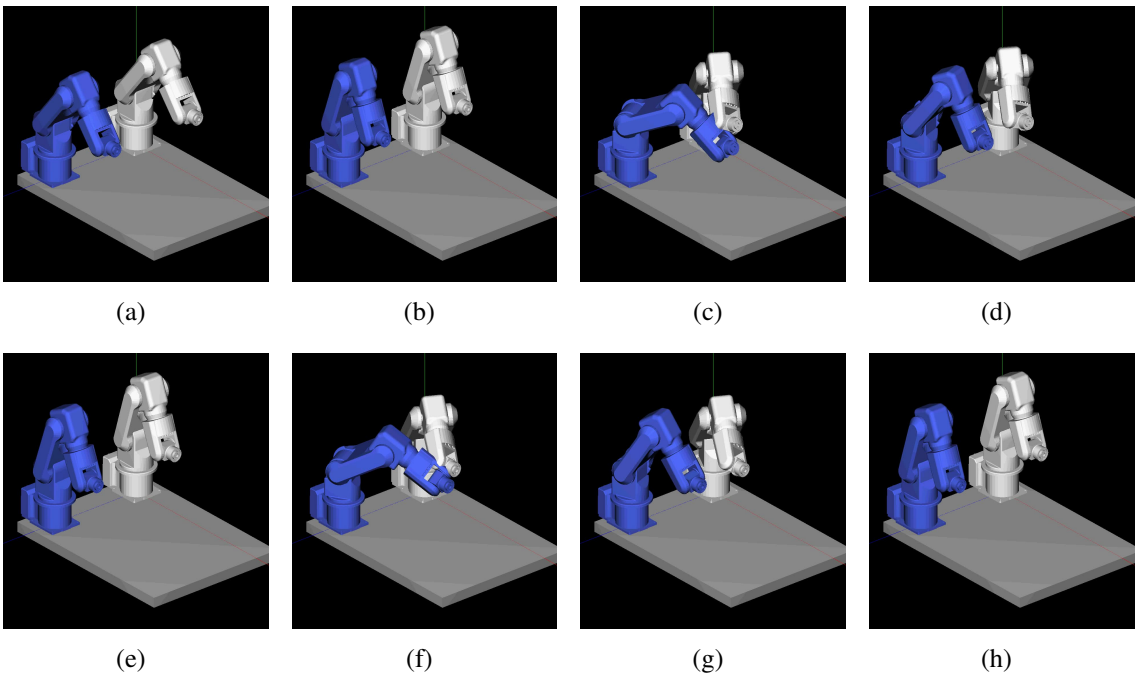
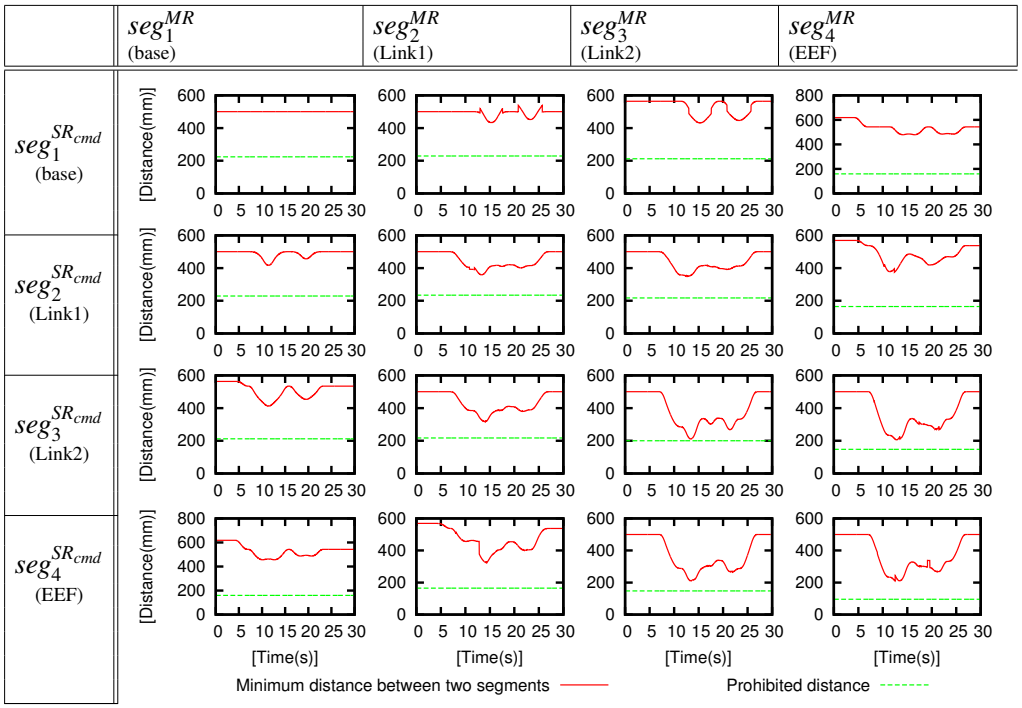
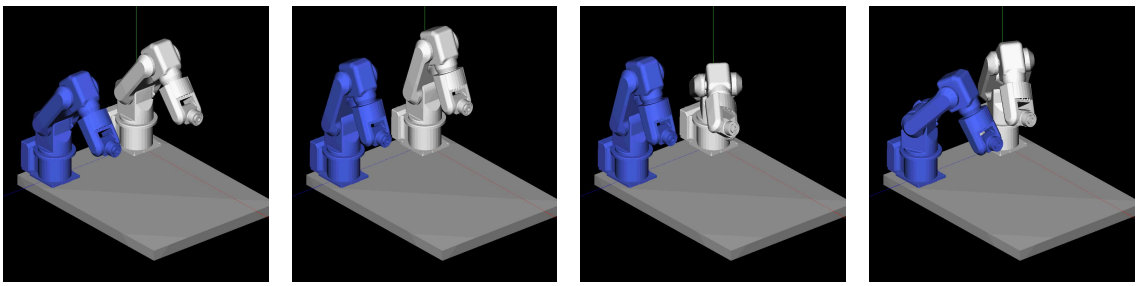


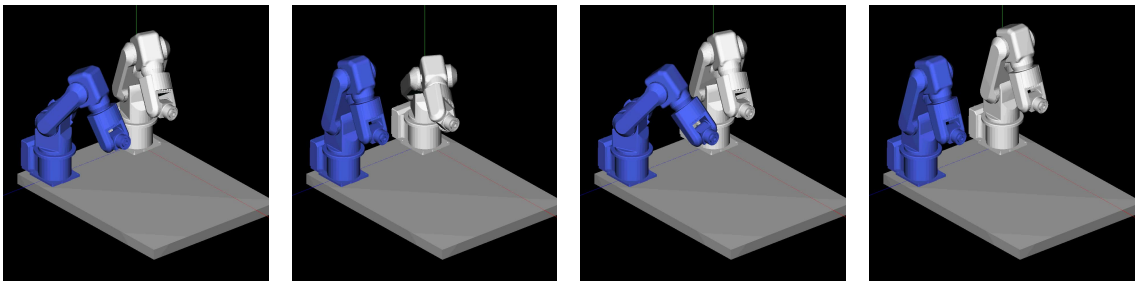
Fig. 4.7 Snapshots for robots' motion in Exp.1.1.

Table 4.3 Tracking results of minimum distance between the links of both robots after applying collision avoidance system in Exp. 1.2.





(a) (b) (c) (d)



(e) (f) (g) (h)

Fig. 4.8 Snapshots for robots' motion in Exp.1.2.

Table 4.4 PTP commands that are used in Exp.2 for $R1$ and $R2$.

***R*1-Commands (*x*,*y*,*z*,roll,pitch,yaw)**	***R*2-Commands (*x*,*y*,*z*,roll,pitch,yaw)**
$C11(300, 250, 330, 0, 0, 0)$	$C21(300, -250, 330, 0, 0, 0)$
$C12(350, 0, 200, 0, 0, 0)$	$C22(350, 0, 200, 0, 0, 0)$
$C13(350, -51, 200, 0, 0, 0)$	$C23(350, 51, 200, 0, 0, 0)$
$C14(400, 250, 330, 0, 0, 0)$	$C24(400, -250, 330, 0, 0, 0)$

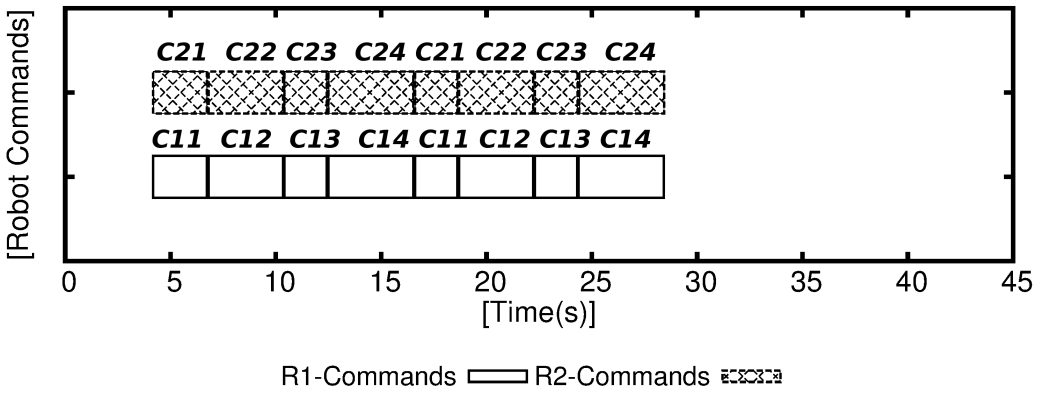
As in the experiment 1, the proposed system could avoid all collisions successfully. Additional visual evaluation of robots' motion showing that both robots are navigating without any collisions. The **Fig. 4.11** illustrates the snapshots of second part of experiment 2.

4.5 Discussion

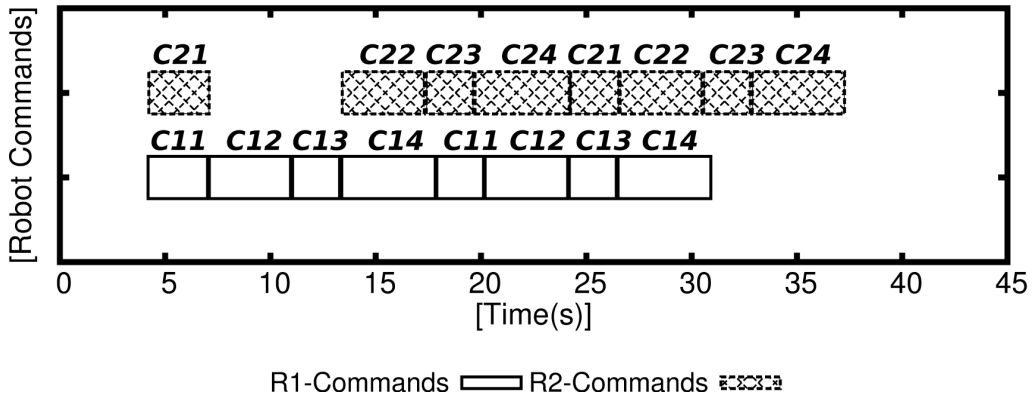
The aforementioned simulation results implemented ACM many times for achieving collision avoidance. The observation of the time dt_m which is required to build each ACM indicates many values. These values are ranging from $[1 \sim 25]$ ms for creating one ACM. The PC, which is used to achieve those experiment, has feature of *Intel Core i5, 2.5GHz* with *memory: 4Gb*. The resulted values for the time dt_m is acceptable to be implemented in on-line. These values can be decreased more with usage of high tech PC.

The proposed system has been evaluated using openGL-based simulator and real measuring of minimum distances between the links of both robots. It becomes obvious that the proposed system can overcome the previously discussed limitation of the collision detection method that developed by Lee et al. [10]. The original collision map devised by Lee illustrates the information on potential collisions between only the EEFs of both robots, whereas in the proposed system, the original map has been improved to have an advanced collision map. The ACM has the same scheme as the original map plus significant advantage of the ability to show the collisions between the whole bodies of the two robots.

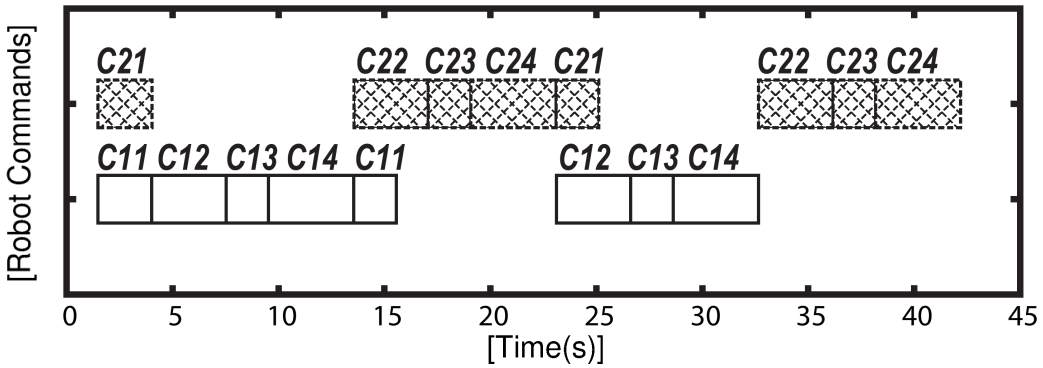
A comparison with the work of Zhou et al. [1], a system which has the same problem formulation as our work, shows that the proposed system proves an advantage in terms of time efficiency. The method proposed by Zhou [1], which is zone-blocking, is avoiding the collisions by determining a specific area in the workspace. If any of both robots is entered inside that area, the flag is activated for preventing other robot to enter that area. Zhou's method is simple and safe but large amounts of time are wasted due to the inability of the



(a) Exp2.1: Command chart without collision avoidance system.



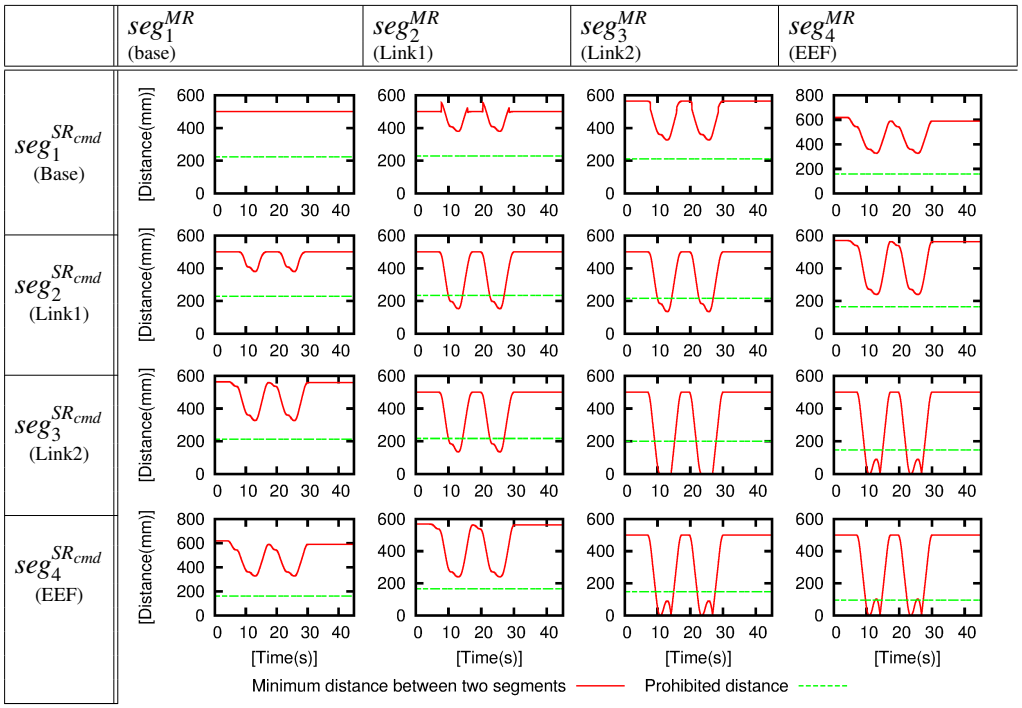
(b) Exp2.2: Command chart after applying collision avoidance system.



(c) Command chart by applying previous method [1].

Fig. 4.9 Commands execution timing of $R1$ and $R2$ in Exp.2.

Table 4.5 Tracking results of minimum distance between the links of both robots before applying collision avoidance system in Exp. 2.1.



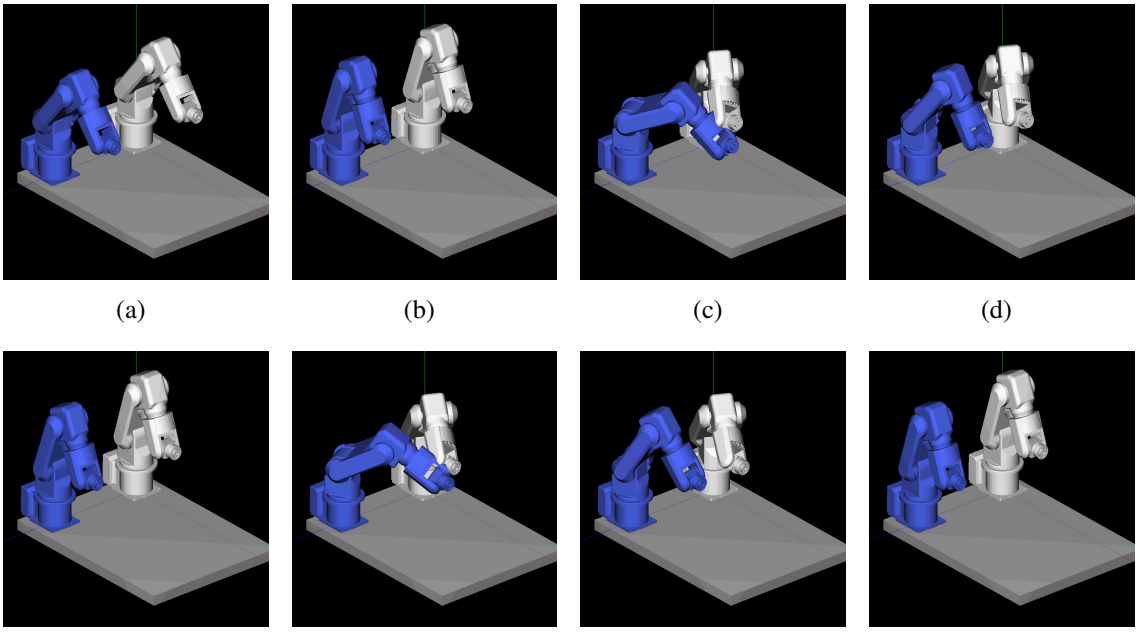
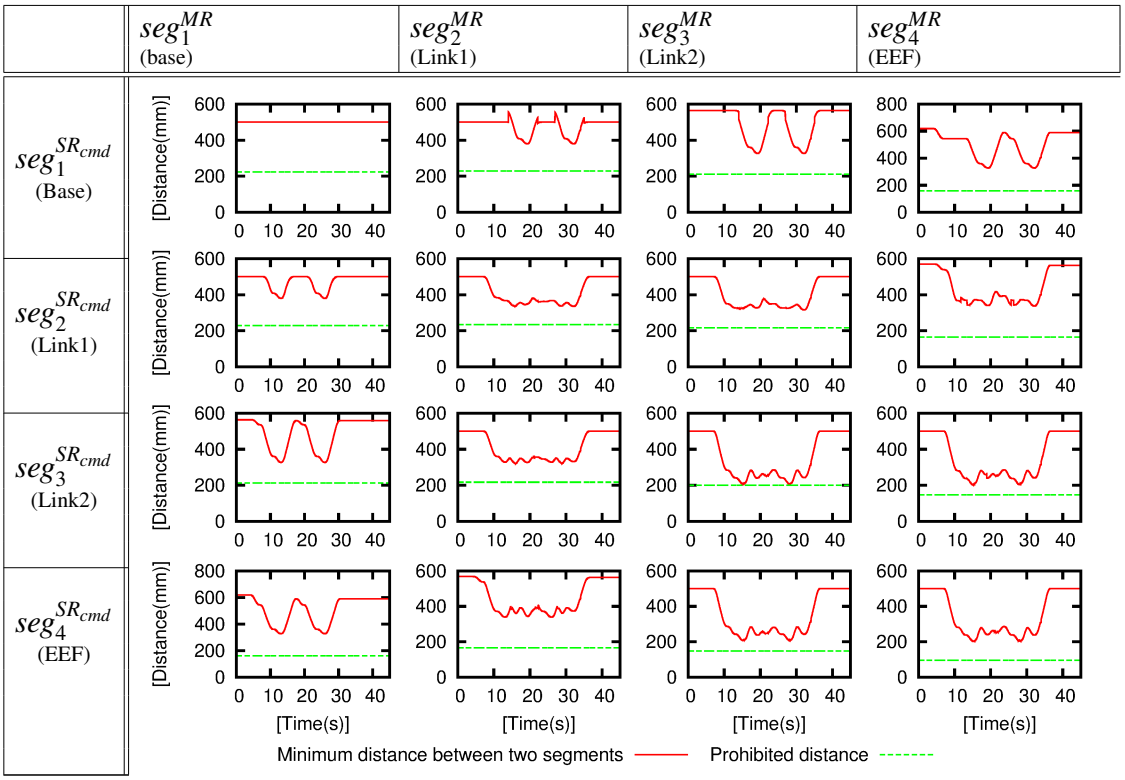


Fig. 4.10 Snapshots for robots' motion in Exp. 2.1.

Table 4.6 Tracking results of minimum distance between the links of both robots after applying collision avoidance system in Exp. 2.2.



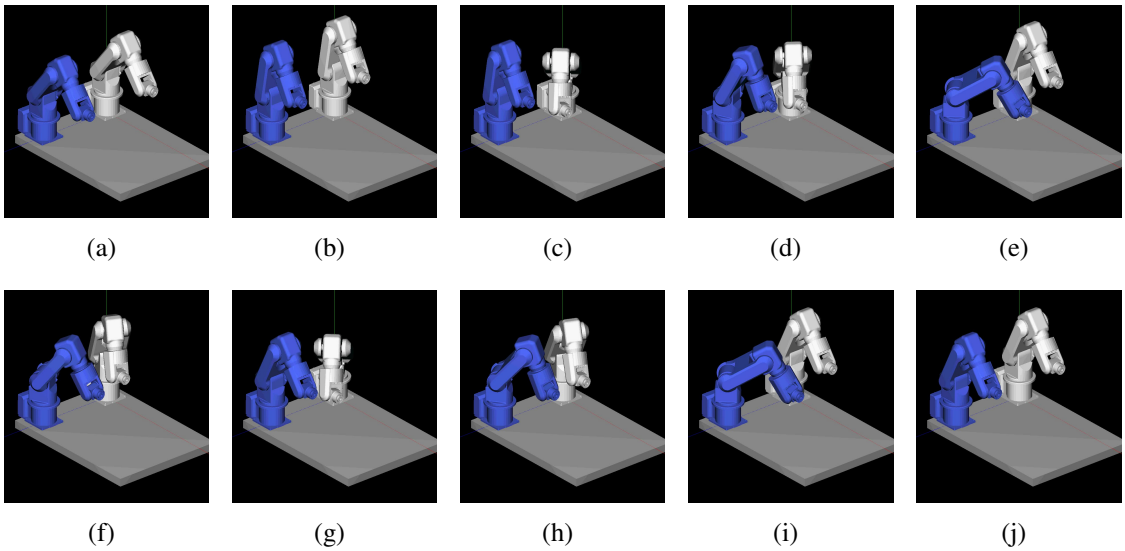


Fig. 4.11 Snapshots for robots' motion in Exp. 2.2.

Table 4.7 Comparison between current and previous methods.

	Without Method	**Previous Method**	**Proposed Method**
CA Method	None	Zone-blocking	Advanced collision map
Geometric Modelling	None	None	SSV
Time spent to execute commands in Exp2	24.3 s	40.7 s	33.1 s

robot to enter the workspace if another robot is operating in it. The experiment described in Zhou's paper [1] has been performed with the same configuration but using our proposed system. The results of this experiment is already demonstrated in Section 4.4.2. The time required to execute all the commands is calculated for both method. Then, a comparison between the proposed method and Zhou's method is performed. The results are shown in **Table 4.7**. As it is illustrated in that table, our proposed method shows a good efficiency comparing with Zhou's mehtod.

Chapter 5

Two-Robot Deadlock Avoidance

This chapter presents the solution for deadlock problem which is common in on-line system. The structure of the chapter starting from Section 5.1 which introduces the deadlock problem of two robots. Section 5.2 describes the method of deadlock avoidance which is developed in this work. Afterwards, some simulation results are illustrated in Section 5.3. Finally, Section 5.4 gives some discussions and conclusions of deadlock avoidance algorithm.

5.1 Introduction

The deadlock is an expression that refers to a situation in which two or more actions are each waiting for the other to finish, and thus neither of them is completed. That means the algorithm is stuck in endless loop with no solution.

In this dissertation, deadlock means the situation in which each robot becomes obstacle to the other, and hence both of them are not able to complete any command as shown in **Fig. 5.1**. The deadlock is common problem in industrial applications which are operated either under off-line or on-line mode. Unlike on-line mode environment, the motion of the robots with off-line mode applications can be planned in advance, and therefore, the deadlocks are avoided before starting the system.

In the previous chapter, two robots collision avoidance system is built neglecting the deadlock cases. The simulation examples in previous chapter have been chosen to be with no deadlock case. But those kind of examples are rare and don't reflect the real application examples which are being used in industry.

To solve the problem of deadlock between two robots, it should be clear when the deadlock happens. In the proposed system, the deadlock situation may occur in two cases regarding the system design as follows:

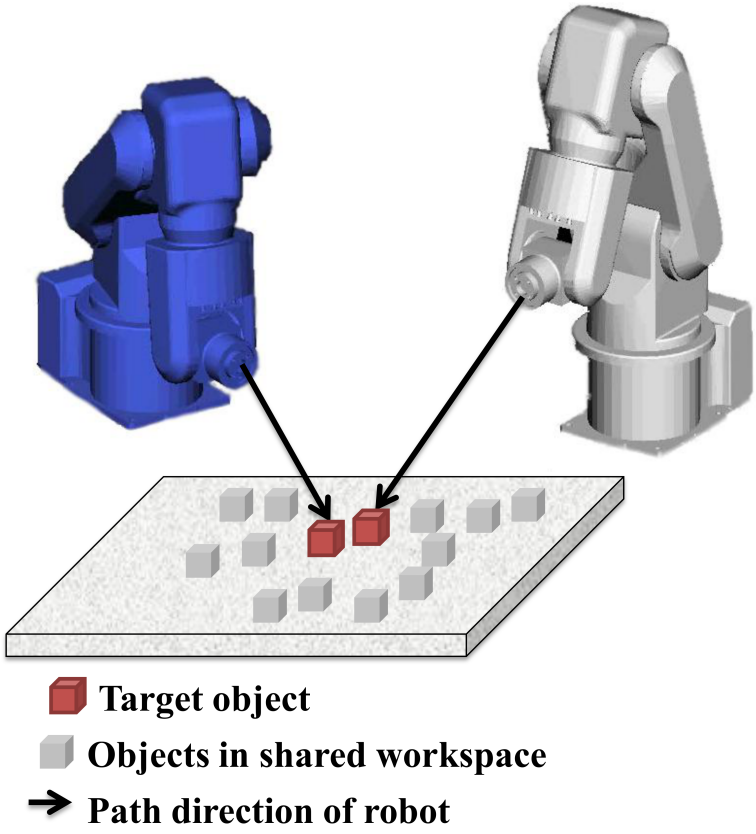


Fig. 5.1 Graphical example of deadlock situation in case of two robots

1. SR_{cmd}-MR combination: When an unavoidable collision areas are obtained from ACM, it will cause deadlock. That means if the output of collision avoidance algorithm is $dt_{cf} = \infty$ as described in Section 4.2.

2. SR_{cmd}-SR combination: According to Section 4.1, if any collision is detected between the robots, it will be judged as unavoidable collision, and thus the robots are in deadlock situation.

In both cases the SR_{cmd} can not execute its command. Those are the only case which causes the system to enter in deadlock situation. Here, the *Deadlock Avoidance* (DA) algorithm is added to previously describe two-robot on-line collision avoidance system. That means, the overall algorithm is able to avoid any collisions and deadlocks between two robots in on-line mode. In the next section, the methodology for avoiding the deadlocks are described in details.

5.2 Methodology

There is no way to avoid deadlock in on-line mode except allowing one robot to escape because the system has no prior knowledge of the PTP commands which will be sent. Thus, the motion of the robots can not be planned before operating the system.

In this dissertation, an *Escaping Technique* is proposed to avoid the deadlocks. The condition for starting the DA process is set to be when the combination is SR_{cmd}-SR only. Therefore, in the first case of deadlock occurrence, SR_{cmd} will wait the MR to accomplish its command and turn into SR mode, and then SR_{cmd}-SR combination can be get. Here, it can be said that SR is the main reason to cause deadlock to other robot. The merit of choosing this combination to start the deadlock process is that both robots are in stopping conditions which gives the generosity of calculation time. That means even if the calculation time is long it will not result to have collision between the robots because both of them are in stop condition.

The key for avoiding any deadlock is to find a safe position of SR, then, command SR to move toward the that position at suitable timing. This position is referred as *Escaping Position*. In this scope, two indispensable information should be decided for calculating the escaping position: The direction of escaping and the distance needed to escape.

The algorithm proposes six directions for escaping, the collision is checked to designated direction. If it is free of collision, it will be the direction of escaping, and then the goal of escaping is set to be the most far point where the robot can reach. On other hand, if proposed direction is not free of collision, the next direction is chosen and so on. Here, ACM is used intelligently to decrease the meaningless travelling if the proposed goal of escaping is far.

The way of deciding direction and distance is going to be explained in next subsections assuming that SR_{cmd} has a PTP command to move from initial position S to target position E with path length $l_{SR_{cmd}}$ and the time needed to reach the target is $t_{tar}^{SR_{cmd}}$.

5.2.1 Direction of Escaping

The direction of deadlock avoidance is decided based on approximated sweeping area of SR_{cmd} and current position of SR which is obstacle in front of SR_{cmd}. The sweeping area of SR_{cmd} is an area which the robot occupies during the motion from initial to target position if the command is executed.

The sweeping area is approximated by a cuboid [Appendix C]. The initial form of the cuboid fits the initial posture of SR_{cmd}. This cuboid is re-formed at each time sample along $t_{tar}^{SR_{cmd}}$ based on the calculation result of SR_{cmd}'s posture at that time. In other words, the cuboid is expanded at each time a joint of the robot exceeds the borders of the cuboid. Thus,

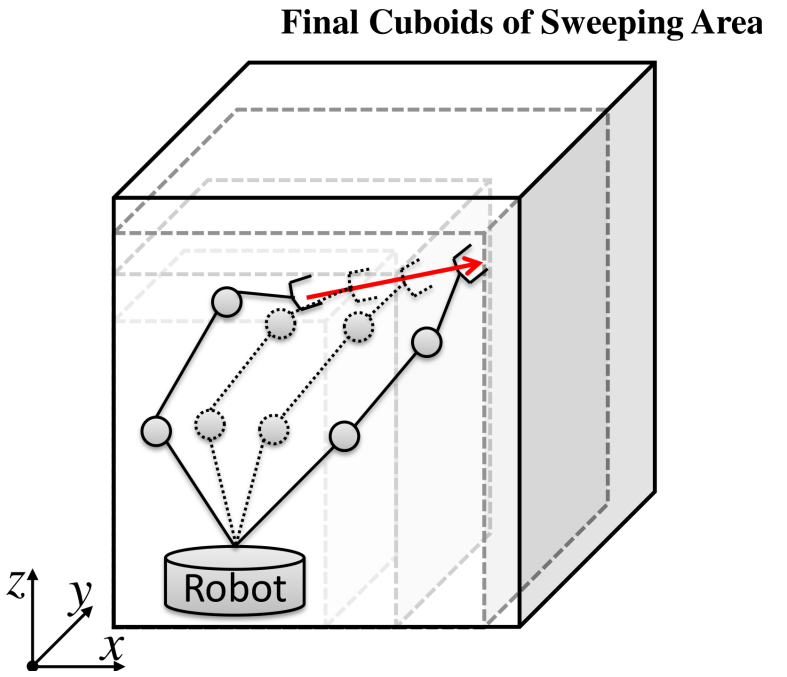


Fig. 5.2 Approximated sweeping area of robot motion using cuboid

at the end of travelling time, a final cuboid is calculated as shown in **Fig. 5.2**. The purpose of forming the cuboid is to inform the other robot, which is SR, about the danger area which should be avoided. Here, the question is: In which direction the SR should escape to avoid the deadlock?

To solve that issue, the distances between the EEF of SR and all sides of the cuboid are calculated. As a result, six possible directions for escaping can be proposed for escaping: X, -X, Y, -Y, Z and -Z as shown in **Fig. 5.3**. Those directions are arranged in ascending order regarding the distances, and then the direction with minimum distance will be suggested to be the direction of escaping at first. Afterwards, collision detection is performed to assure that this direction has no collision. However, if that direction is not collision-free, the next direction will be chosen and so on. If all directions are not free of collision, the algorithm will enter in *Warning Situation* for intervention of human factor to solve the problem.

5.2.2 Distance of Escaping

The distance of escaping can be obtained using advanced collision map. For simplicity, we are going to explain the concept using 2D-space.

Assuming SR_{cmd} will move from S to E. First, the direction of escaping is decided based on the cuboid, then, the escaping goal P is proposed to be a position on the boundary of the workable area of SR in the direction of escaping . This position is referred as

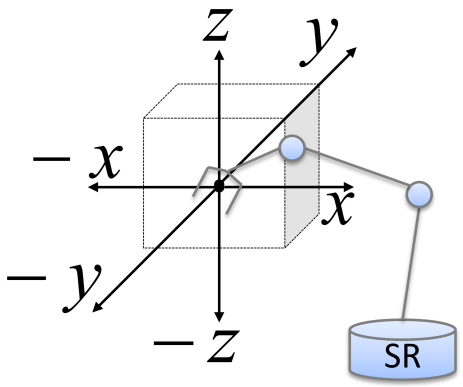


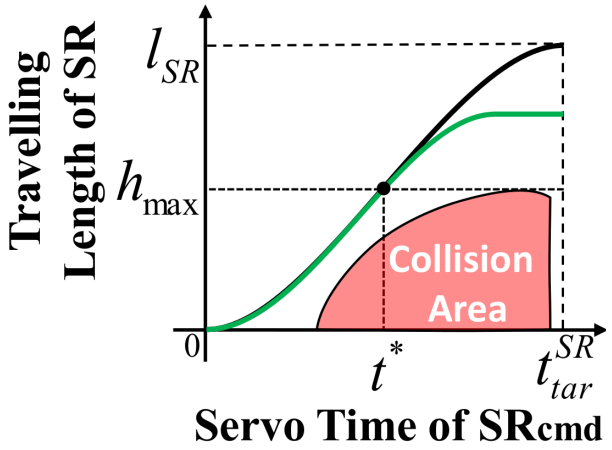
Fig. 5.3 Proposed directions of escaping for deadlock avoidance

Maximum Reachable Goal. However, moving the robot to that goal may fall into wasteful travelling length.

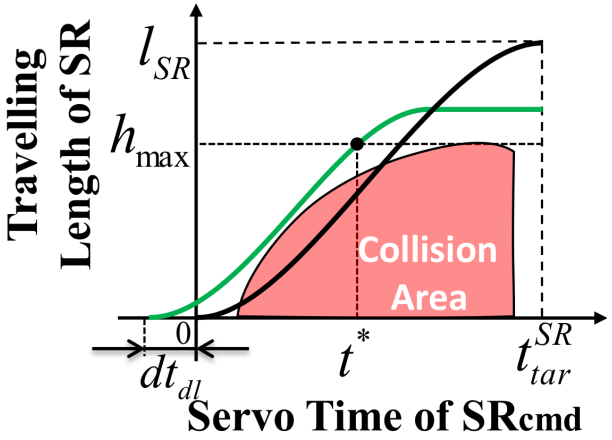
To this end, ACM is used intelligently to decrease the travelling length. As mentioned before, the deadlock is caused by the SR, so that, it becomes obstacle to SR_{cmd}. That means, there is unavoidable collision if SR_{cmd} is executed its command while SR is being at the same position. Thus, creating the ACM for SR_{cmd}-SR combination with newly proposed maximum reachable goal of SR will form a collision area in the lower part of the map. Based on the area, the collision can be avoided. Thus, three possible collision areas can be obtained:

(a) Safe collision area: When the trajectory of SR does not pass through the collision area as shown in **Fig. 5.4(a)**. In this case, decreasing the travelling length of escaping is main goal. By using the characteristic of the ACM, the maximum height of the area h_{max} is calculated. This indicates the maximum travelling length among all collision ranges of the area. Since the trajectory of SR is safe after that travelling length. Thus, the trajectory is modified to start deceleration at t^* whose travelling length is h_{max}. As a result, the modified trajectory will determine final distance of escaping i.e. the final goal for avoiding the deadlock. This goal is called *Real Escaping Goal*.

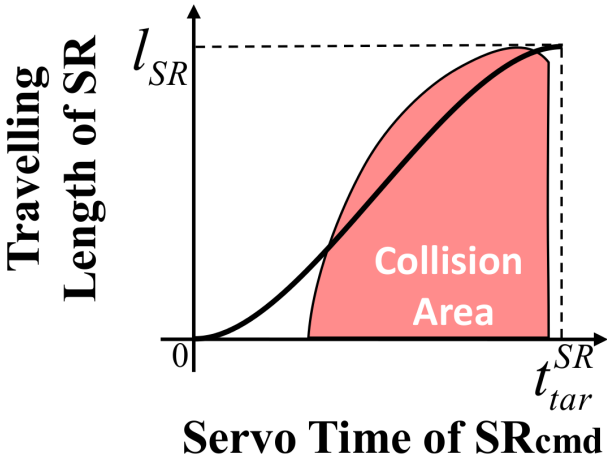
(b) Avoidable collision area: When the trajectory of SR passes through the collision area which has maximum height $h_{max} < l_{SR}$ as shown in **Fig. 5.4(b)**. Where, l_{SR} is the maximum travelling length of SR to reach maximum reachable goal. Since both robots are in stopping condition, the trajectories of both of them is modifiable. Here, the area can be avoided if the trajectory is shifted back to $t < 0$ which means executing SR before SR_{cmd} with necessary delay time dt_{cf}. Furthermore, the trajectory of SR can be modified based on the maximum height of the area as mentioned in the case (a).



(a) Safe collision area



(b) Avoidable collision area



(c) Unavoidable collision area

Fig. 5.4 Possible collision areas in case of deadlock avoidance.

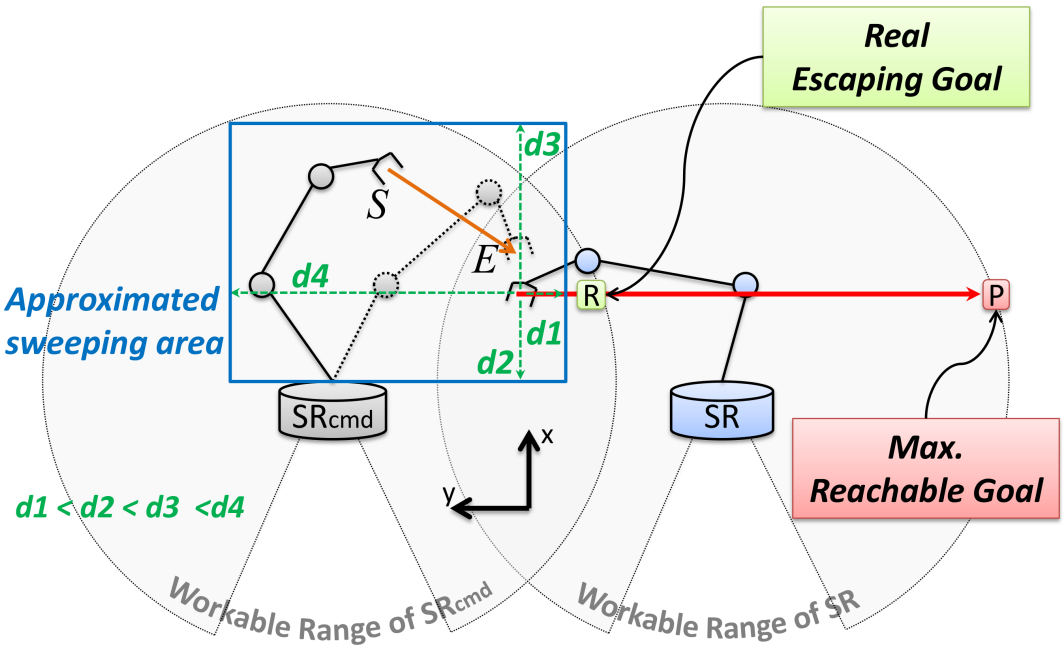


Fig. 5.5 Description of deadlock avoidance concept using 2D-space

(c) Unavoidable collision area: When the maximum height of collision area is equal to maximum travelling length of SR, i.e. $h_{max} = l_{SR}$ as shown in **Fig. 5.4(c)**. This means there is unavoidable collision, so, the direction of escaping should be changed to the next proposed one as mentioned in previous subsection.

An illustrative figure of deciding escaping position in 2D-space is shown in **Fig. 5.5**.

5.2.3 Deadlock Avoidance Algorithm

To understand the methodology of DA, the algorithm is described. As it is illustrated in **Fig. 5.6**, the algorithm of DA start with creating the cuboids of sweeping area of SR_{cmd}. The sweeping area which is the area where the robot is occupied during executing the command. Afterwards, the distances between SR's EEF and all sides of the cuboids are measured. Those distances are arranged in ascending order. Then, the algorithm is entered in a loop to check the collision-free direction starting from direction $k = 1$. The steps are as follows:

1. The algorithm will proposed the maximum reachable goal for escaping to be on the boundary of workable range of SR in the designated direction k.

2. Creating ACM assuming the higher priority is given to SR_{cmd}, so it is MR. And SR with new proposed goal is treated as lower priority, so it is SR_{cmd}.

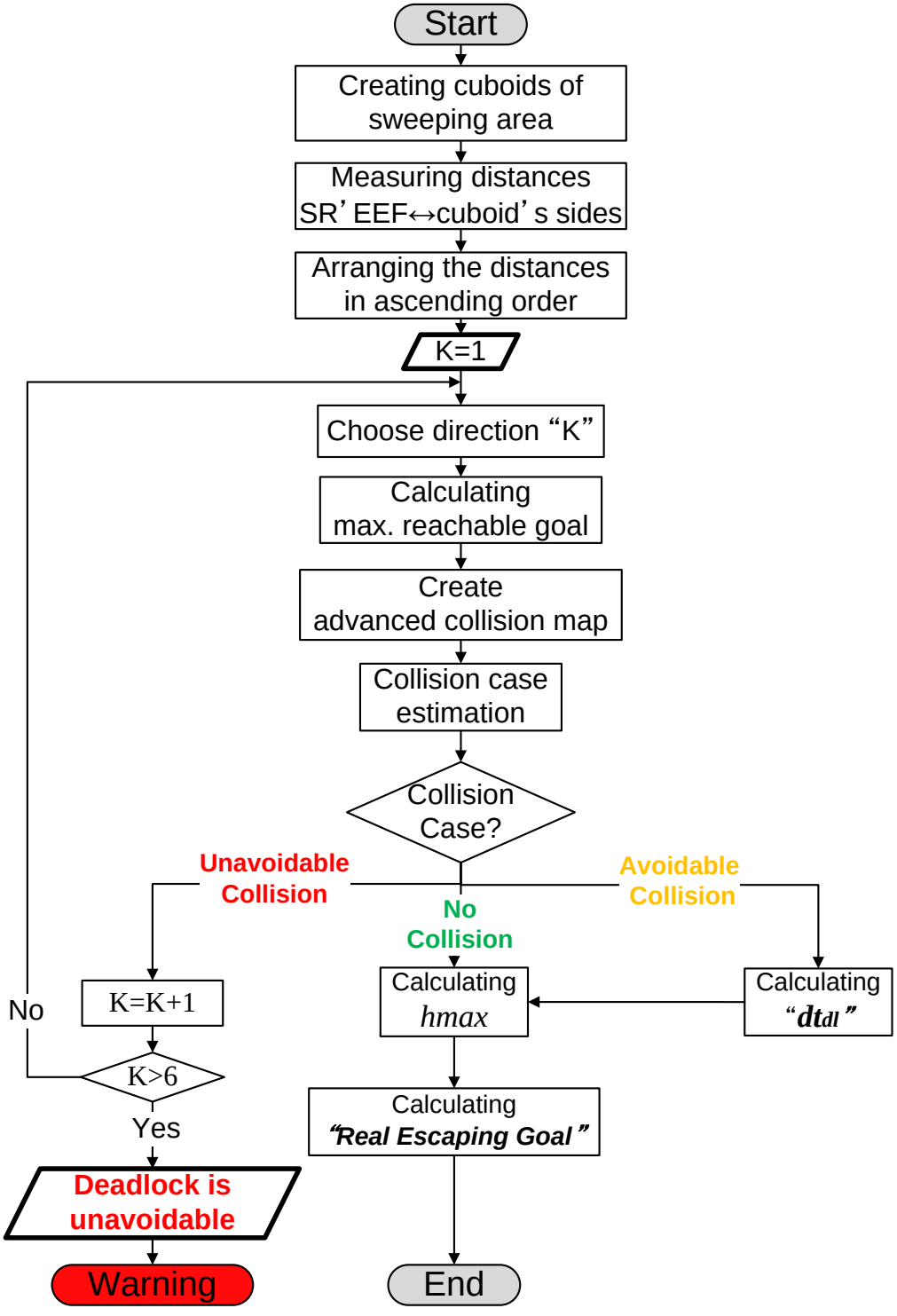


Fig. 5.6 Flow chart of deadlock avoidance algorithm

3. The collision case is examined based on ACM result.

   (a) If the collision case is "No Collision", the possibility of reducing travelling length is examined.

   (b) Thus, h_{max} which is the maximum height of collision area on the ACM is calculated.

   (c) Based on h_{max} the trajectory is modified to start deceleration at that travelling length, and so, the "Real Escaping Goal" is calculated that guarantees deadlock-free path.

   (d) If the collision case is "Avoidable Collision", the necessary delay time dt_{dl} for avoiding the collision area is calculated. Then , steps 3(b) and 3(c) are performed.

   (e) If the collision case is "Unavoidable Collision", the next proposed direction is selected; meant $k = k + 1$

4. In the case that all the directions are not free of collision, the deadlock is not able to be avoided in this algorithm. Thus, the algorithm gives *Warning* in order to allow human factor intervention to solve the problem.

5.3 Simulation Results

The CA system of two robots is tested and evaluated including DA algorithm. Testing is performed using openGL-based simulator which has been already described in Section 4.4.

The simulations imitate movement of two robots that are fixed on a board, with a distance between the central axis of the robots' bases being $500mm$. The global coordinate system is located in the middle between both robot bases.

5.3.1 Simulation Experiment 1

This example uses the PTP commands which are illustrated in the **Table 5.1**. Each command includes information of EEF posture according to the global coordinate system $[x(mm), y(mm), z(mm), roll(deg), pitch(deg), yaw(deg)]$. The experiment is divided into two parts. The first part is performed without collision & deadlock avoidance module; meant with no planner, so, the commands are sent directly to the robot controller. The second part is carried out including collision & deadlock avoidance module.

Let us assume that $R1$ is the gray color robot and $R2$ is the blue robot as shown in **Fig. 4.5**, and the maximum velocity and acceleration of each robot EEF are set to be

Table 5.1 PTP commands which are used to control R1 and R2 in Exp.1-with DA system.

$R1$**-Commands (x,y,z,roll,pitch,yaw)**	$R2$**-Commands (x,y,z,roll,pitch,yaw)**
$C11(300,250,300,-90,0,0)$	$C21(300,-250,300,90,0,0)$
$C12(400,-50,300,-90,0,0)$	$C22(400,50,300,90,0,0)$
$C13(400,-50,100,-90,0,0)$	$C23(400,50,100,90,0,0)$
$C14(400,400,100,0,0,0)$	$C24(400,-400,100,0,0,0)$
$C15(300,250,300,0,0,0)$	$C25(300,-250,300,0,0,0)$

$V1_{max} = V2_{max} = 100mm/s$ and $a1 = a2 = 100mm/s^2$ respectively. The initial posture of the robots' EEFs are $R1_{init}(450,250,300,0,0,0)$ and $R2_{init}(450,-250,300,0,0,0)$.

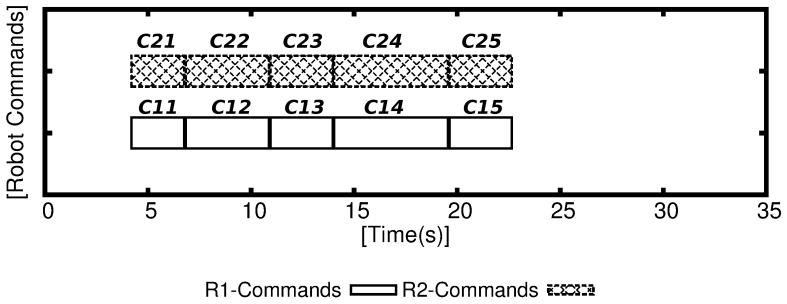
In the first part of the experiment (Exp.1.1), the PTP commands are sent double times directly to the controller. Thus, the commands operation times from the start up of the system are illustrated in **Fig. 5.7(a)**. It is worth mentioning that the period before executing the first command is the time required for initializing the system. The minimum distance between each pair of segments of the robots is tracked, and the results for all possibilities of pairs are acquired as shown in **Table 5.2**. The distance curve is illustrated in a solid red line, and the prohibited distance, which is the radii summation of spheres which model the tracked segments, is shown as a dashed green line. It is notable that several curves have passed through the prohibited distance, which means there are unavoidable collisions.

The second part of the experiment (Exp.1.2) has been done with including the collision & deadlock avoidance module. Thus, the commands operation times for this experiment are shown in **Fig. 5.7(b)**. It can be noticed that there are two deadlocks have been avoided by adding two new commands. The results of the distance curve of each pair are illustrated in **Table 5.3**. It is observable that all curves are navigating away from the prohibited distance. This means the system is able to avoid all potential collisions and deadlocks successfully.

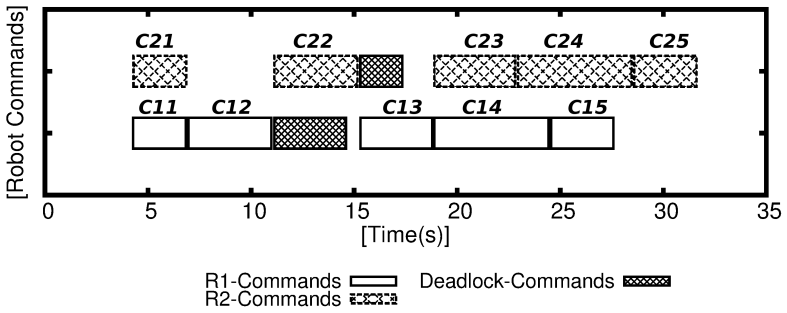
To evaluate the experiment visually, the snapshots of robots' motion are taken in both parts of the experiment. The snapshots are shown in **Fig. 5.8** and **Fig. 5.9** for Exp.1.1 and Exp.1.2 respectively. It is clear that the system can successfully avoid all potential collisions.

5.3.2 Simulation Experiment 2

Another simulation experiment is presented. It imitates the same experiment which is performed in [1]. The results will be used for making the final comparison and evaluation between our proposed system and previous system.



(a) Exp.1.1: Command chart without CA and DA system.



(b) Exp.1.2: Command chart of applying CA and DA system.

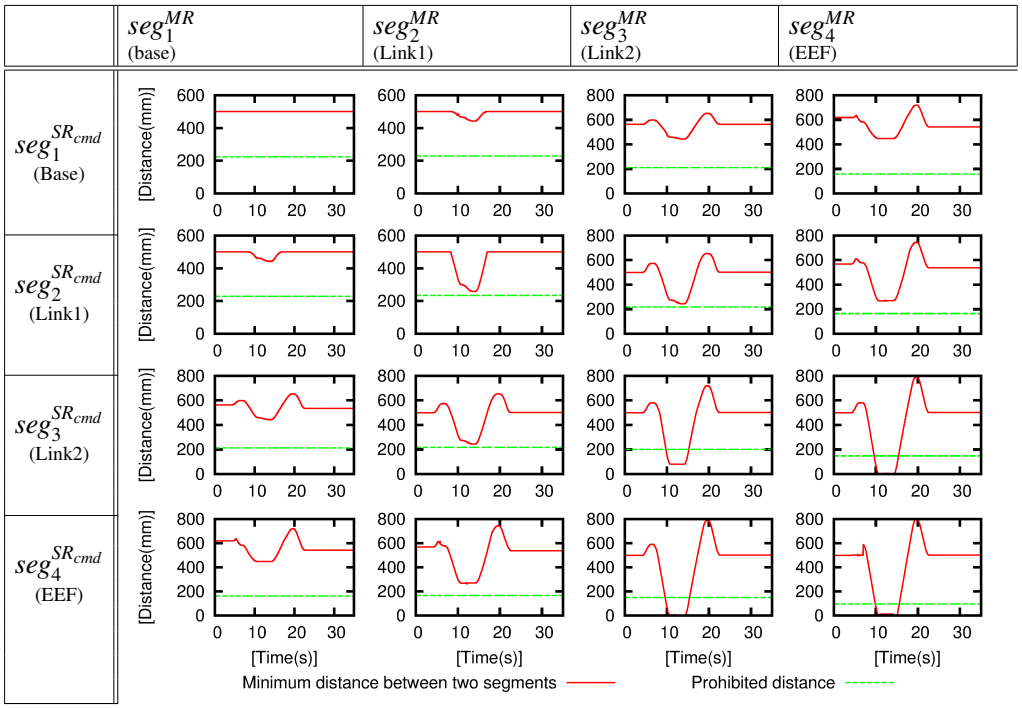
Fig. 5.7 Commands execution timing of R1 and R2 in Exp.1.

This example uses the PTP commands which are illustrated in the **Table 5.4**. The experiment 2 is carried out with the same assumption of experiment 1. Again, the experiment is divided into two parts. The first part is performed without collision & deadlock avoidance module; meant with no planner. Conversely, the second part is carried out including collision & deadlock avoidance module.

In the first part of the experiment (Exp.2.1), the PTP commands are sent double times directly to the controller. Thus, the commands operation times from the start up of the system are illustrated in **Fig. 5.10(a)**. It is worth mentioning that the period before executing the first command is the time required for initializing the system. The minimum distance between each pair of segments of the robots is tracked, and the results for all possibilities of pairs are acquired as shown in **Table 5.5**. The distance curve is illustrated in a solid red line, and the prohibited distance, which is the radii summation of spheres which model the tracked segments, is shown as a dashed green line. It is notable that several curves have passed through the prohibited distance, which means there are unavoidable collisions.

The second part of the experiment (Exp.2.2) has been done with including the collision & deadlock avoidance module. Thus, the commands operation times for this

Table 5.2 Tracking results of minimum distance between the links of both robots without applying CA and DA system in Exp.1.1.



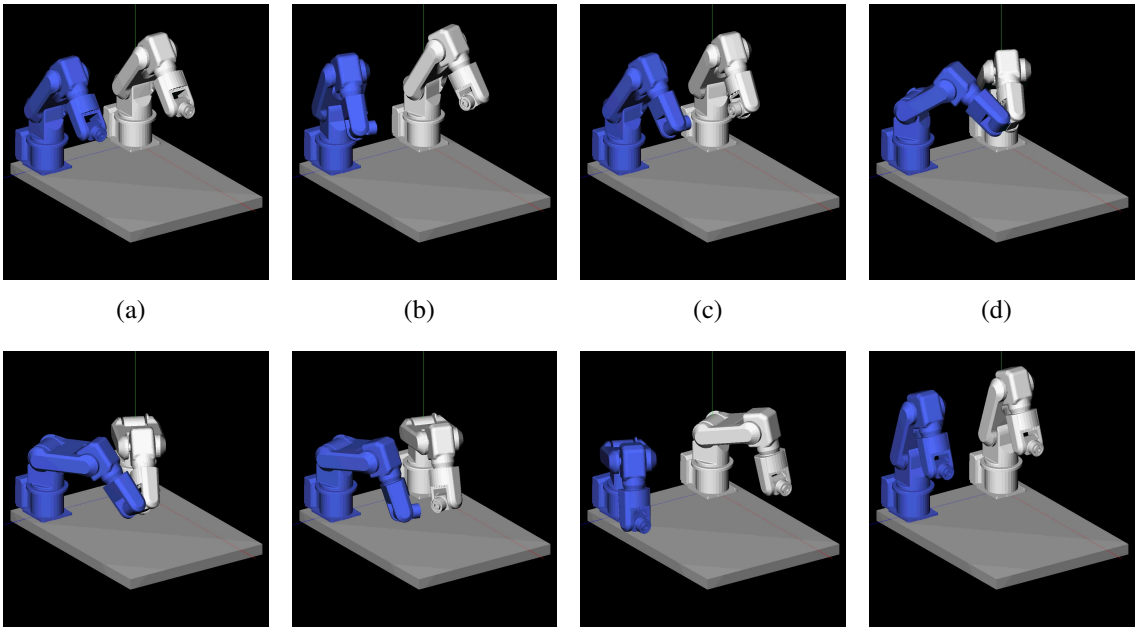
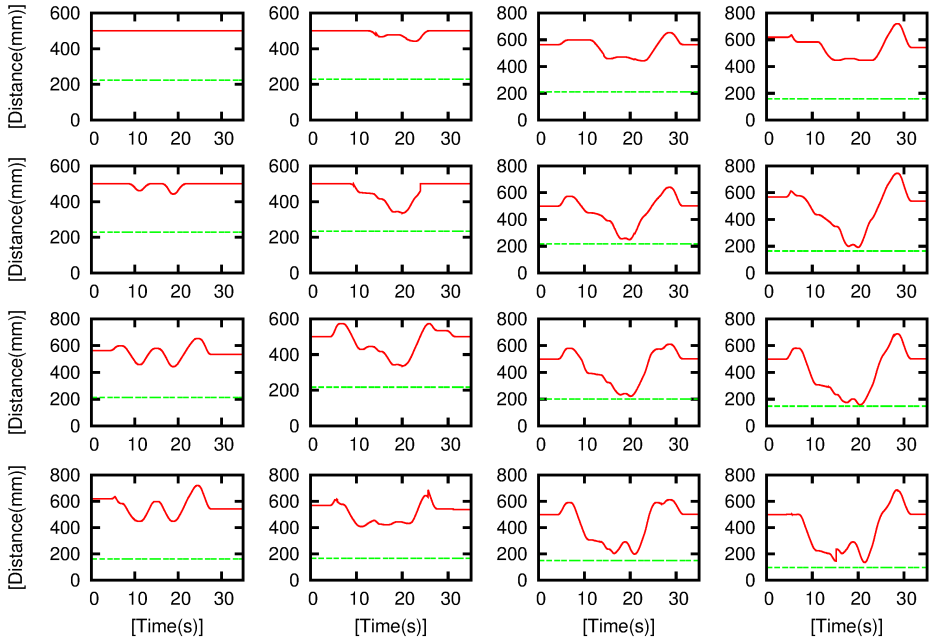
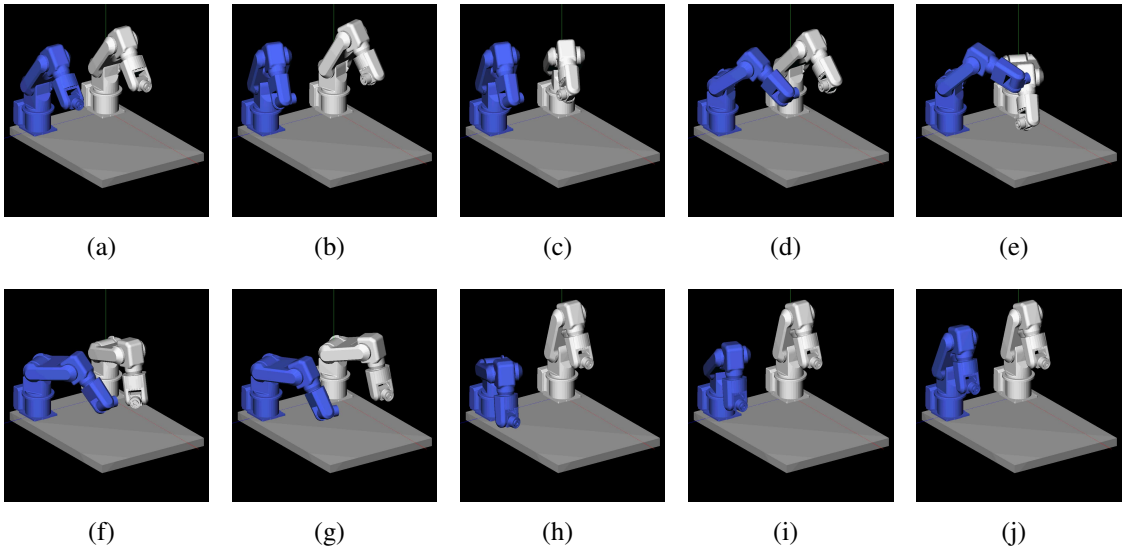


Fig. 5.8 Snapshots for robots' motion in Exp.1.1.

Table 5.3 Tracking results of minimum distance between the links of both robots after applying CA and DA system in Exp.1.2.

	seg_1^{MR} (base)	seg_2^{MR} (Link1)	seg_3^{MR} (Link2)	seg_4^{MR} (EEF)
$seg_1^{SR_{cmd}}$ (Base)				
$seg_2^{SR_{cmd}}$ (Link1)				
$seg_3^{SR_{cmd}}$ (Link2)				
$seg_4^{SR_{cmd}}$ (EEF)				



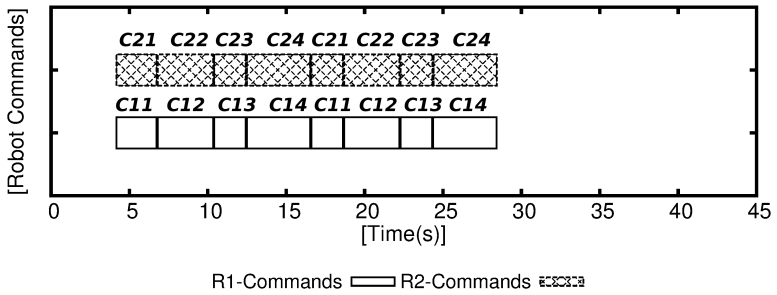
(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

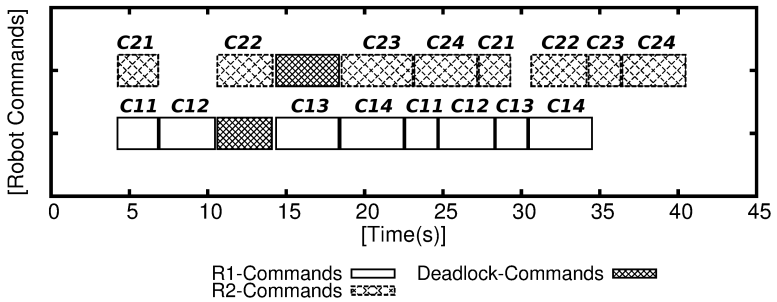
Fig. 5.9 Snapshots for robots' motion in Exp.1.2.

Table 5.4 PTP commands which are used to control R1 and R2 in Exp.2-with DA system.

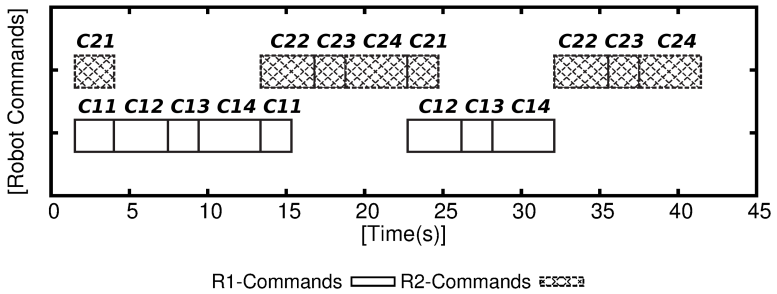
$R1$**-Commands (x,y,z,roll,pitch,yaw)**	$R2$**-Commands (x,y,z,roll,pitch,yaw)**
$C11(300,250,330,0,0,0)$	$C21(300,-250,330,0,0,0)$
$C12(350,0,200,0,0,0)$	$C22(350,0,200,0,0,0)$
$C13(350,-51,200,0,0,0)$	$C23(350,51,200,0,0,0)$
$C14(400,250,330,0,0,0)$	$C24(400,-250,330,0,0,0)$



(a) Exp.2.1:Command chart without CA and DA system.



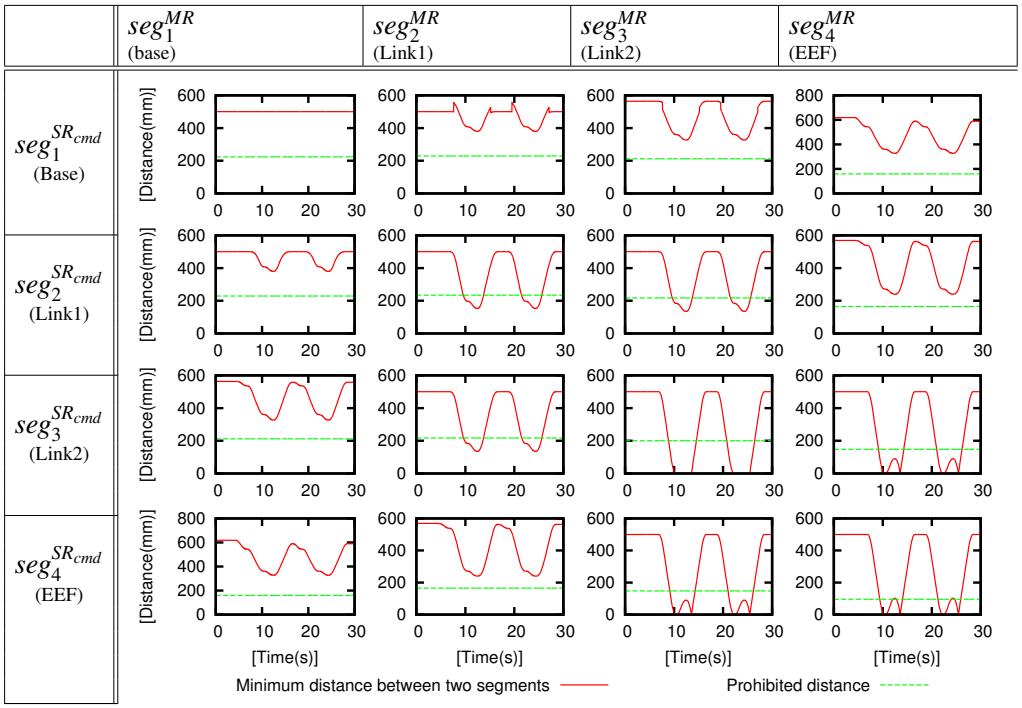
(b) Exp.2.2:Command chart of applying CA and DA system.

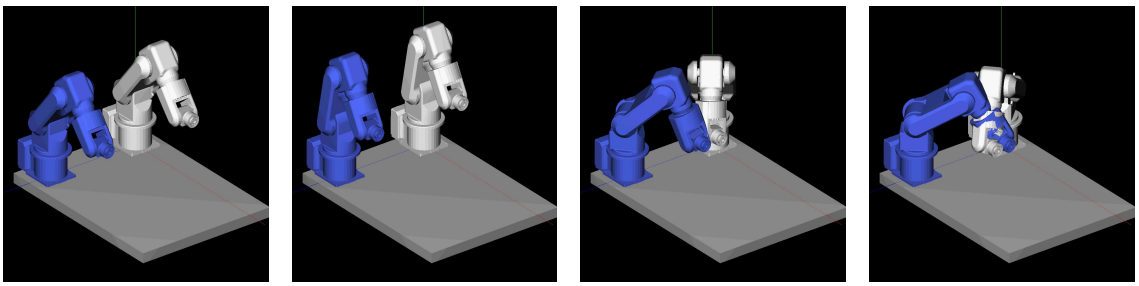


(c) Exp.2.3:Command chart of applying previous method (Zone-blocking [1]).

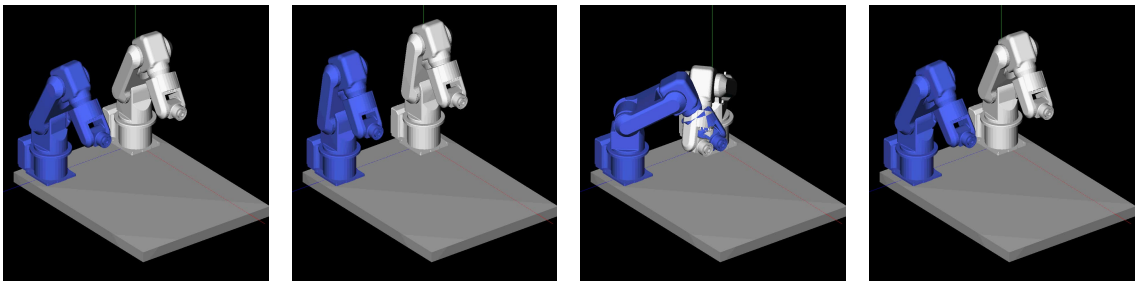
Fig. 5.10 Commands execution timing of R1 and R2 in Exp.2.

Table 5.5 Tracking results of minimum distance between the links of both robots without applying CA and DA system in Exp.2.1.





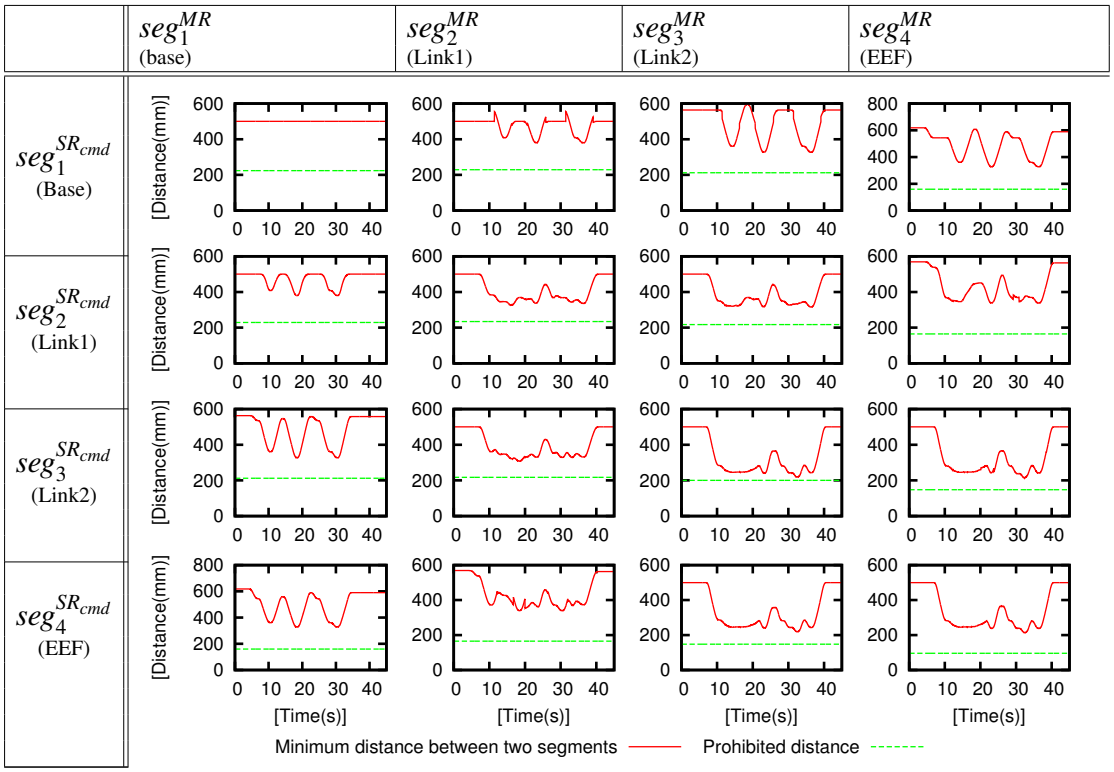
(a) (b) (c) (d)



(e) (f) (g) (h)

Fig. 5.11 Snapshots for robots' motion in Exp.2.1.

Table 5.6 Tracking results of minimum distance between the links of both robots after applying CA and DA system in Exp.2.2.



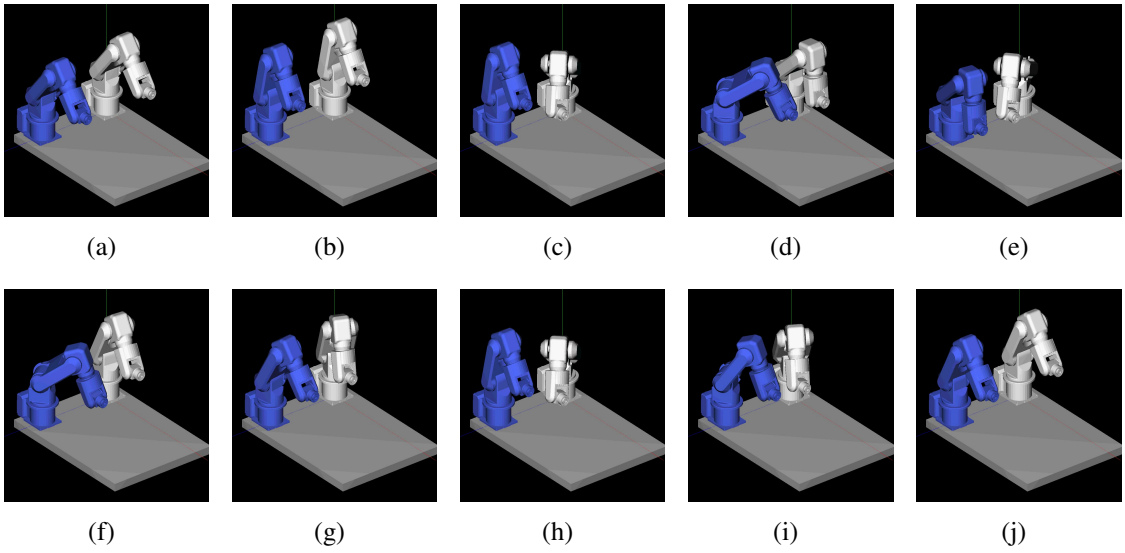


Fig. 5.12 Snapshots for robots' motion in Exp.2.2.

Table 5.7 Comparison between current method including DA and previous method.

	Without CA & DA	**Zhou's Method**	**Proposed Method**
CA Method	None	Zone-blocking	Advanced collision map
Geometric Modelling	None	None	SSV
Time spent to execute commands in Exp2	24.3 s	40.7 s	36.2 s

experiment are shown in **Fig. 5.10(b)**. It can be noticed that there are two deadlocks have been avoided by adding two new commands. The results of the distance curve of each pair are illustrated in **Table 5.6**. It is observable that all curves are navigating away from the prohibited distance. This means the system is able to avoid all potential collisions and deadlocks successfully.

As in the experiment 1, additional visual evaluation of robots' motion shows that both robots are navigating without any collisions. The snapshots of those motion are illustrated in **Fig. 5.11** and **Fig. 5.12** for Exp.2.1 and Exp.2.2 respectively.

5.4 Discussion

Two-robot CA system has been improved. It can detect and avoid the collision between whole body of both robot besides the ability to avoid the deadlocks between the robots which may happen due to unpredictable control commands. The simulations have proved successful results for avoiding the collisions and deadlocks although the distance between the robot manipulators is short. However, some experiments have fallen into local minima situations, i.e. six proposed directions for avoiding the deadlock could not provide collision-free path. In this case, the algorithm will gave up and rely on human factor for helping. Increasing the number of escaping directions could be one solution to decrease the situation of local minima but on other hand a big time cost for calculating collision-free paths should be take into consideration.

The proposed system with CA and DA has been evaluated. The experiment described in Zhou's paper [1] has been performed with the same configuration but using our proposed

system. The results of this experiment have already presented in Section 5.3.2. The time required to execute all the commands has been calculated for both methods. Then, a comparison between the proposed method and Zhou's methods has been performed. The results are shown in **Table 5.7**. As it is illustrated in that table, our proposed method demonstrates good efficiency performance even with the presence of DA algorithm.

Chapter 6

n-Robot Collision and Deadlock Avoidance System

In the previous chapters 4 and 5, the two-robot collision and avoidance system has been described. That was necessary to be performed before generalizing the problem. This chapter explains the final form of on-line collision and deadlock avoidance system including n-robot manipulators. The structure of this chapter starts with Section 6.1 which introduces the necessary modifications for generalizing the concept. Section 6.2 describes detailed design of entire on-line collision and deadlock avoidance (CA and DA) algorithm. Followed by Section 6.3 which presents the new concept of n-robot DA algorithm.Afterwards, Section 6.4 demonstrates some simulations examples to prove the successfulness of proposed system. Finally, some discussions are presented in Section 6.5.

6.1 Introduction

Generalizing the concept of collision and deadlock avoidance to include n-robot in the system playing significant role for future use in the industry. That because many industrial applications devote multiple robot manipulators in the common workspace to perform the tasks.

In Chapter 2, the structure of the system has been described for n-robot case. Then, the problem of collision and deadlock has been addressed only for two robots i.e. $n = 2$. Therefore, the entire algorithm of CA and DA, which is limited to two robots, requires to be modified to suit the case of unlimited robots in the workspace i.e. $n = [2 \sim \infty]$.

Mainly the proposed system includes three techniques that are integrated together using intelligent algorithm to achieve the main goal which is on-line CA and DA. Those techniques

have already been used in case of 2 robots CA and DA system. Therefore, the main modifications in case of n-robot should be performed on those techniques, as well as the entire algorithm which integrates those techniques. As a reminder to those techniques, we are going to list them as follows:

1. Advanced Collision Map (ACM): This has been developed in this work to perform collision detection between the robots. The detail design of ACM in case of two robots has already been explained in Section 3.2.1. Then, the modification which is needed to design n-robot ACM has been explained in Section 3.2.2.

2. Time Scheduling: This technique has been applied to avoid the collision areas on ACM (see Section 4.2). As mentioned in Section 3.2.2, the final form of ACM in case of n-robot is a single map which relates travelling length to servo time. Although with the case of n-robot the number of collision areas on ACM could be more than one, time scheduling technique is valid with no modification. That because this technique is based on shifting the trajectory at each detection of intersection with collision areas on ACM.

3. Escaping Technique: This is also has been invented in this work to avoid the deadlocks which may occur between the robots in on-line systems. The description of modifications of this technique is presented later in Section 6.3.

6.2 n-Robot Collision and Deadlock Avoidance Algorithm

6.2.1 Design of Algorithm

To design n-robot CA and DA algorithm, some strategies are put as follows:

1. The permissible number of SR_{cmd} is only one robot among n-robot which can be MRs or SRs or both of them together. That means, the collision detection starts as soon as having SR_{cmd}.

2. According to our definition of robot modes, having more than two robots in the workspace will result in many combinations. Thus, collision detection is performed for SR_{cmd}-SRs combination at first before SR_{cmd}-MRs combination. That because the detection procedure of SR_{cmd}-SRs is simpler and faster, and thus, if any collision is detected, the collision detection procedure will stop immediately.

3. Owing to existence of many robots in the workspace, decreasing the calculation cost is desired. Thus, the collision detection is performed with close robots. This strategy will be explained in detail in next subsection.

4. Deadlock situation is assumed if unavoidable collision is obtained in either SR_{cmd}-SRs or SR_{cmd}-MRs combinations.

5. DA procedure can only start if the robots' combination is $SR_{cmd} - SRs$, i.e. all robots are in stopping condition.

The n-robot CA and DA algorithm consists of five main processes for each robot in the workspace; which meant the total processes are $5 \times n$. The five processes are listed as follows:

1. *Command Acquisition*, which is responsible to read the control commands.

2. *Collision Detection*, which is specialized to do collision checking between the robots using ACM and simple collision detection based on robots' combinations.

3. *Deadlock Avoidance*, in which the deadlocks between the robots are avoided using a new concept that is developed in this work.

4. *Command Delaying*, which is responsible for avoiding the collisions on ACM.

5. *Command Execution*, in which the acquired command is executed in the controller.

The detailed explanation of each process and how those processes are connected together are presented in next paragraphs.

Command Acquisition

The control commands of each robot are sent by application module to the planner. Those commands are distributed and accumulated in many queues according to the robot which is going to execute them. Those queues are called *Main Message Queues*. Command acquisition process starts with resetting all variables to its initial values except the variables which are dealing with DA process. Then, the process checks two things before starting to read the main message queue of designated robot which are:

1. First, if there is deadlock warning. Here, the process will wait for the DA process to be finished before acquiring commands from *Sub Message Queue*. This queue is responsible for saving the command of DA process, i.e. the command with which the

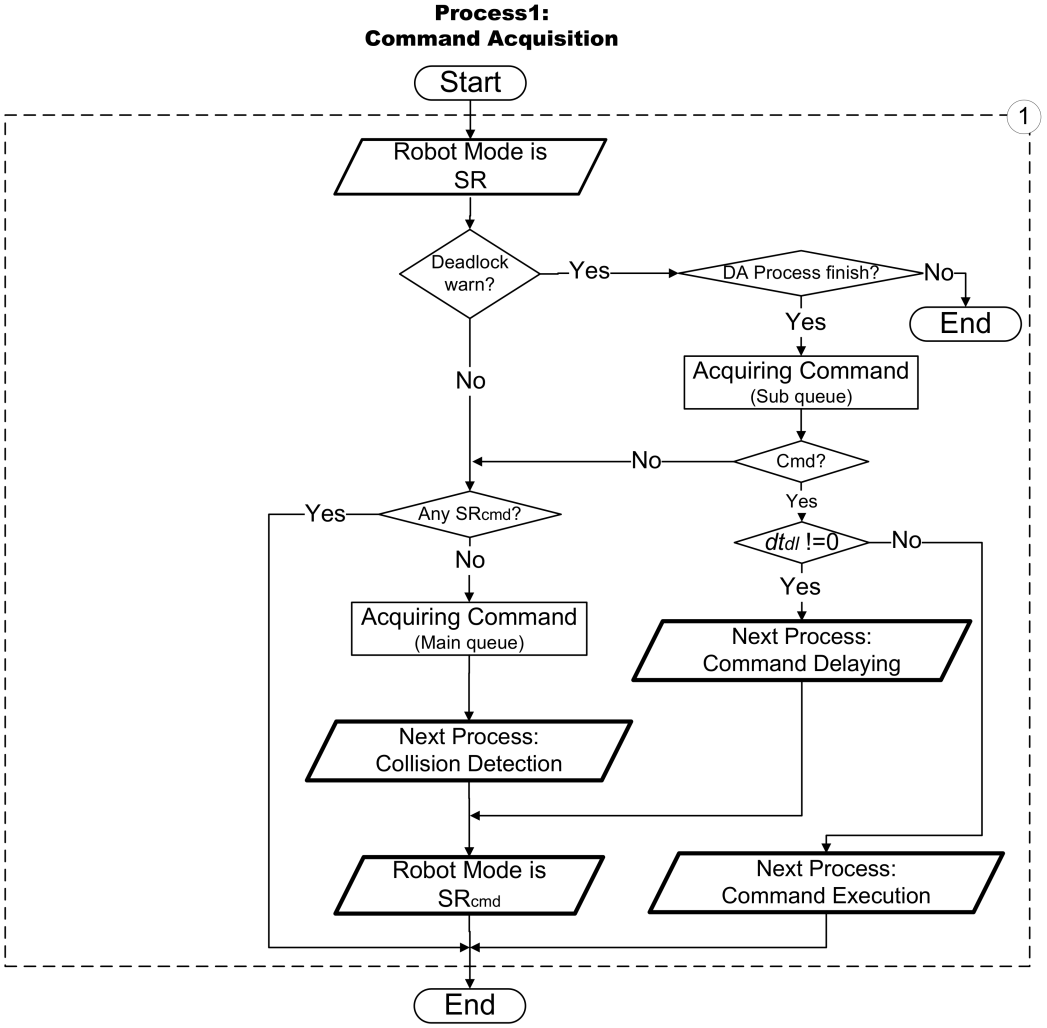


Fig. 6.1 Process 1: Command acquisition

deadlock is avoided if it is executed at suitable time. If there is a command in sub message queue, the process is ended, Otherwise, the process flows to read the main message queue.

2. Second, if there is any SR_{cmd} among the other robots. Here, the process is ended immediately because the condition of n-robot system is to have only one SR_{cmd} in the workspace.

The output of this process is a command information which is acquired from main or sub message queue, and thus robot mode is turned into SR_{cmd}. The description flow chart of this process is shown in **Fig. 6.1**.

Collision Detection

This process starts as soon as acquiring a command by command acquisition process. Here, the other robots' mode are examined. As explained in algorithm strategies, the collision detection will start with SR_{cmd}-SRs combination before SR_{cmd}-MRs as follows:

- First step is for SR_{cmd}-SRs combination. If there is at least one SR, a simple collision detection is performed. Otherwise going to next step directly. That method has already been explained in detail in Section 4.1. The concept is exactly similar but in case of n-robot, the number of checking distances will be increased because there are more than one SR; which meant there are more link segments. Thus, if at least one collision is detected, deadlock warning flag is activated to warn all other robots that there is deadlock situation before ending this process. Otherwise, the process flows to the next step of detection.

- Second step is for SR_{cmd}-MRs combination. If there is at least one MR, ACM is created, otherwise this process is ended. After creating ACM, time scheduling method (see Section 4.1) is applied to estimate the case of collision. Here, three possible collision cases can be obtained as follows:

  1. No Collision, with which the process of collision detection ends.
  2. Unavoidable Collision, with which the deadlock warning is activated before ending the process.
  3. Avoidable Collision, with which the process ends with output dt_{cf}; the necessary delay time to avoid the collision areas on ACM.

The brief flow chart of this process is shown in **Fig. 6.2**.

Deadlock Avoidance

The management of this process is only held by SR_{cmd}. In this process, the deadlocks are avoided using n-robot DA method which will be explained later in Section 6.3. Since the DA method cannot be applied until the combination is SR_{cmd}-SRs. Therefore, if there is any MRs, the process keeps looping until all MRs in the workspace stop and turn into SRs. Afterwards, the DA method is applied. In general, the output of DA method is escaping commands which are sent to sub message queues. And necessary delay times dt_{dl}, if any, for the robots which are going to escape.

Hence, the delay time dt_{dl} is linked with execution timing of one of the robots, we refer to that robot as *Reference Robot*. It is the robot that is going to execute escaping command at

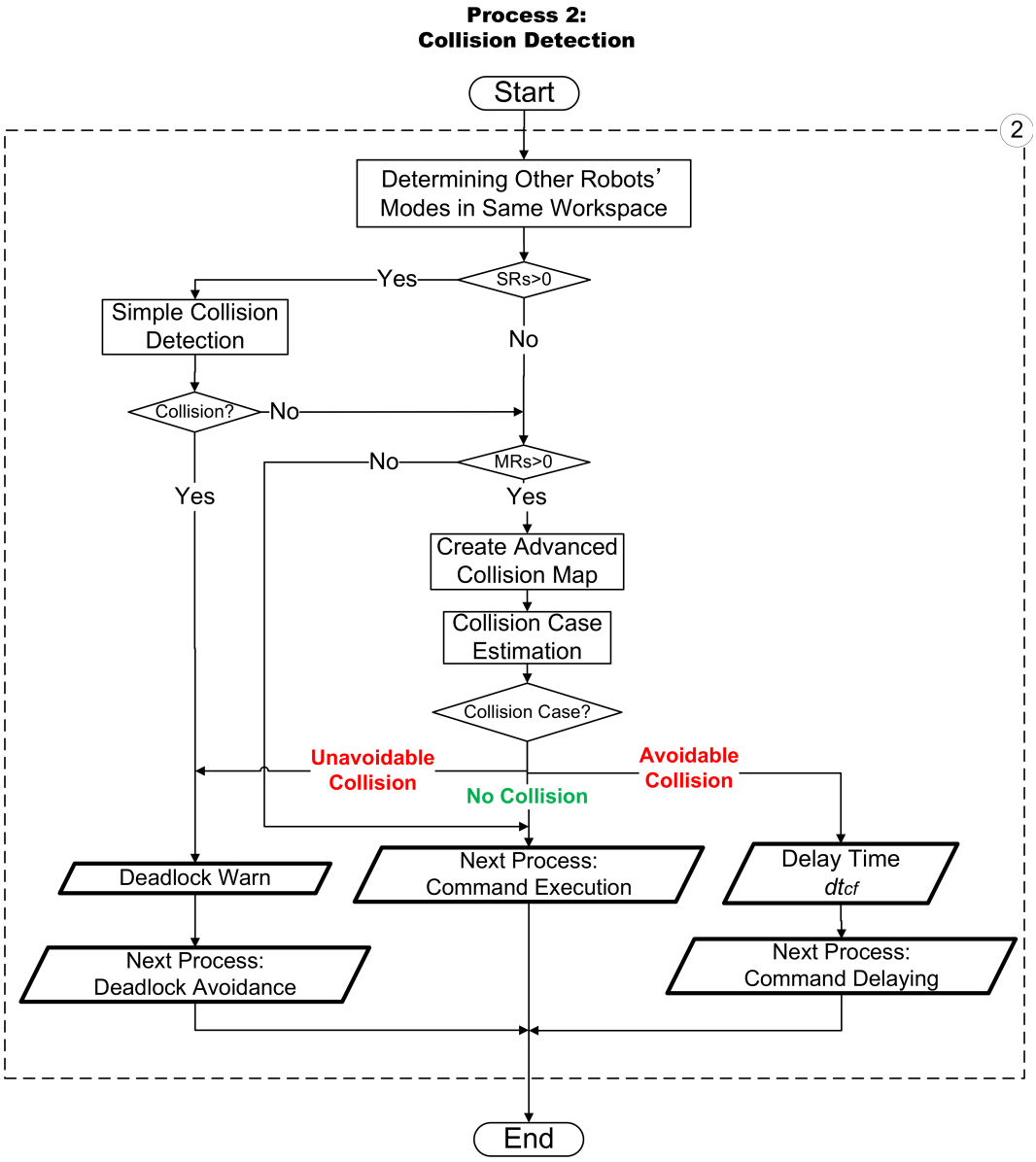


Fig. 6.2 Process 2: Command detection

first before any other robots. Thus, the delay dt_{dl} is performed after assuring that the reference robot is executed. That is managed by command delaying process which is explained next. The flow chart of deadlock avoidance process is illustrated in **Fig. 6.3**.

Command Delaying

Command delaying process is the process where the command, which is acquired for designated robot, is delayed before sending it to the controller. Although the command of the robot is not executed yet in this process the execution time is already decided by previous

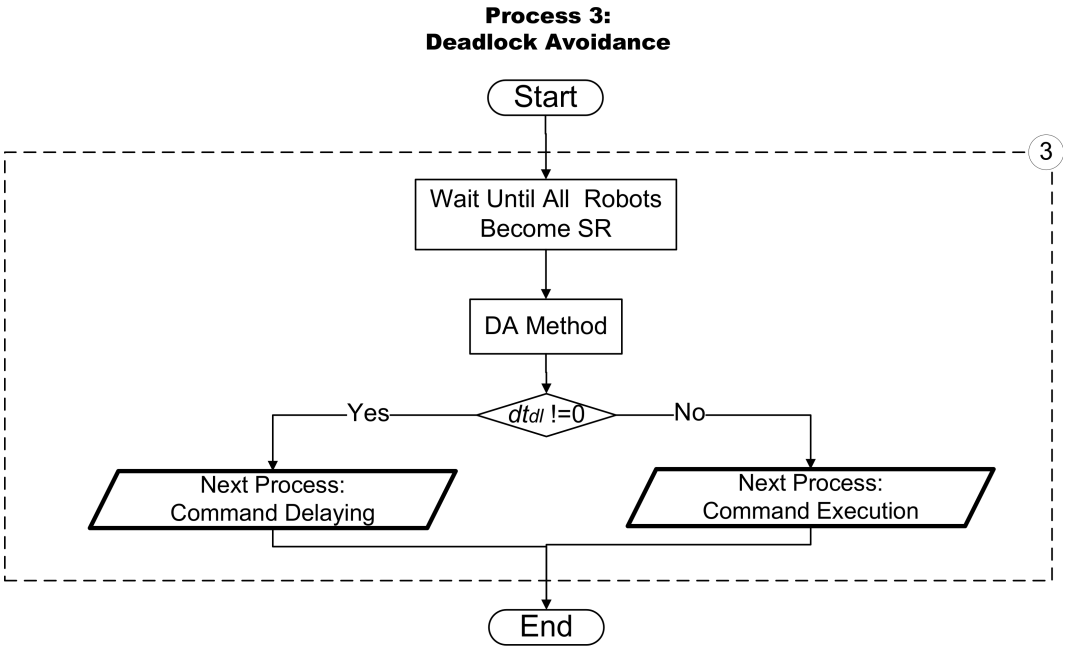


Fig. 6.3 Process 3: Deadlock avoidance

processes; which are collision detection or deadlock avoidance. Therefore, keeping the robot mode as SR_{cmd} causes other robots to wait until the command of SR_{cmd} is executed. That is lose of time, and therefore the robot mode is turned into MR.

To delay the command, a real-time software timer is put in this process. At each cycle, the timer time is compared with the necessary delay time dt_{cf}. If the timer time reaches to delay time the ,process ends. Here, the cycle time is already fixed same as sampling time dt_s to simplify the process of delaying. Therefore, by indicating the cycle time as t_{cycle} and the number of cycles to perform delay time dt_{cf} as n_{cycle}, then:

$$n_{cycle} = \frac{dt_{cf}}{t_{cycle}} \tag{6.1}$$

The process is looping n_{cycle} times before ending.

In case that there is delay time dt_{dl}, as it is explained in previous paragraph, this process will wait a while to ensure that the reference robot executes its command. Then, the delaying starts using software timer same as described before. The description flowchart of this process is shown **Fig. 6.4**.

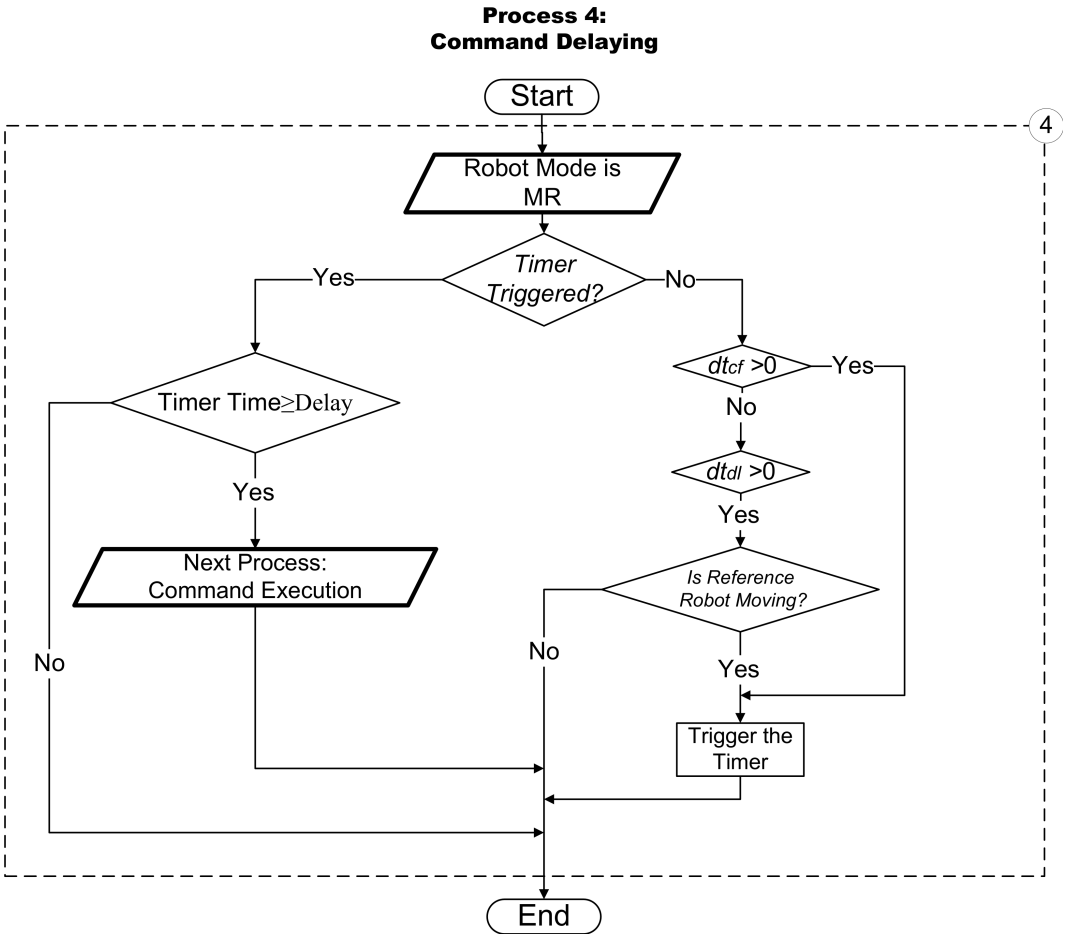


Fig. 6.4 Process 4: Command delaying

Command Execution

Here, the command is going to be executed by sending it to the controller of designated robot, and thus, the robot mode is turned into MR. Then the process loops to check if the executed command is accomplished. That is done by communicating with the controller. As a result, if the command is accomplished, the process ends. The description flowchart of this process is shown **Fig. 6.5**.

6.2.2 Structure of Algorithm

The previous subsection has described the algorithm of each process of a robot; which are five processes. Here, the way of connecting all processes of n-robot is explained.

As mentioned in Chapter 2, the planner is designed to be centralized system, and thus, all robots' processes should be run inside the planner. The problem is how to perform that

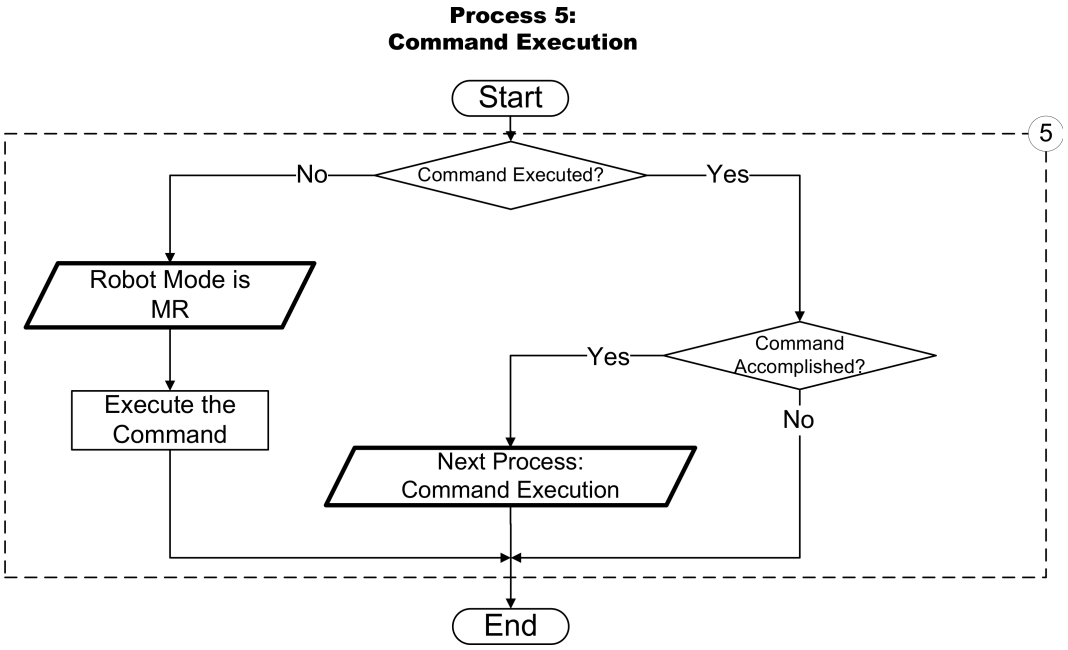


Fig. 6.5 Process 5: Command execution

connection in harmony with other robots and without affecting functionalities of other robots' processes ?

An innovative strategy is put to solve this problem. The entire algorithm consists of n similar sub algorithms; referred to as *S-algo*. Each S-algo includes 5 processes as shown in **Fig. 6.6**. The strategy is to allow only one process of a robot to be run at each cycle time of the planner. In other words, the planner is running within a fixed-time loop which is cycle time t_{cycle}, and thus, at each cycle time, only one process among $5 \times n$ processes is run. Then, next cycle time, a process of next robot is run and so on. Here, it is necessary to set a *Returning Point* for each robot in order to inform the entire algorithm which process of the robot should be run next time. The returning point is set in every process algorithm before ending it. It can be noticed that every algorithm has some outputs with label "Next Process is:...".

Regarding the order of the robots in the entire algorithm is set to be in sequence from 1 to n. That means the priority of robots' order is neglected. This is normal in the proposed system because the condition is independence in work of each robot (see Section 2.2). That means each robot task is performed independently without cooperation with other robots in the workspace.

Based on aforementioned strategy, the final form of entire algorithm is shown in **Fig. 6.7**.

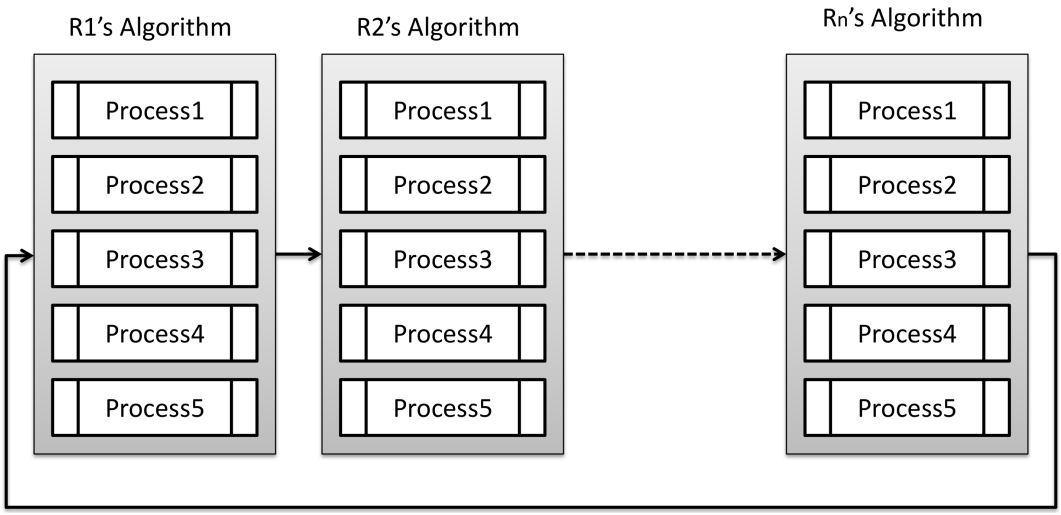


Fig. 6.6 Structure of n-robot CA and DA algorithm

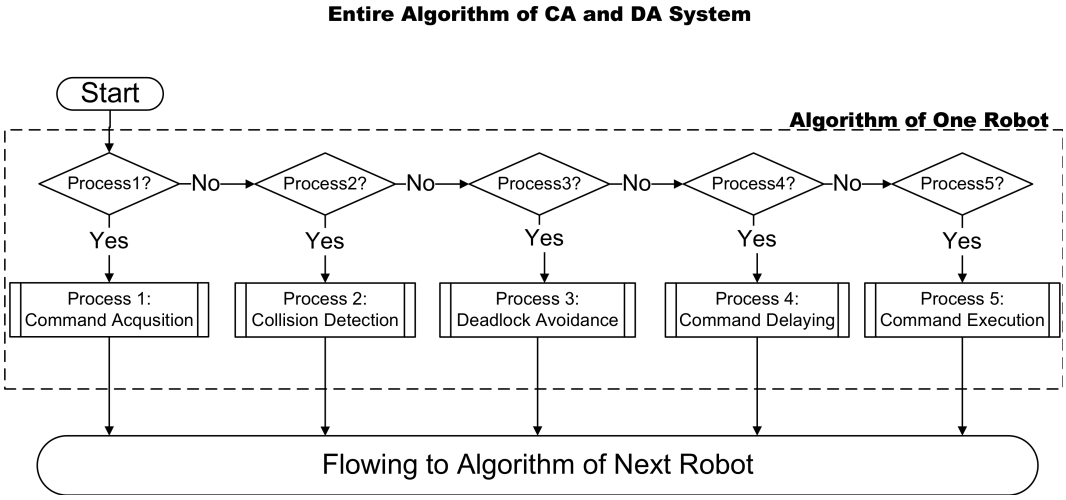


Fig. 6.7 Entire algorithm of CA and DA system

6.2.3 Workspace of n-robot

Some applications devote many robots to be included in the same workspace. On he other hand, others arrange the robots in series to have more than one workspace. Hence, the workspace definition in the proposed system is divided into two classifications as follows:

1. *Shared Workspace*: This contains all robots in the same workspace. That means the workable area of each robot is overlapped with all other robots as shown in **Fig. 6.8**.

2. *Overlapped Workspace*: This contains different number of robots in different workspaces. In this case, the workable area of a robot may overlap with some other robots or all robots as shown in **Fig. 6.9**.

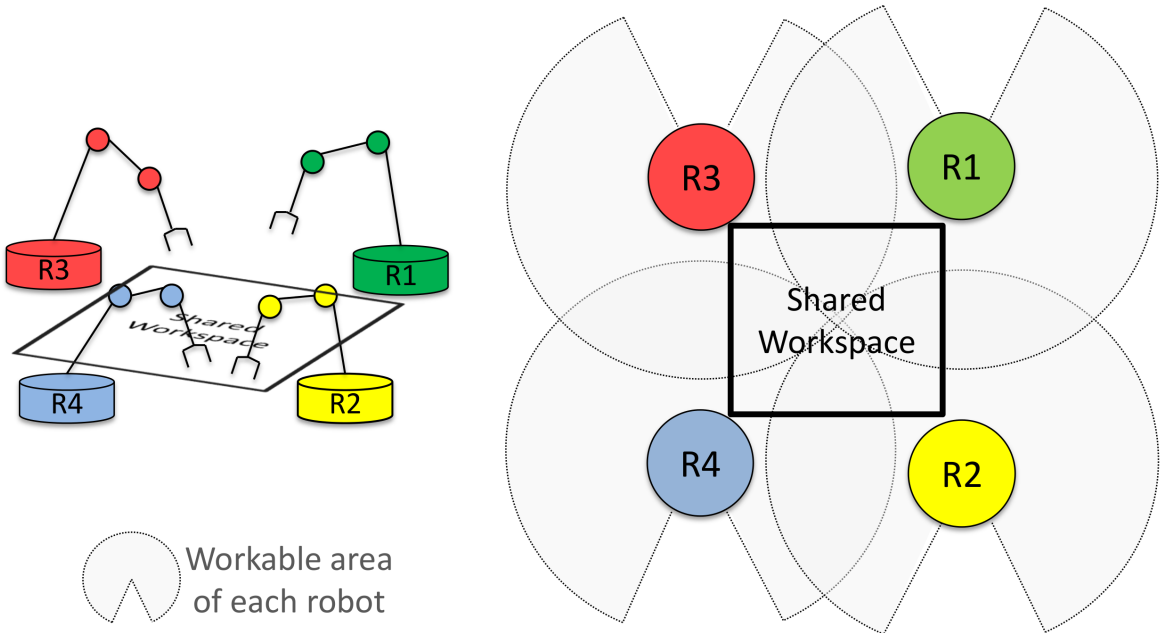


Fig. 6.8 Shared workspace including four robot manipulators.

This classification is put to decrease the calculation cost of collision detection process. The workable areas which are overlapped with each others are determined and put into the database of the system before start up. Consequently, in shared workspace, if a robot turns into SR_{cmd}, it has to perform collision detection with all other robots. Conversely, in overlapped workspace, the robot which turns into SR_{cmd} should perform collision detection with close robots; which meant with the robots that workable areas are overlapped with SR_{cmd}.

Some industrial applications contain huge number of robots in series. Thus, a lot of time is required to do collision detection for that number of robots. However, in reality the maximum number of robot manipulators which can share same workspace will be around eight robots or fewer more. Therefore, with implementing aforementioned strategy, the calculation burdensome is decreased significantly.

6.3 n-Robot Deadlock Avoidance

6.3.1 Introduction

One of the important modifications to n-robot CA and DA system is escaping technique. This technique has been described in detail for two robots (see Chapter 5). With case of two robots, there is only one robot that could cause the deadlock situation. However, including

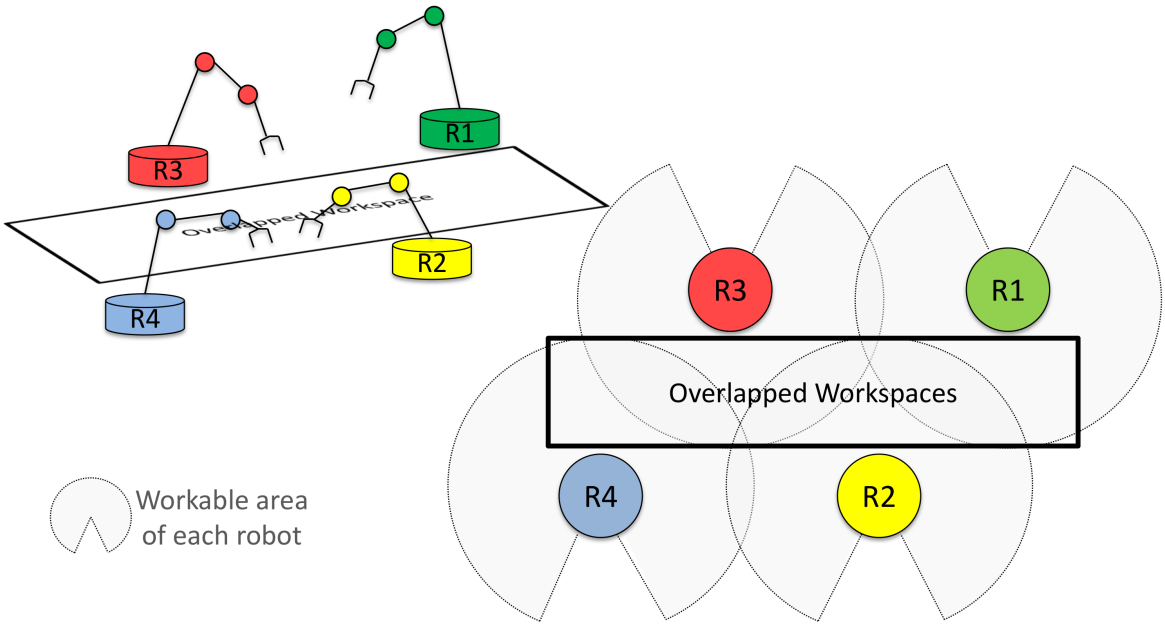


Fig. 6.9 Overlapped workspace including four robot manipulators.

many robots in the proposed system may introduce the deadlock situation by more than one robot at same time. Thus, it is necessary to develop a concept that can solve this problem.

6.3.2 Methodology of n-Robot Deadlock Avoidance

Owing to dividing the collision detection into two steps based on the combination type, the deadlock may occur only in one of both steps:

1. Case of SR_{cmd}-SRs combination: The deadlock occurs if at least one collision is detected for any pair SR_{cmd}-SR_i combination. Where, $i = [1-n]$ indicates the robot ID.

2. Case of SR_{cmd}-MRs combination: Here, the deadlock occurs if an unavoidable collision is obtained after applying time scheduling method (see Section 4.2) on ACM.

As described in previous subsection, The DA process cannot start unless the combination is SR_{cmd}-SRs only. Therefore, in the second case of deadlock occurrence, SR_{cmd} will wait for all MRs to accomplish their commands and turn into SRs mode, then, the combination be SR_{cmd}-SRs.

The previous concept of escaping technique, which is described in Chapter 5, is to find safe position of SR which causes the deadlock, and then, command it to move toward the new position at suitable timing. We referred to this position as *Escaping Position*.

The modified concept of escaping technique is to decide the escaping positions of all obstructive SRs one by one and fix it. In other words, the DA process will be performed for a pair of SR_{cmd}-SR_i regarding all other SRs as obstacles. Then, calculating the escaping position and the time of execution of that SR_i, then fixing it. Next, another pair of SR_{cmd}-SR_j will be considered for deadlock avoidance regarding that SR_{cmd} and previous SR_i have fixed paths and trajectories. This procedure will be repeated until finding escaping positions for all obstructive SRs. Also, with modified concept, the escaping position could be calculated based on two significant parameters: The direction of escaping and the distance needed to escape. Those information are necessary either in previous or modified concept of escaping technique.

To simplify the concept, we are going to explain the steps using 2D space assuming that there are three robots, two of them are obstructive robots SR1 and SR2. In addition, the SR_{cmd} has a PTP command to move from S to E with path length $l_{SR_{cmd}}$ and the time needed to reach the target is $t_{tar}^{SR_{cmd}}$. The steps are going to be explained according to **Fig. 6.10** as follows:

1. *Forming Cuboid*: First step is to approximate the sweeping area occupied by SR_{cmd} during the motion from initial to target position. The method of approximation is based on forming a cuboid which has been described in Section 5.2. The illustrative picture of cuboid is shown in **Fig.5.2**. The purpose of forming the cuboid is informing the other robots about the danger area which should be avoided.

2. *Deciding Direction of Escaping*: This step is based on the cuboid. Here, six directions for escaping are proposed for each obstructive SR which are X, -X, Y, -Y, Z and -Z as mentioned before in **Fig. 5.3**. To decide which direction should be chosen to be the direction of escaping, it is necessary to consider that there could be many obstructive SRs. Thus, which robot should be selected first?

   To solve that issue, the distances between the EEF of each SR and all sides of the cuboid are calculated. Each distance is linked to one direction among the proposed directions. Then the directions of all SRs are arranged in ascending order regarding the distances. Thus, the direction with minimum distance is suggested to be the direction of escaping at first. Accordingly, the escaping order of the SRs will be based on the result of calculated distances. In proposed example, the result of arranging the distances are $d1 < d1' < d2 < d2 < d3 < d3' < d4' < d4$. Where, $d1, d2, d3$, and $d4$ are distances between SR1's EEF and all sides of approximated area. And $d1', d2', d3'$, and $d4'$ are the distances between SR2's EEF and all sides of that area. Thus, the

minimum distance is $d1$ which means SR1 should be selected first to escape in the direction of -Y.

3. *Proposing Goal*: After choosing a direction for one of the robots, a proposed goal for escaping is set to be a position P on the boundary of the workable area of considered SR in the direction of escaping, i.e. SR1 in -Y direction. We refer to this position as *Maximum Reachable Goal* which indicates to maximum distance of escaping.

4. *Collision Detection*: In this step, collision detection is performed between SR_{cmd} and selected robot with newly proposed goal regarding all other robots in the workspace as obstacles. To this end, we assume implicitly that SR_{cmd} as MR because it is the robot which should accomplish its command, so it has high priority. And selected robot as SR_{cmd} whereas all other robots are kept as SRs. According to collision detection strategy which has been mentioned in previous section, the collisions are checked with SRs before MRs. If there is no collision with all SRs, then ACM is created. In this example, collision detection is done for SR_{cmd}-SR2 at first, then SR_{cmd}-SR1 combination.

5. *Deciding Distance of Escaping*: Maximum reachable goal is proposed for escaping, but it may fall into long travelling distance as explained in previous method of escaping technique (see Section 5.2). However, the deadlock is caused by SR1 and SR2, therefore there are unavoidable collisions if they didn't move. Thus, by creating ACM for selected robot SR1 with new proposed goal, an collision area is formed in the lower right part of the map. If a safe or avoidable collision area is obtained, as shown in **Fig. 5.4(a)** and **Fig. 5.4(b)**, the characteristic of ACM is utilized to decrease the long travelling distance if possible. In this example, an avoidable collision area is presented as shown in **Fig. 6.11(a)**. The height h_{max}, which is maximum height of collision area, is used to modify the trajectory of selected robot SR1. As a result, escaping goal is re-calculated to get *Real Escaping Goal*.

6. *Fixing Escaping Goal*: In this step, the real escaping goal of selected robot SR1 is fixed; which meant the path and trajectory of SR1 is fixed. In addition, the trajectory of SR_{cmd} is fixed. That means both robots can be implicitly assumed as MRs.

7. *Finding Escaping Goal of Next SR*: After deciding the path and trajectory of SR1 and fixing them, SR2's escaping goal is calculated using same steps 2, 3, 4, and 5. Here, the only distances $d1', d2', d3'$, and $d4'$ are arranged in ascending order to choose the direction, and then, the maximum reachable goal is proposed. But, to calculate the real escaping goal, it should be considered carefully that the paths and trajectories of SR_{cmd}

and SR1 have already been fixed. Therefore, creating ACM for SR2 while considering SR_{cmd} and SR1 as MRs. An illustrative ACM is shown in **Fig. 6.11(b)**.

6.3.3 Improving the Planning Method of Escaping Goal

The direction of escaping is limited in this method to six directions. Thus, the algorithm should exploit those directions intelligently to provide safe escaping goal for avoiding the deadlocks.

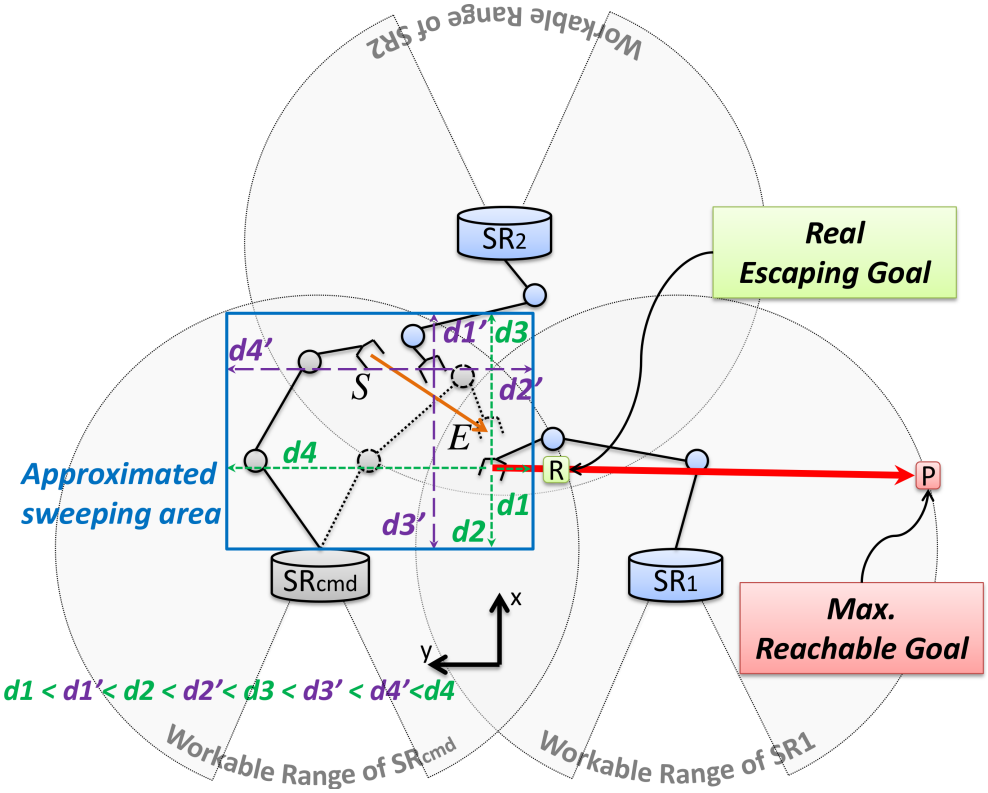
According to the methodology of finding escaping positions, which has been explained in previous Section 6.3.2, it proposes the escaping goal to be maximum reachable position of the robot, then, the collision is examined for that goal, and thus, if there is unavoidable collision, the next direction is chosen. However, in some cases, the maximum reachable goal could be free of collision if it is proposed to be at a shorter position than maximum reachable goal. Those cases usually happen if a robot manipulator is positioned at near maximum reachable goal.

The DA methodology is improved to suit aforementioned cases. Let us assume that there are three robot manipulators R1, R2, and R3 that are positioned in sequence. R2 (SR2) causes deadlock to robot R1 which is SR_{cmd}. R3 (SR3) is positioned near the boundary of workable area of R2. Applying the proposed DA methodology, the algorithm is suggested to make the escaping goal at position P which is not free of collision because R3 is very near to that position. Here, the improved methodology suggests to perform simple collision check between R3 and R2 which is proposed to move to maximum reachable goal. And then, when first collision is detected at certain time t^*, it will indicate that all travelling length before that time is safe and free of collision. Thus, the algorithm sets a new proposed goal of to be at travelling length $TL(t^* - dt_s)$, where dt_s is sampling time. This goal is called *Safe Reachable Goal* as shown in **Fig. 6.12**. Afterwards, collision detection of SR_{cmd}-SR2 is performed for newly proposed safe reachable goal. Then the improved methodology follows the same steps 5, 6, and 7 of aforementioned methodology as explained in previous Section 6.3.2.

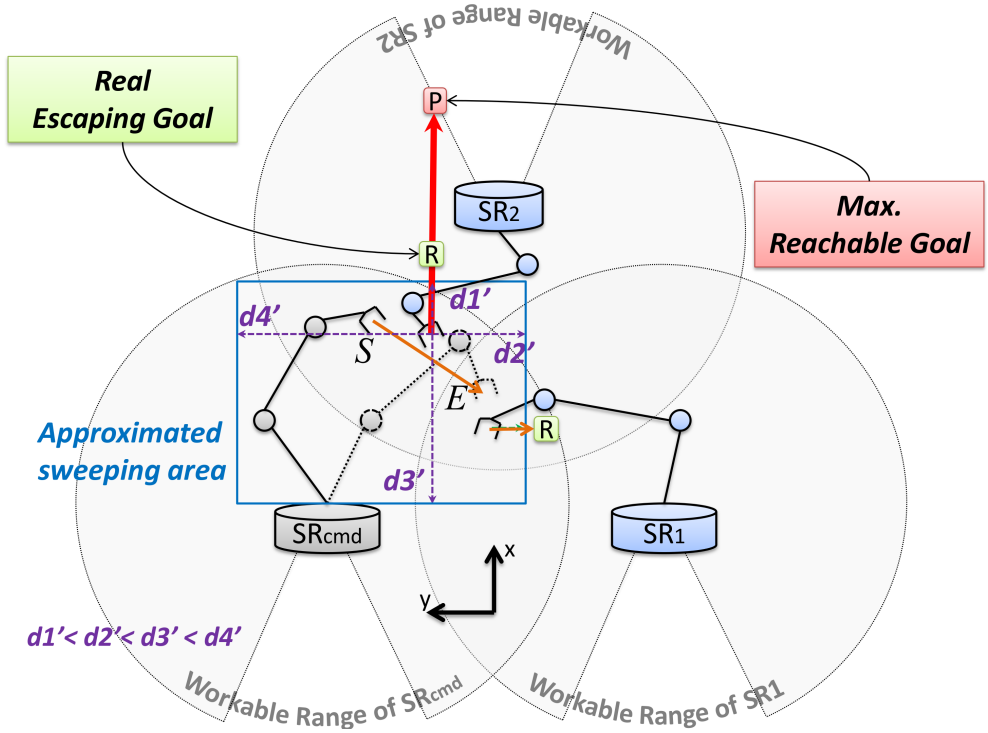
6.4 Simulation Results

Many experiments have been performed with new n-robot CA and DA system. The same openGL-based simulator as in case of two robots is used for monitoring the motion, but the number of the robots are increased in the workspace.

Two experiments are presented in this work with different number of robots in different positions to prove the successfulness of the proposed system. Each experiment is divided

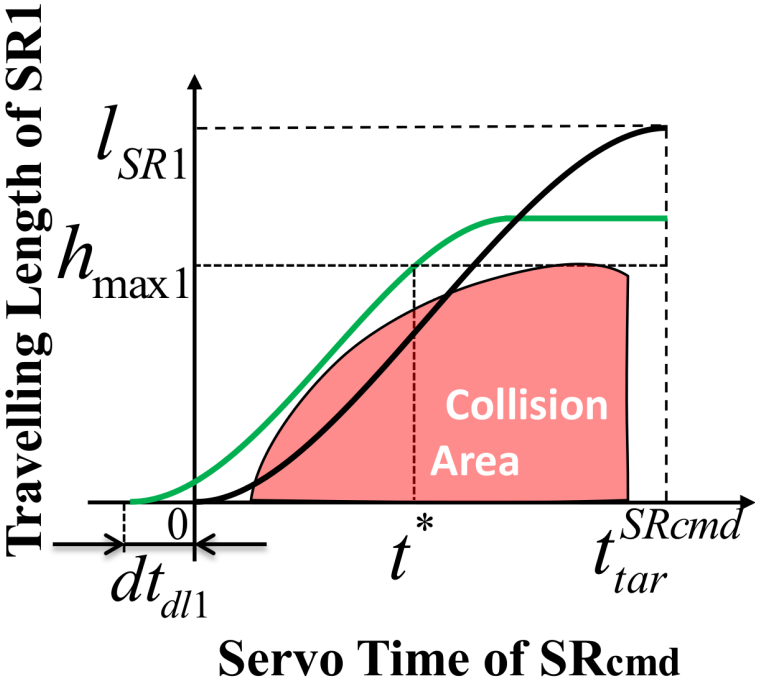


(a) Step 1

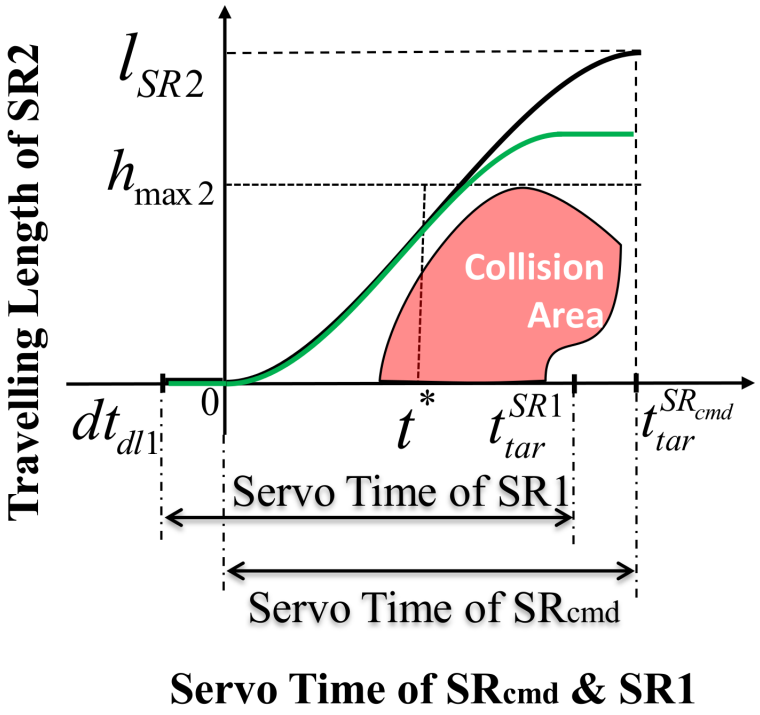


(b) Step 2

Fig. 6.10 Description of essential steps of n-robots deadlock avoidance concept using 2D-space



(a) Advanced collision map of SR_{cmd} and SR1



(b) Advanced collision map of SR_{cmd} and SR2

Fig. 6.11 Illustrative example of advanced collision map of deadlock avoidance in case of three robots

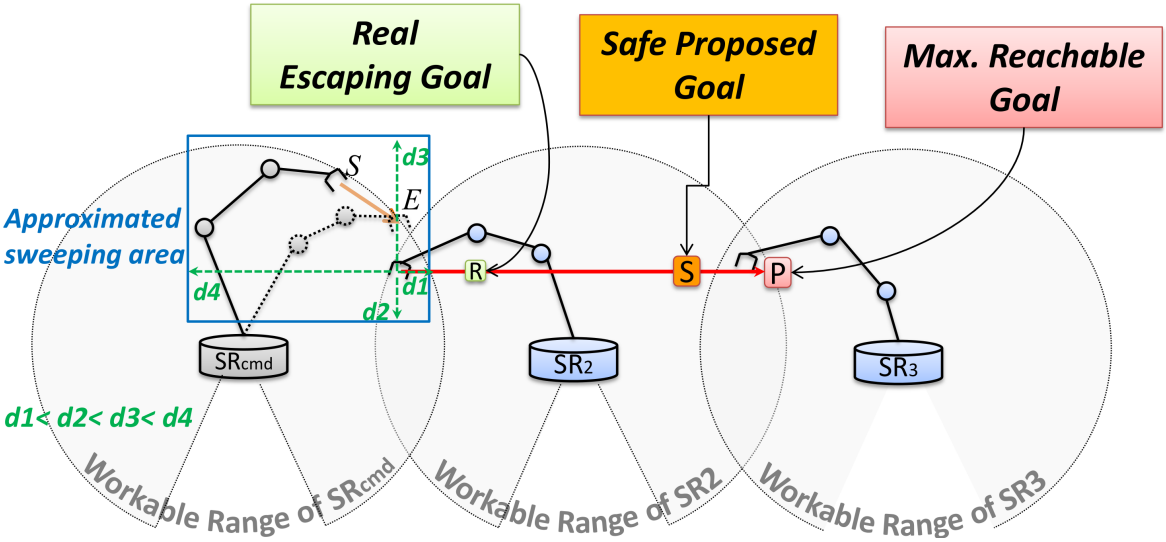


Fig. 6.12 Improved n-robot DA concept: This is an example of three robots in sequence

into two parts. The first part is performed without n-robot CA and DA algorithm, so the commands are sent directly to the robot controller, after which the motion is monitored. The second part is performed with CA and DA algorithm.

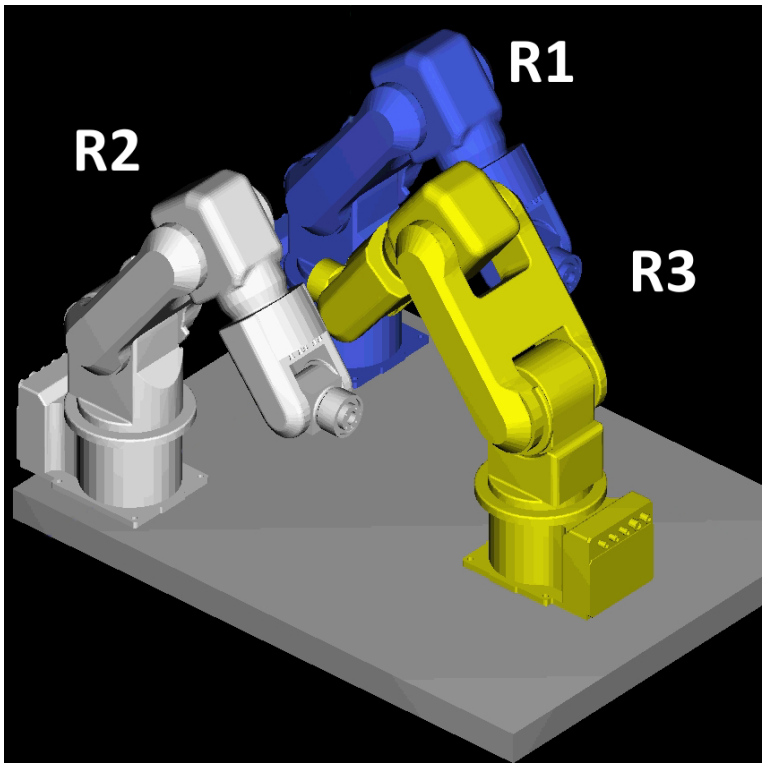


Fig. 6.13 OpenGL-based simulator including three robot manipulators in shared workspace

Table 6.1 PTP commands that are used in Exp1 for $R1$, $R2$, and $R3$.

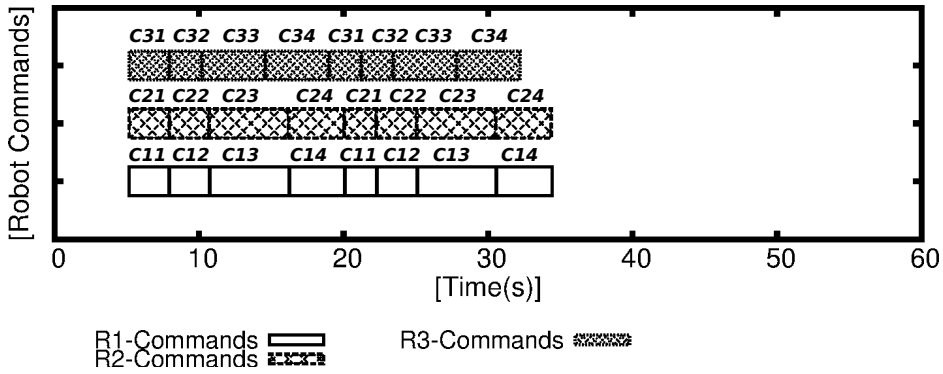
***R*1-Commands** **(x, y, z, roll, pitch, yaw)**	***R*2-Commands** **(x, y, z, roll, pitch, yaw)**	***R*3-Commands** **(x, y, z, roll, pitch, yaw)**
$C11\ (300, 250, 300, 0, 0, 0)$	$C21\ (300, -250, 300, 0, 0, 0)$	$C31\ (300, 0, 300, -180, 0, 0)$
$C12\ (300, 400, 360, 30, 0, 0)$	$C22\ (300, -400, 150, -30, 0, 0)$	$C32\ (200, 0, 200, -180, 0, 0)$
$C13\ (300, 0, 150, 0, 0, 0)$	$C23\ (300, 0, 150, 0, 0, 0)$	$C33\ (450, -300, 200, 90, 0, 0)$
$C14\ (300, 250, 300, 0, 0, 0)$	$C24\ (300, -250, 300, 0, 0, 0)$	$C34\ (300, 0, 300, -180, 0, 0)$

6.4.1 Simulation with Three Robots

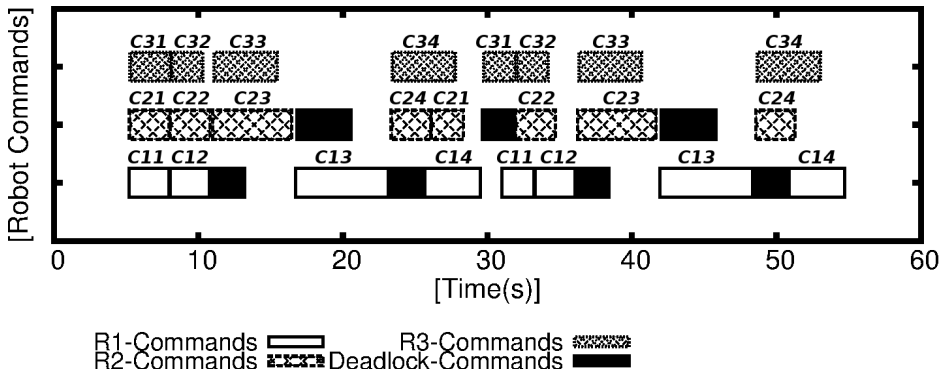
The robot manipulators in this experiment are set in shared workspace. In other words, all workable areas of the robots are overlapping with each others. The snapshot of the simulator including three robots is shown in **Fig. 6.13**. Let us assume that $R1$ is the blue color robot in the rightmost of the simulator figure, $R2$ is the gray robot in the leftmost of the figure, and $R3$ is the yellow robot. The maximum velocity and acceleration of each robot EEF are set as $V1_{max} = V2_{max} = V3_{max} = 100$ mm/s and $a1 = a2 = a3 = 100$ mm/s^2, respectively. All robots have the same length of links which are $l_{base\text{-}joint1} = 290$ mm, $l_{joint1\text{-}joint2} = 260$ mm, $l_{joint2\text{-}joint3} = 270$ mm, and $l_{joint3\text{-}EEF} = 90$ mm. The radii of the spheres which sweep on the line primitives that model the robot are $r_{base} = 112$ mm, $r_{link1} = 117$ mm, $r_{link2} = 100$ mm, and $r_{EEF} = 48$ mm. The global coordinate system is set to be in mid distance between the bases of $R1$ and $R2$, where x-axis is directed to $R3$ and y-axix is directed to $R1$. The PTP commands which are going to be used in this experiments are illustrated in **Table 6.1**. Each command includes information of EEF posture according to the global coordinate system $[x$ [mm], y [mm], z [mm], roll [deg], pitch [deg], yaw [deg]$]$. The initial posture of each robot is assumed to be: $R1_{init}\ (450, 250, 300, 0, 0, 0)$, $R2_{init}\ (450, -250, 300, 0, 0, 0)$, and $R3_{init}\ (150, 0, 300, -180, 0, 0)$.

In the first part of the experiment (Exp.1.1), the commands are sent directly to the controller double times. The execution timing of the commands from the startup of the system is illustrated in **Fig. 6.14(a)**. It is worth mentioning that the period before the execution of the first command is the time required to initialize the system.

The second part of the experiment (Exp.1.2) are done after applying n-robot CA and DA algorithm. The execution timing of the commands after applying the collision avoidance method is shown in **Fig. 6.14(b)**. The black solid boxes in command chart refer to deadlock avoidance commands which are added to original PTP commands. The snapshots of robots' motion are taken to evaluate the system visually. That are taken in both parts of experiments

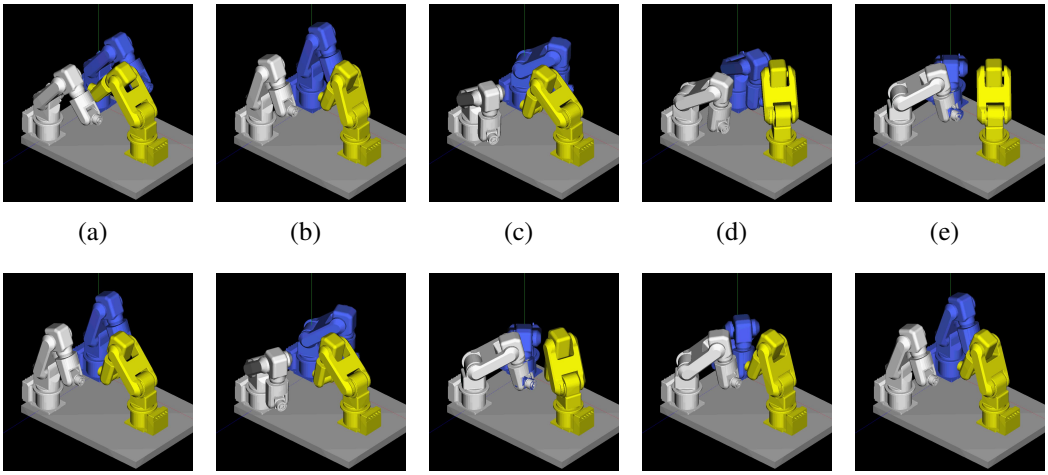


(a) Exp1.1: Command chart before applying n-robot CA and DA algorithm.



(b) Exp1.2: Command chart after applying n-robot CA and DA algorithm.

Fig. 6.14 Command execution timing of $R1$, $R2$, and $R3$ in Exp.1.



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Fig. 6.15 Snapshots for robots' motion in Exp. 1.1 including three robots.

as shown in **Fig. 6.15** and **Fig. 6.16** respectively. It is clear that all robot manipulators

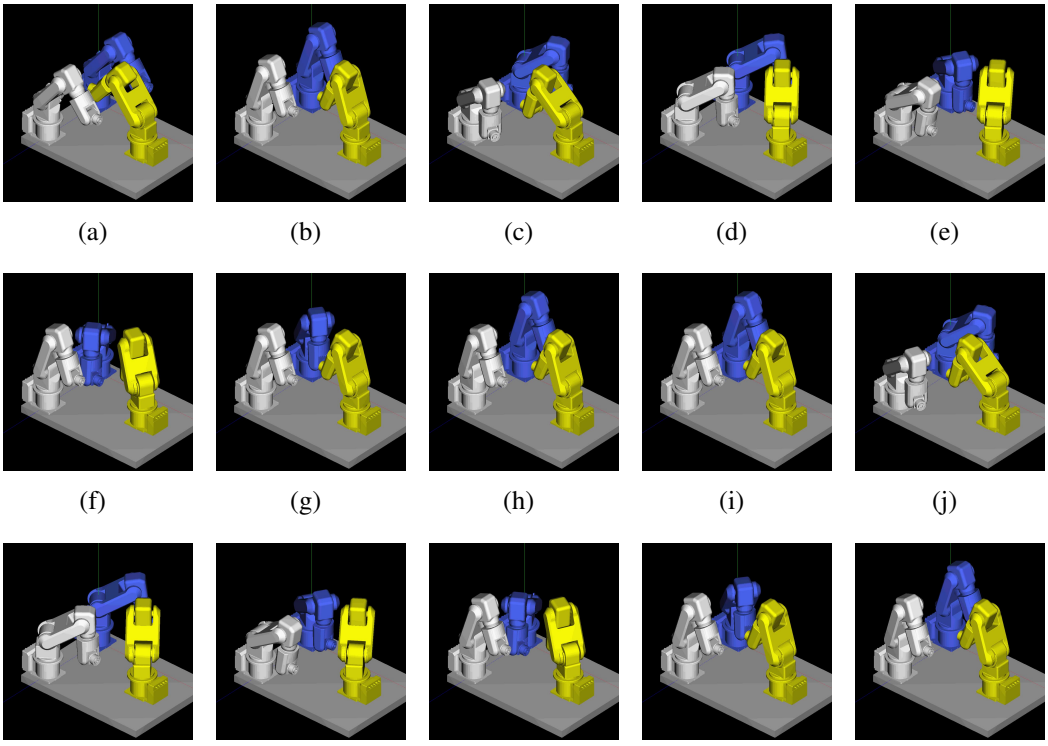


Fig. 6.16 Snapshots for robots' motion in Exp. 1.2 including three robots.

are completing the commands with no collision or deadlock. That means the system is successfully working.

6.4.2 Simulation with Four Robots

The robot manipulators in this experiment are set at sequence positions; meant overlapped workspace. So that the workable area of a robot is overlapped with some other robots in the workspace. The snapshot of the simulator including four robots is shown in **Fig. 6.17**. Let us assume that $R1$ is the Blue color robot on the rightmost of the simulator figure, $R2$ is the gray robot, $R3$ is the yellow robot, and $R4$ is the dark red robot in the left most of the figure. The global coordinate system is located at mid distance between the bases of $R2$ and $R3$. The maximum velocity and acceleration of each robot EEF are set as $V1_{max} = V2_{max} = V3_{max} = V4_{max} = 100$ mm/s and $a1 = a2 = a3 = a4 = 100$ mm/s^2, respectively. All robots have the same length of links which are same as described in previous experiment. The radii of the spheres which sweep on the line primitives that model the robot are also same as previous experiment. In this experiment, we are going to use PTP commands which are illustrated in **Table 6.2**. The initial posture of each robot is assumed to be:

Table 6.2 PTP commands that are used in Exp2 for $R1$, $R2$, $R3$, and $R4$.

***R*1-Commands** **(*x*, *y*, *z*, roll, pitch, yaw)**	***R*2-Commands** **(*x*, *y*, *z*, roll, pitch, yaw)**	***R*3-Commands** **(*x*, *y*, *z*, roll, pitch, yaw)**	***R*4-Commands** **(*x*, *y*, *z*, roll, pitch, yaw)**
$C11\ (300,750,300,0,0,0)$	$C21\ (300,250,300,0,0,0)$	$C31\ (300,-250,300,0,0,0)$	$C41\ (300,-750,300,0,0,0)$
$C12\ (400,1000,150,30,0,0)$	$C22\ (400,0,150,-30,0,0)$	$C32\ (400,0,200,30,0,0)$	$C42\ (400,-1000,200,-30,0,0)$
$C13\ (400,500,150,-30,0,0)$	$C23\ (400,500,150,30,0,0)$	$C33\ (400,-500,200,-30,0,0)$	$C43\ (400,-500,200,30,0,0)$
$C14\ (400,750,300,0,0,0)$	$C24\ (400,250,300,0,0,0)$	$C34\ (400,-250,300,0,0,0)$	$C44\ (400,-750,300,0,0,0)$

$R1_{init}\ (450,750,300,0,0,0)$, $R2_{init}\ (450,250,300,0,0,0)$, $R3_{init}\ (450,-250,300,0,0,0)$, and $R4_{init}\ (450,-750,300,0,0,0)$.

In the first part of the experiment (Exp.2.1), the commands are sent to the controller double directly, i.e there is no CA and DA system. The execution timings of the commands from the start up of the system is illustrated in **Fig. 6.18(a)**. It is worth mentioning that the period before the execution of the first command is the time required to initialize the system. The snapshot of this part of the experiment is shown in **Fig. 6.20**.

The second part of the experiment (Exp.2.2) is done after applying n-robot CA and DA algorithm. The execution timing of the commands after applying the collision avoidance method is shown in **Fig. 6.18(b)**. The snapshots of robots' motion are taken for this part of the experiment as well which is shown in **Fig. 6.21**. It is clear that all robots are moving without any collisions or deadlocks, thus the system schedules the commands perfectly using

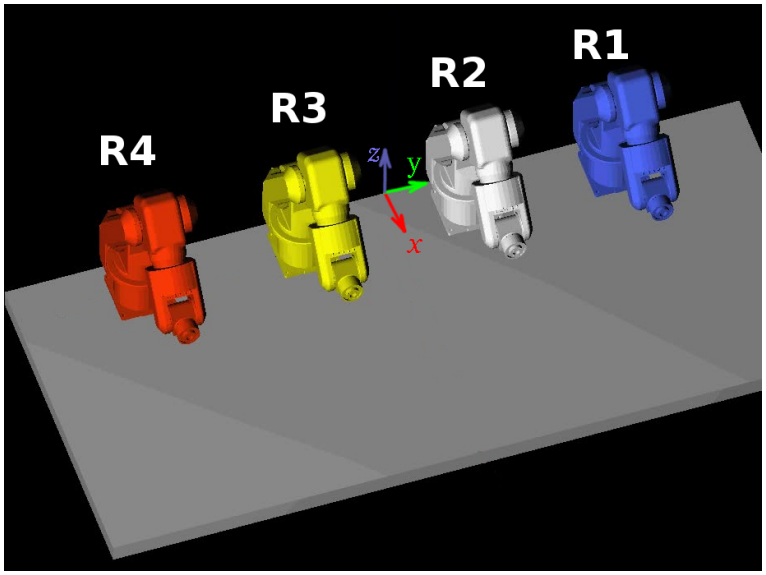
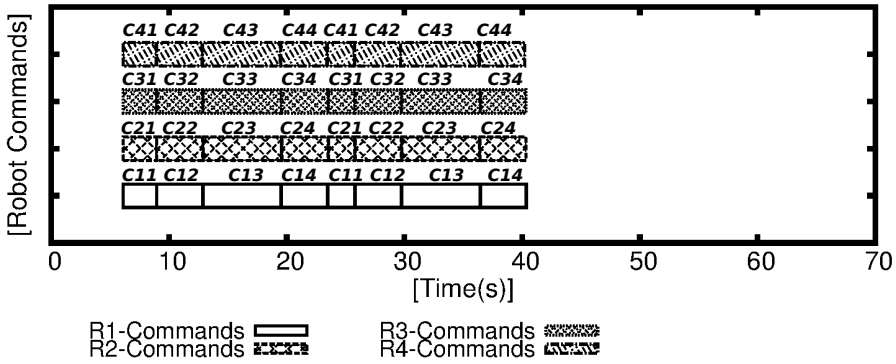
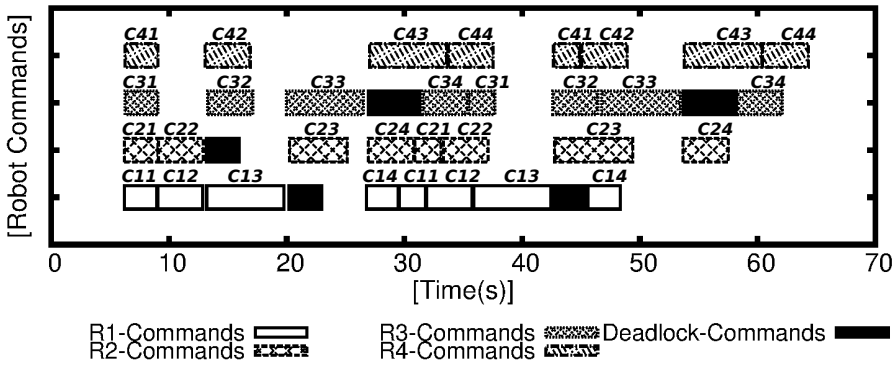


Fig. 6.17 OpenGL-based simulator including four robot manipulators in overlapped workspace



(a) Exp2.1: Command chart before applying n-robot CA and DA algorithm.



(b) Exp2.2: Command chart after applying n-robot CA and DA algorithm.

Fig. 6.18 Command execution timing of $R1$, $R2$, $R3$, and $R4$ in Exp.2.

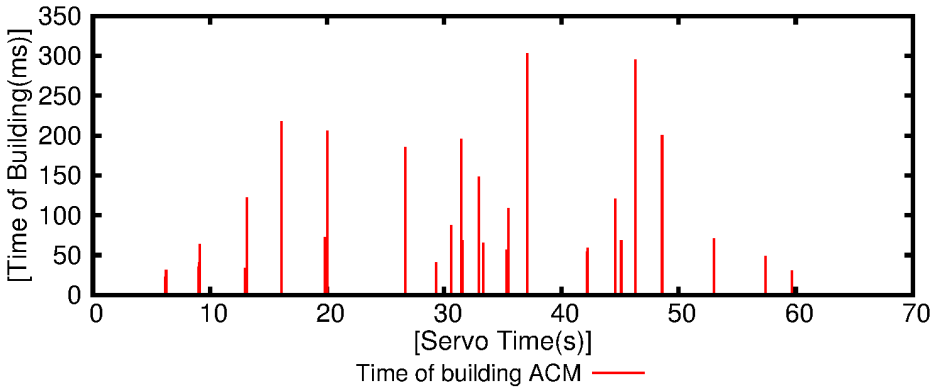
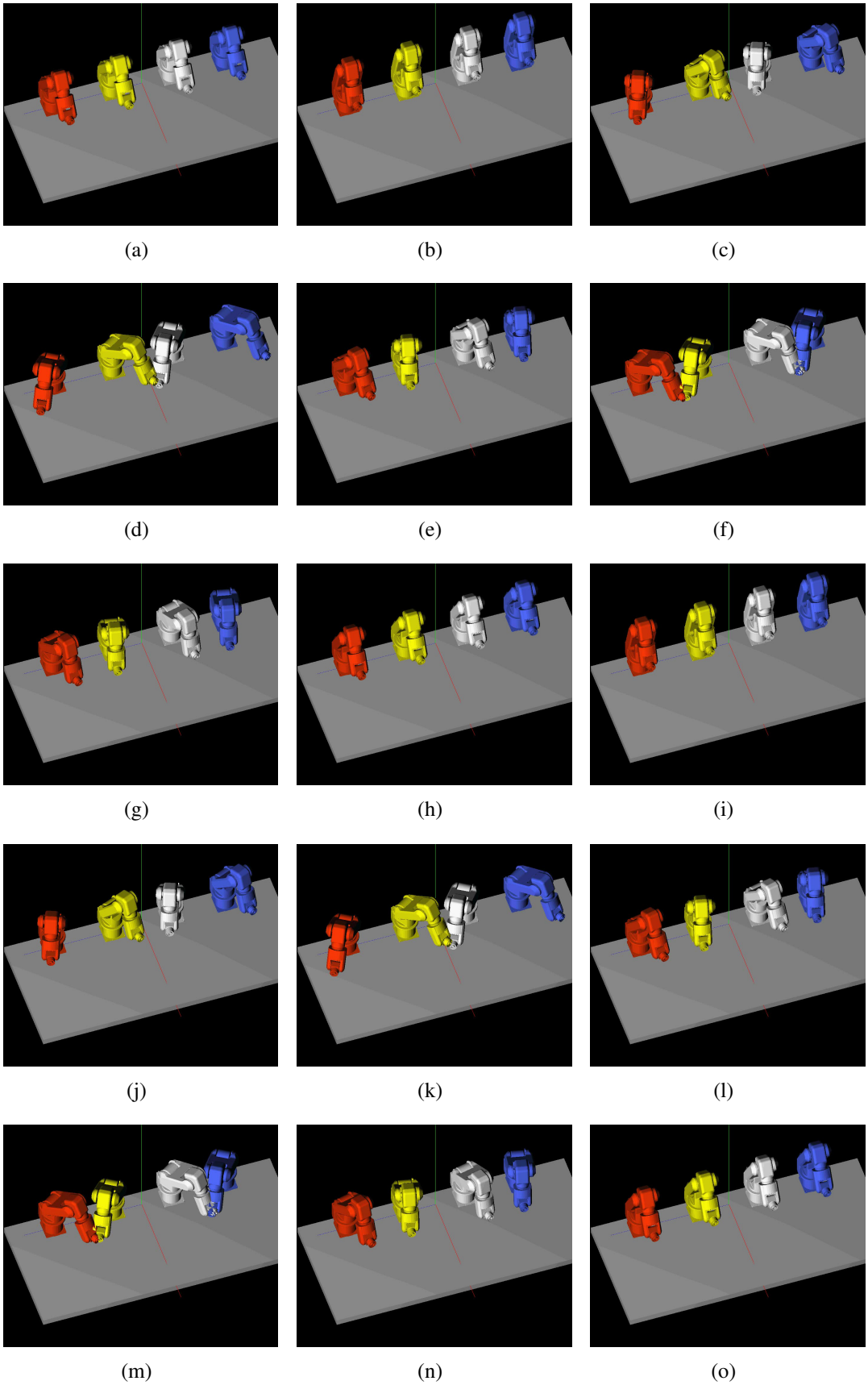


Fig. 6.19 Time of building advanced collision map in Exp.2.2

proposed algorithm. Because the number of the robots has increased, the time of building ACM has been memorized at each creation of the map to show the incremental of building time comparing with 2-robot. The result of ACM building time is shown in **Fig. 6.19**. The average time of building the map in this case is 100ms.



(a) (b) (c)

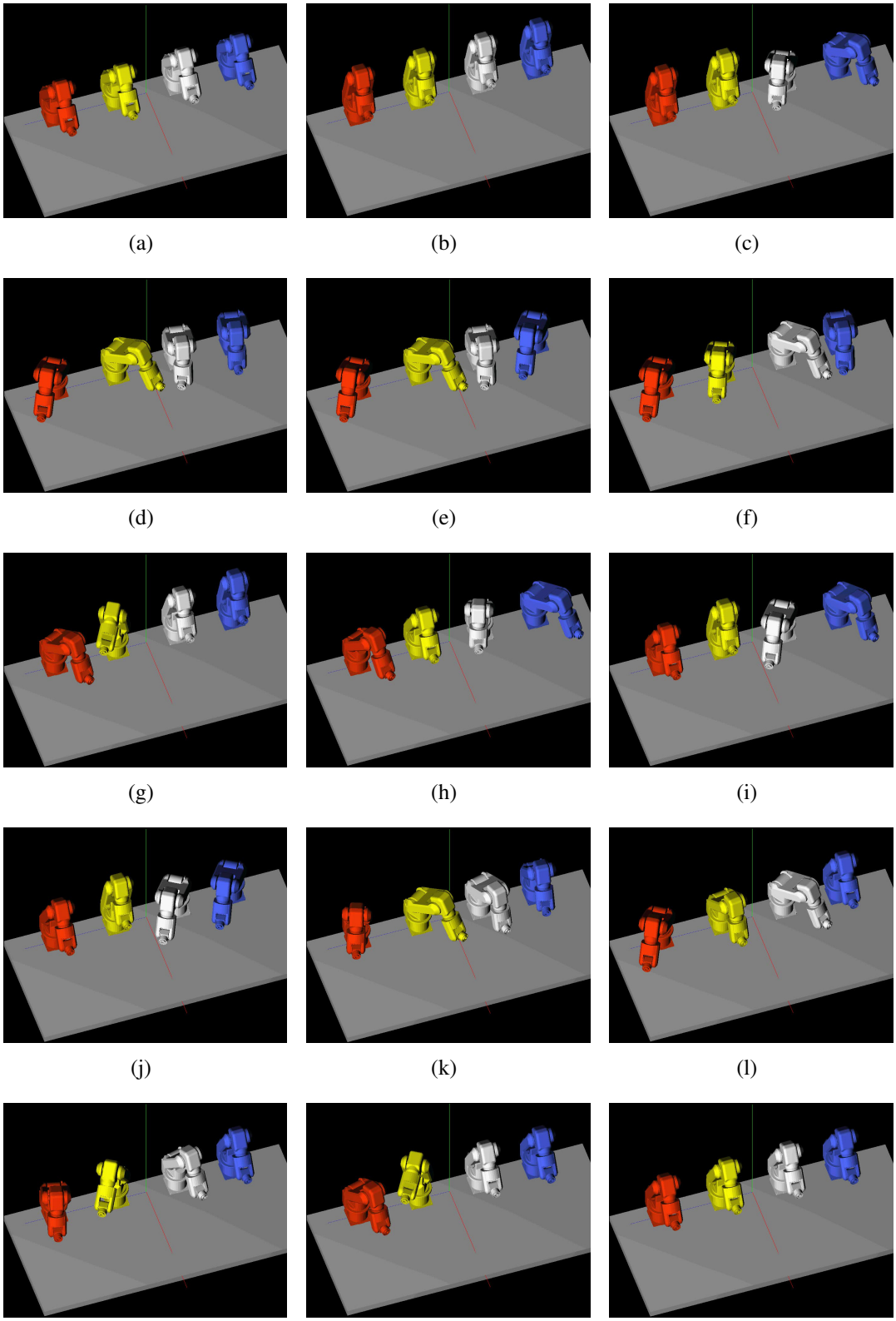
(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

(m) (n) (o)

Fig. 6.20 Snapshots for robots' motion in Exp. 2.1 including four robot manipulators.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

(m) (n) (o)

Fig. 6.21 Snapshots for robots' motion in Exp. 2.2 including four robot manipulators.

6.5 Discussion

The aforementioned simulations have demonstrated successful results to avoid any collisions or deadlocks. The proposed method of n-robot CA and DA system has proved that the system is able to detect and avoid the collisions and deadlocks between multiple number of robots. This advantage gives superiority to proposed system comparing with previous on-line CA and DA systems such as Lee's system [10] and Zhou's system [1] [56], which are limited to address the problem of CA and DA for only two robot manipulators.

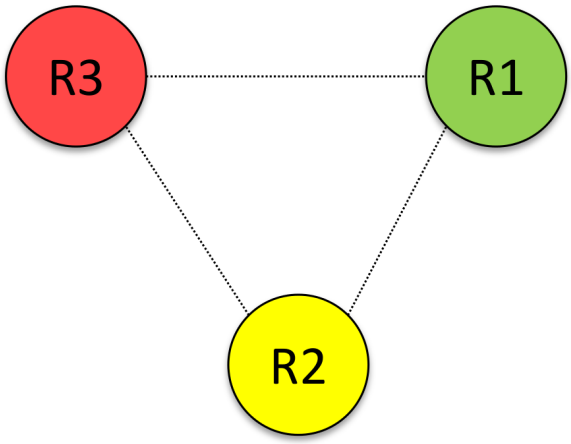
The evaluation of collision detection algorithm has showed that ACM is feasible method regarding proposed system conditions and assumptions. ACM can be applied for any kind and any number of robots because it deals with the robots in the workspace as moving segments which are modelled using SSV modelling. However, the time of building ACM, which is dt_m, registered incremental values by including more robots in workspace. The observation of the dt_m has showed that increasing the number of robot manipulators in the workspace are affected on dt_m linearly, i.e. the mapping time is doubled by increasing one robot manipulator in the workspace. That is true based on Equation 3.10 which has been described in Section 3.3. Consequently, to determine the suitability of proposed system in industry, we did many experiments for different patterns of robots' positions using a PC with normal power whose specification is *Intel Core i5, 2.5GHz* with *memory: 4Gb*. The robots' positions are chosen to be same as industrial applications. Thus, many patterns are obtained. The pattern for 3-robot are as follows:

1. Triangle pattern as shown in **Fig. 6.22(a)**
2. Serial pattern as shown in **Fig. 6.22(b)**

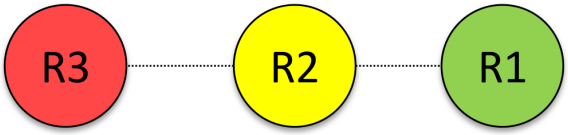
And the patterns for 4-robot are:

1. Zigzag pattern as shown in **Fig. 6.23(a)**
2. Square pattern as shown in **Fig. 6.23(b)**
3. Serial pattern as shown in **Fig. 6.23(c)**

Three experiments of each pattern are presented to evaluate the suitability. In each experiment, many ACMs have been created during operating the robots to avoid the collisions and deadlocks. The time of building each ACM, which is dt_m, is calculated, and the average of those times are taken as well. The PTP commands which are used for each experiment are different from the other, that is done to evaluate the pattern for different ways of movement. The results of those experiments are illustrated in **Table 6.3**. The resulted values of dt_m are

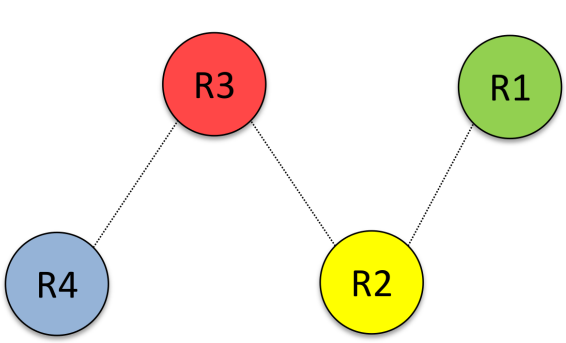


(a) Triangle pattern

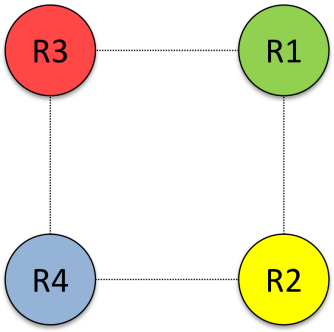


(b) Serial pattern

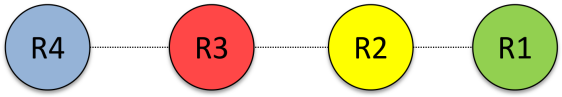
Fig. 6.22 Patterns in case of three robots



(a) Zigzag pattern pattern



(b) Sequence pattern



(c) Serial pattern

Fig. 6.23 Patterns in case of four robots

acceptable to be used in industry according to application type. But if the application needs very fast calculation, it is possible to decrease those values by increasing the PC power. On the other hand, the presented experiments showing the successful results which have no

Table 6.3 Experiments' results of each pattern showing time of building ACMs

Number of Robots	**Pattern Type**	**Exp. No.**	dt_m**-Range[ms]**	dt_m**-Average[ms]**
3	Triangle	Exp.1	$6 \sim 153$	56
3	Triangle	Exp.2	$2 \sim 140$	92
3	Triangle	Exp.3	$3 \sim 278$	52
3	Serial	Exp.1	$23 \sim 137$	67
3	Serial	Exp.2	$21 \sim 88$	48
3	Serial	Exp.3	$16 \sim 82$	49
4	Zigzag	Exp.1	$9 \sim 176$	74
4	Zigzag	Exp.2	$7 \sim 305$	104
4	Zigzag	Exp.3	$11 \sim 285$	112
4	Square	Exp.1	$9 \sim 203$	77
4	Square	Exp.2	$11 \sim 207$	118
4	Square	Exp.3	$7 \sim 202$	71
4	Serial	Exp.1	$22 \sim 303$	100
4	Serial	Exp.2	$23 \sim 111$	67
4	Serial	Exp.3	$17 \sim 164$	65

deadlock warning situation. This situation takes place if the deadlock is unavoidable, and so, the algorithm will ask for human factor help as shown in **Fig. 5.6**.

Time scheduling method which is applied to avoid collision areas on ACM is valid with n-robot in the workspace. That is because the ACM scheme, which relates the travelling length with servo time, is fixed with any number of robots. Therefore we can evaluate that time scheduling method is very suitable to be used with proposed system.

The evaluation of DA system shows ability to avoid the deadlocks even with short distances between the robots' positions. Proposing only six directions for escaping is feasible to be applied with on-line system because applying general methods of motion planning, i.e. C-space [57] and potential field [26], will fall into very burdensome and long planning time that is not desirable in on-line systems. The intelligent use of ACM in DA process could provide the system with suitable escaping goals without meaningless travelling. In addition, the improvement which is described in Section 6.3.3 could exploit the escaping directions more effectively. However, the local minima issue which means inability of the algorithm to find a safe escaping goal has happened with some simulations according to the patterns which have been described before. That issue is common with on-line motion planning

Table 6.4 Evaluation of each pattern based on local minima

Number of Robots	**Pattern Type**	**Local Minima Quantity**
3	Triangle	few
3	Serial	rare
4	Zigzag	High
4	Square	Normal
4	Serial	rare

methods but not desirable for use in industry, and thus, the pattern with low number of local minima is desirable. consequently, after performing many experiments with aforementioned patterns we get results as shown in **Table 6.4**. Those results could give an insight that the proposed DA system is very suitable to be used with serial pattern either with 3, 4, or more robots. The triangle pattern is still suitable with some applications. But the zigzag and square patterns are not suitable to be used in industry and need to consider them in the future. In proposed DA system, the local minima could be decreased if the number of proposed escaping directions are increased. But conversely, that incremental will affect the time needed to calculate escaping positions.

The proposed system is designed regardless the priority of PTP commands to achieve the cooperative tasks between the robots. That is because the overall assumption of the system is the independence of robots' tasks from each other. Even though, the proposed system could be used with the application that needs task priority if some modifications are performed on overall algorithm.

Finally, we can conclude that the proposed system of on-line CA and DA is successful and suitable to be used in industry with many robot manipulators that are controlled using independent controllers with PTP commands. Especially, if the robot number in the workspace is two, three, or four-with serial pattern. We can say that the goal of this research has been accomplished.

Chapter 7

Conclusions and Future Works

7.1 Summary of Dissertation

We have proposed a novel concept for on-line avoidance of collisions and deadlocks between n-robot industrial manipulators. The robots are controlled using PTP commands, and have no prior knowledge of commands that are going to be sent. Therefore, the advanced collision map has been created for detecting potential collisions between the whole robot bodies. Furthermore, the map has been used skilfully to avoid the deadlocks which may happen if the robots become obstacles to each other. To decrease the computational cost of the collision detection, the robot links have been approximated geometrically using SSV with line primitives which is tight modelling for a near-tube robot shape. Owing to restricted conditions of overall system i.e. black box characteristic of the controllers and using PTP commands for controlling, it was successful to apply time scheduling method for the execution time of PTP commands to avoid the collision areas on advanced collision map. The simulations have demonstrated a successful avoidance of the collisions and deadlocks as well as an advantage in time efficiency comparing to previous methods.

In Chapter 2, the overall on-line collision and deadlock avoidance system has been described. The conditions and assumptions have been presented to determine the main framework. In addition, the essential strategies have been put to be used in overall design of the algorithm. Those strategies are: Command Acquisition, Robot Modes, and Robot Modelling.

In Chapter 3, the advanced collision map (ACM) concept which is the main contribution of this work has been presented before explaining the design of the main algorithm. Basically, ACM was created from collision map concept which was developed to detect the collisions between only EEFs of two robot manipulators. ACM has been developed at first to detect

the collision between whole bodies of two robot manipulators. Then, the concept has been generalized to suit the case of n-robot manipulators in the workspace.

In Chapter 4, the algorithm of on-line collision avoidance between two robot manipulators has been explained. This step was very important to be performed before building the generalized system of n-robot collision avoidance system. The algorithm which mainly consists of 4 stages, *Command Acquisition*, *Collision Detection*, *Command Scheduling*, and *Command Execution*, have been explained. The system has been tested and evaluated using openGL-based simulator. In addition, the proposed system has been compared with previous method. That proved the effectiveness of proposed system.

In Chapter 5, the deadlock situation in which both robots become obstacles for each other has been addressed. The method is based on escaping one of the robots to safer position to allow another robot to complete its command. The ACM has been used intelligently in deadlock avoidance process to detect the collisions in addition to decrease meaningless travelling of the robot.

In Chapter 6, the on-line collision and deadlock avoidance system, which has been addressed for two robots in previous chapters, has been improved to include n-robot manipulators in the workspace. That was very important to be performed in order to make the generalized system more feasible for industrial use where many robots are operating the workspace. The necessary modifications on the techniques have been described. Then, the system has been tested with several number of robots in different situations to prove the success of the proposed system.

7.2 Recommended Future Works

The proposed system has been designed to be useful and feasible for industrial use. Therefore, some points for consideration as future works are recommended. Those points are:

- The proposed system has been tested using openGL-based simulators, therefore, it is recommended to check the system with real robot manipulators. Since the proposed system depends completely on time space to avoid the collisions and deadlocks, thus, the time factor is very important. Therefore, with real robot manipulators, the time of correspondence between the PC and robots' controllers should be considered strongly. That because there is usually a bit of delay of correspondence to acquire and send information from and to the robot manipulators.

- One of the assumptions which have been put is the independency; meant that the tasks of all robot manipulators are performed without any priorities. This assumption

is useful and feasible with many industrial applications where the robots' tasks are independent. However, some applications require to have tasks' priority to complete the work. Therefore, to allow the proposed system to have wider use in different kind of applications, it is recommended to include the concept of task priority in the proposed algorithm.

- By increasing the number of the robots in the shared workspace, the overlapped workable areas of the robot are increased. However, the deadlock avoidance becomes very complicated to find solution, and thus, it may fall into the problem of local minima where the deadlock avoidance algorithm cannot find solution. This situations are normal with any methods of motion planning. Therefore, it is recommended to improve the proposed method of deadlock avoidance to exploit the directions of escaping more efficiently. So, the local minima problem could be decreased as much as possible.

Appendix A

Calculating Minimum Distance between Two Line Segments

Introduction

Minimum distance between lines or line segments are necessary to be obtained in some applications. In this work, the robot manipulator bodies have been modelled geometrically using swept sphere volume (SSV) modelling with line primitives. Although some previous works have discussed the methods of calculating distance between SSV primitives e.g. [59] [60] but they are complicated. The complexity is not required in this work. That because the proposed modelling uses only line primitive which can be considered as line segment. Therefore, in order to make collision detection it is necessary to calculate the minimum distance between the robots' links, i.e. the line segments which modelled those links.

In this appendix, the steps of finding minimum distance between two line segments are explained. The steps are adopted from [69].

Distance between Lines

To understand how to calculate the minimum distance between two line segments, it is important to know how to calculate the minimum distance between lines in 3D space.

Consider two lines in 3D-space as shown in **Fig. A.1** as follows:

$L1 : P(\lambda) = P_0 + \lambda(P_1 - P_0) = P_0 + \lambda u$

$L2 : Q(\xi) = Q_0 + \xi(Q_1 - Q_0) = Q_0 + \xi v$

Let $W(\lambda, \xi) = P(\lambda) - Q(\xi)$ be a vector between points on the two lines. The purpose is to find the $W(\lambda, \xi)$ that has a minimum length for all λ and ξ. In any 3D-space, the two lines

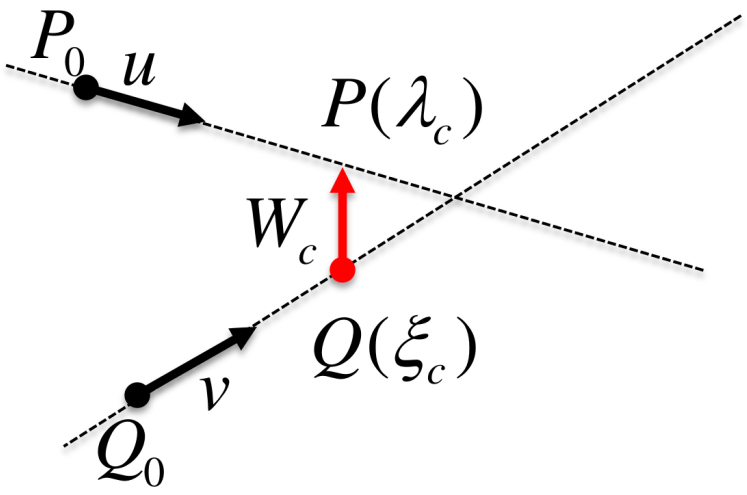


Fig. A.1 Two lines P and Q in 3D-space

$L1$ and $L2$ are closest to each other with a line segment that is perpendicular to both of them. In other words, the lines are closest at unique points $P_C = P(\lambda_C)$ and $Q_C = Q(\xi_C)$ for which $W(\lambda_C, \xi_C)$ is the unique minimum for $W(\lambda, \xi)$. Moreover, if L1 and L2 are not parallel and do not intersect each other, then the segment $P_C Q_C$ is simultaneously perpendicular to both lines. That is, the vector $w_C = W(\lambda_C, \xi_C)$ is uniquely perpendicular to the line direction vectors u and v, and this is equivalent to it satisfying the two equations:

$$u \cdot W_C = 0 \tag{A.1a}$$

$$v \cdot W_C = 0 \tag{A.1b}$$

The equations can be solves by substituting $W_C = P(\lambda_C) - Q(\xi_C) = W_0 + \lambda_C u - \xi_C v$, where $W_0 = P_0 - Q_0$, into each one to get two simultaneous linear equations:

$$(u \cdot u)\lambda_C - (u \cdot v)\xi_C = -u \cdot W_0 \tag{A.2a}$$

$$(v \cdot u)\lambda_C - (v \cdot v)\xi_C = -v \cdot W_0 \tag{A.2b}$$

Then, putting $a = u \cdot u$, $b = u \cdot v$, $c = v \cdot v$, $d = u \cdot W_0$, and $e = v \cdot W_0$, we solve for λ_C and ξ_C as:

$$\lambda_C = \frac{be - cd}{ac - b^2} \tag{A.3a}$$

$$\xi_C = \frac{ae - bd}{ac - b^2} \tag{A.3b}$$

whenever the denominator $ac - b^2$ is non-zero.

When ac–b2 = 0, the two lines are parallel, and the distance between the lines is constant. The solution is by fixing the value of one parameter and using either equation to solve for the other. Selecting $\lambda_C = 0$, we get $\xi_C = d/b = e/c$.

Having solved for λ_C and ξ_C, we have the points P_C and Q_C on the two lines L1 and L2 where they are closest to each other. Then the distance between them can be calculated as follows:

$$d(L1, L2) = |P(\lambda_C) - Q(\xi_C)| \tag{A.4}$$

Distance between Line Segments

The distance between segments are not same as the distance between their extended lines. That because the closest points on the extended infinite lines may be outside the range of the segments. Let assume that:

- First segment $Seg1 = [P_0, P_1]$ is represented $P(\lambda) = P_0 + \lambda(P_1 - P_0) = P_0 + \lambda u$ whereas $0 \leq \lambda \leq 1$.
- Second segment $Seg2 = [Q_0, Q_1]$ is represented $Q(\xi) = Q_0 + \xi(Q_1 - Q_0) = Q_0 + \xi v$ whereas $0 \leq \xi \leq 1$.

The steps of calculating minimums distance between two segments are as follows:

1. First step in computing a distance including segments is to get the closest points for the infinite lines that they lie on. So, computing λ_C and ξ_C for L1 and L2.
2. If those points are both in the range of the respective segment, then they give closest points. But if they lie outside the range of either, then they are not and we have to determine new points that minimize $W(\lambda, \xi) = P(\lambda) - Q(\xi)$ over the ranges of interest.
3. Minimizing the length of W , i.e. minimizing $|W^2| = W \cdot W = (W_0 + \lambda u - \xi v)(W_0 + \lambda u - \xi v)$. This is quadratic function of λ and ξ.

   In fact, $|W^2|$ defines a parabaloid over the (λ, ξ)-plane with a minimum at $C = (\lambda_C, \xi_C)$, and which is strictly increasing along rays in the (λ, ξ)-plane that start from C and go in any direction. But, when segments are involved, it is necessary to be the minimum over a subregion A of the (λ, ξ)-plane, and the global absolute minimum at C may lie outside of A. However, in these cases, the minimum always occurs on the boundary of A, and in particular, on the part of A's boundary that is visible to C

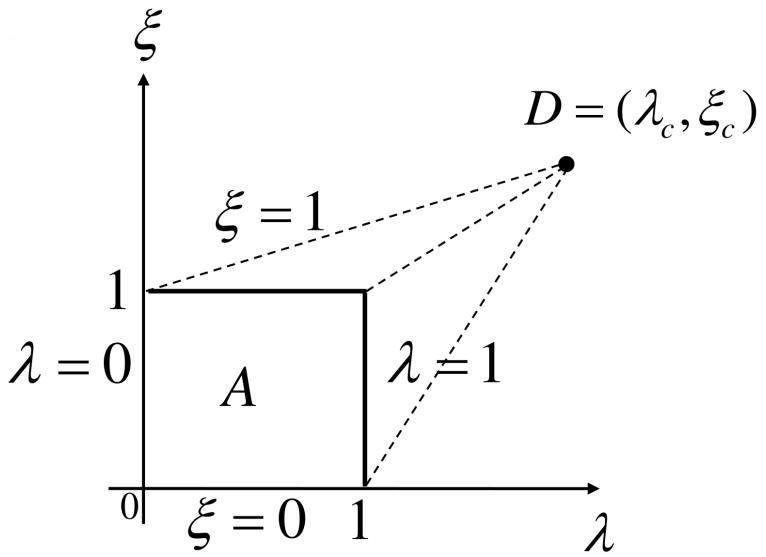


Fig. A.2 Unit square to calculate minimum distance

4. Forming the area $A = \{(\lambda, \xi) | 0 \leq \lambda \leq 1 \text{ and } 0 \leq \xi \leq 1 = [0,1] \times [0,1]\}$ is the unit square as shown in **Fig. A.2**. The four edges of the square are given by $\lambda = 0$, $\lambda = 1$, $\xi = 0$, and $\xi = 1$.

5. Determining the candidate edge for a minimum of $|W^2|$ based on the position of C. So, if $C = (\lambda_C, \xi_C)$ is outside A, then it can see at most two edges of A.
   If $\lambda < 0$, C can see the $\lambda = 0$ edge; if $\lambda > 1$, C can see the $\lambda = 1$ edge.
   If $\xi < 0$, C can see the $\xi = 0$ edge; if $\xi > 1$, C can see the $\xi = 1$ edge.
   Clearly, if C is not in A, then at least 1 and at most 2 of these inequalities are true.

6. Calculating the position of minimum occurrence on each candidate edge, either in its interior or at an endpoint. Consider the edge $\lambda = 0$, along which $|W^2| = (W_0 - \xi v)(W_0 - \xi v)$. Taking the derivative with ξ we get a minimum when:

$$0 = \frac{d}{dt}|W^2| = -2v \cdot (W_0 - \xi v) \tag{A.5}$$

   which gives a minimum on the edge at $(0, \xi_0)$ where:

$$\xi_0 = \frac{v \cdot W_0}{v \cdot v} \tag{A.6}$$

   If $0 \leq \xi \leq 1$, then this will be the minimum of $|W^2|$on A, and $P(0)$ and $Q(\xi_0)$ are the two closest points of the two segments. However, if ξ_0 is outside the edge, then an endpoint of the edge, $(0,0)$ or $(0,1)$, is the minimum along that edge; and further, we

will have to check a second visible edge in case the true absolute minimum is on it. The other edges are treated in a similar manner.

Code of Calculating Minimum Distance

To understand the idea deeply, here is the code which is written using C++ programming language. The code is showing the function $seg2seg_{mindis}(seg1, seg2)$ which calculate the minimum distance between two line segments using aforementioned method. The following functions are needed for some arithmetic calculation of two vectors: The $dot(a,b)$ calculates the dot product of two vectors, $minus(a,b)$ calculates the result of subtraction of two vectors, and $plus(a,b)$ calculates the result of addition of two vectors.

The code is as follows:

```
//==========Structure definition==========\
struct xyz{
        double x,y,z;
};

//==========Calculating the minimum distance==========\
double seg2seg_mindis (xyz seg1_str,xyz seg1_tar,
                       xyz seg2_str, xyz seg2_tar){

//>>Local Varibales
 double a,b,c,d,e;
 double Sc, Tc; \\ Where, Sc means lambda & Tc is xi
 double dmin;

//>>Definitions
 P0.x = seg1_str.x;
 P0.y = seg1_str.y;
 P0.z = seg1_str.z;
 P1.x = seg1_tar.x;
 P1.y = seg1_tar.y;
 P1.z = seg1_tar.z;

 Q0.x = seg2_str.x;
```

```
 Q0.y = seg2_str.y;
 Q0.z = seg2_str.z;
 Q1.x = seg2_tar.x;
 Q1.y = seg2_tar.y;
 Q1.z = seg2_tar.z;

//>>Calculating the statistics of each segment
 minus(P1,P0,&u);
 minus(Q1,Q0,&v);
 minus(P0,Q0,&W0);
 a = dot(u,u);
 b = dot(u,v);
 c = dot(v,v);
 d = dot(u,W0);
 e = dot(v,W0);

//>>Determine if the two line intersected or parallel
 if((a*c-pow(b,2)) == 0){
   //fixing one value and calculate the other
   Sc = 0;
   if(b!=0) Tc = d/b;else if(c!=0)Tc = e/c;
 }else{
   Sc = (b*e - c*d)/(a*c-pow(b,2));
   Tc = (a*e - b*d)/(a*c-pow(b,2));
 }

//>>Determine the area of Sc && Tc and modify it if needed
 if(Sc>= 0 && Sc <= 1 && Tc>=0 && Tc <=1){
   printf("The minimum distance vector is inside segments");
 }else{
   if(Tc>1){
    Tc = 1;
    Sc = (b - d)/a;
    if(Sc>1)Sc=1;else if(Sc<0)Sc=0;
  }else if(Tc<0){
    Tc = 0;
```

```
    Sc = -d/a;
    if(Sc>1)Sc=1;else if(Sc<0)Sc=0;
  }else{
    if(Sc<0){
     Sc = 0;
     Tc = e/c;
     if(Tc>1)Tc=1;else if(Tc<0)Tc=0;
   }else if(Sc>1){
     Sc = 1;
          Tc = (e+b)/c;
          if(Tc>1)Tc=1;else if(Tc<0)Tc=0;
   }
  }
 }

//>>Calculating the minimum distance and
//>>its points position on both segments
        Pc.x = P0.x + Sc*u.x;
        Pc.y = P0.y + Sc*u.y;
        Pc.z = P0.z + Sc*u.z;
        Qc.x = Q0.x + Tc* v.x;
        Qc.y = Q0.y + Tc* v.y;
        Qc.z = Q0.z + Tc* v.z;
        minus(Pc,Qc,&Wc);
        dmin = sqrt(pow(Wc.x,2) + pow(Wc.y,2) + pow(Wc.z,2));
        return dmin;
}



//==========dot product between two matrix==========\
double dot(xyz a,xyz b){
        double res = a.x*b.x + a.y*b.y + a.z*b.z;
        return res;
}
//==========addition of two matrix==========\
```

```
void plus(xyz a,xyz b,xyz *res){
        res->x = a.x + b.x;
        res->y = a.y + b.y;
        res->z = a.z + b.z;
}
//==========subtraction of two matrix==========\
void minus(xyz a,xyz b,xyz *res){
        res->x = a.x - b.x;
        res->y = a.y - b.y;
        res->z = a.z - b.z;
}
```

Appendix B

Making Union of Two Ranges

Introduction

The definition of the range is a continuous possible collisions every sampling time dt_s which starts from minimum value min_i and ends at maximum value max_i, where i indicates the range number. It has been explained in the procedures of advanced collision map design that creating many collision timing maps (CTMs) will result to have many ranges of continuous collisions. Thus, to have single CTM, it is necessary to make union of the ranges at each sampling time.

In this appendix, the method and the code of making union of two ranges are going to be explained.

Methodology

To make union of two ranges, it is necessary to understand how many patterns of arrangement could the ranges have. Therefore, let assume that we have two ranges as follows:

- $range_1$ starts from min_1 and ends at max_1
- $range_2$ starts from min_2 and ends at max_2

The patterns of both ranges could be in six total as follows:

1. Pattern 1: When $min_1 \leq min_2$ & $min_2 < max_1 \leq max_2$ as show in **Fig. B.1(a)**.
2. Pattern 2: When $min_2 \leq min_1 \leq max_2$ & $max_1 > max_2$ as shown in **Fig.B.1(b)**.
3. Pattern 3: When $min_1 > min_2$ & $max_1 < max_2$ as shown in **Fig.B.1(c)**.

4. Pattern 4: When $min_1 < min_2$ & $max_1 > max_2$ as shown in **Fig.B.1(d)**.
5. Pattern 6: When $max_1 < min_2$ as shown in **Fig.B.1(e)**.
6. Pattern 5: When $min_1 > max_2$ as shown in **Fig.B.1(f)**.

The results of union of two ranges based on aforementioned patterns are as follows:

1. Pattern 1 will result to have one single range starts from min_1 and ends at max_2.
2. Pattern 2 will result to have one single range starts from min_2 and ends at max_1.
3. Pattern 3 will result to have one single range starts from min_2 and ends at max_2.
4. Pattern 4 will result to have one single range starts from min_1 and ends at max_1
5. Pattern 5 will result to have exactly same ranges that are $range_1$ and $range_2$.
6. Pattern 6 will result to have two ranges but $range_2$ comes before $range_1$, i.e. we get two ranges, first one starts from min_2 and ends with max_2, then second one starts from $min_1 > max_2$ and ends at max_1.

Code of Two-Range Union

The code of getting the result of two ranges union is illustrated. The programming language is C++. In the following is the detail code of function *U2ranges()*, where the inputs are $range_1$ and $range_2$. And the output is variable *res[2]* which is matrix of two ranges. The code is:

```
struct range{
        double min;
        double max;
};
void U2range(range range1_min,range range2,range res[2]){
\\pattern 1
if(range1.min<=range2.min && range1.max <= range2.max){
        res[0].min = range1.min;
        res[0].max = range2.max;
        res[1].min = 0;
        res[1].max = 0;
```

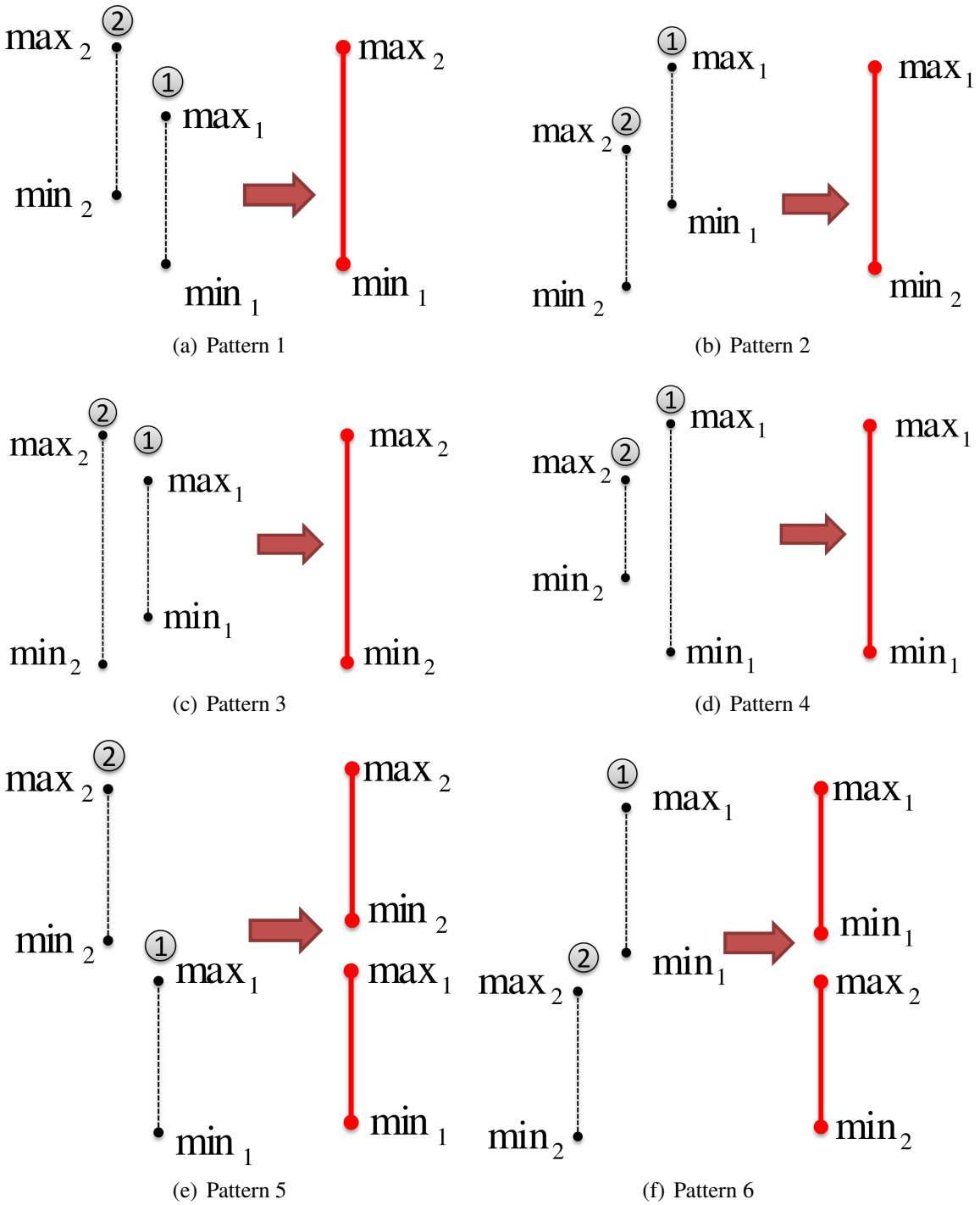


Fig. B.1 Patterns of two ranges' arrangements

```
\\pattern 2
}else if(range1.min>=range2.min && range1.max >=range2.max){
        res[0].min = range2.min;
        res[0].max = range1.max;
        res[1].min = 0;
        res[1].max = 0;
\\pattern 3
else if(range1.min>=range2.min && range1.max<=range2.max){
        res[0].min = range2.min;
        res[0].max = range2.max;
        res[1].min = 0;
        res[1].max = 0;
\\pattern 4
}else if(range1.min<=range2.min && range1.max>=range2.max){
        res[0].min = range1.min;
        res[0].max = range1.max;
        res[1].min = 0;
        res[1].max = 0;
\\pattern 5
}else if(range1.max<range2.min){
        res[0].min = range1.min;
        res[0].max = range1.max;
        res[1].min = range2.min;
        res[1].max = range2.max;
}
\\pattern 6
}else if(range1.min>range2.max){
        res[0].min = range2.min;
        res[0].max = range2.max;
        res[1].min = range1.min;
        res[1].max = range1.max;
}
}
```

Appendix C

Approximating Sweeping Area of Robot Manipulator

Introduction

The sweeping area of robot manipulator as defined in this dissertation is an area which the robot manipulator occupies during motion from initial to target position. However, calculating this kind of area is very burdensome and complicated, and thus, it is necessary to approximate the area as simple as possible. Knowing this area helps to inform other robot manipulators with the danger area that is necessary to be avoided.

In this dissertation, the sweeping area is approximated using cuboid shape. The remain of this appendix is the steps of forming that cuboid.

Main Parameters of Cuboid

The cuboid consists of six quadrilateral faces. Therefore, to form cuboid it is sufficient to calculate the coordinates of two faces only; meant the coordinates of eight points. That can be done using six parameters which are $x_{min}, x_{max}, y_{min}, y_{max}, z_{min}$, and z_{max}. As a result, the coordinates of the points according to those parameters are :

1. Face1: $(x_{min}, y_{min}, z_{min}), (x_{max}, y_{min}, z_{min}), (x_{max}, y_{max}, z_{min}), (x_{min}, y_{max}, z_{min})$.
2. Face2: $(x_{min}, y_{min}, z_{max}), (x_{max}, y_{min}, z_{max}), (x_{max}, y_{max}, z_{max}), (x_{min}, y_{max}, z_{max})$.

The illustrative graph of cuboid parameters is shown in **Fig. C.1**.

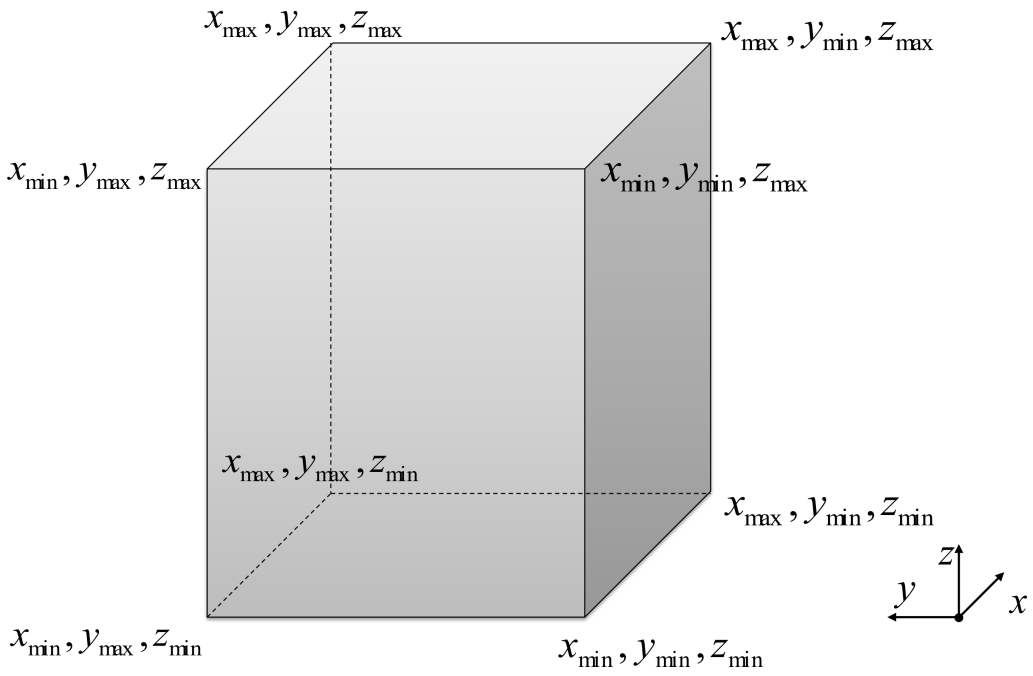


Fig. C.1 Main parameters which can form cuboid in 3D-sapce

Steps of Forming Cuboid

The following steps could be used to form a cuboid of sweeping area for any robot manipulator. Those steps are:

1. Determining the maximum number of robot's joints n_{joint}.
2. Geometric modelling of the robot. The robot is modelled using swept sphere volume (SSV) with line primitives. That means each link is modelled using a line and sphere with specific radius, and thus, each link has thickness which should be considered to form the cuboid.
3. Determining the trajectory of robot manipulator. Regarding path, velocity, and acceleration the trajectory of the robot is determined. Here, every sampling time dt_s the posture of the robot manipulator; meant joints' positions are calculated.
4. Forming initial cuboid, with which the initial posture of the robot manipulator is covered as tight as possible. To simplify the matter, it is going to be explained using 2D space.

Let assume that there is a robot manipulator with two links and $n_{joint} = 3$, the basement joint, link joint, and EEF. Let assume also that:

- link1 is modelled using line l_1 and sphere with radius r_1.
- link2 is modelled using line l_2 and sphere with radius r_2.
- EEF is modelled using line l_3 and spheres with radius r_3.

Here, the position of each joint is acquired:

- $joint_1$ position is (x_1, y_1, z_1).
- $joint_2$ position is (x_2, y_2, z_2).
- $joint_3$ position is (x_3, y_3, z_3).

Then, those positions are used to form the initial cuboid with consideration of links' thickness. It should be noticed that $joint_1$ connects link1 with link2 and $joint_2$ connects link2 with EEF, that means the modelling spheres in those joints are intersected. Thus, to form the cuboid, it is important also to consider that point. Based on previous considerations, the parameters which are necessary to form the cuboid are calculated as follows:

$$x_{min} = Min(x_1 + r_1, x_2 + r_1, x_2 + r_2, x_3 + r_2, x_3 + r_3) \tag{C.1a}$$
$$x_{max} = Max(x_1 + r_1, x_2 + r_1, x_2 + r_2, x_3 + r_2, x_3 + r_3) \tag{C.1b}$$
$$y_{min} = Min(y_1 + r_1, y_2 + r_1, y_2 + r_2, y_3 + r_2, y_3 + r_3) \tag{C.1c}$$
$$y_{min} = Min(y_1 + r_1, y_2 + r_1, y_2 + r_2, y_3 + r_2, y_3 + r_3) \tag{C.1d}$$
$$z_{min} = Min(z_1 + r_1, z_2 + r_1, z_2 + r_2, z_3 + r_2, z_3 + r_3) \tag{C.1e}$$
$$z_{max} = Max(z_1 + r_1, z_2 + r_1, z_2 + r_2, z_3 + r_2, z_3 + r_3) \tag{C.1f}$$

We are going to refer to this cuboid as reference cuboid.

5. Expanding the cuboid. In this step, the cuboid is expanded, if necessary, according to the motion of the robot. At each sampling time t, the positions of all joints are used. Then, Eq. C.1 is applied to calculate the parameters of the cuboid at that time; let call it $cuboid(t)$. Afterwards, those parameters are compared with the same parameter which have formed last cuboid. The comparison provides new parameter values which form a new reference cuboid. This cuboid is going to be compared with $cuboid(t + dt_s)$ and so on until reaching the cuboid at final time.

The final cuboid is the approximated sweeping area which the other robots should avoid. The illustrative graph of forming the cuboid is shown in **Fig. C.2**.

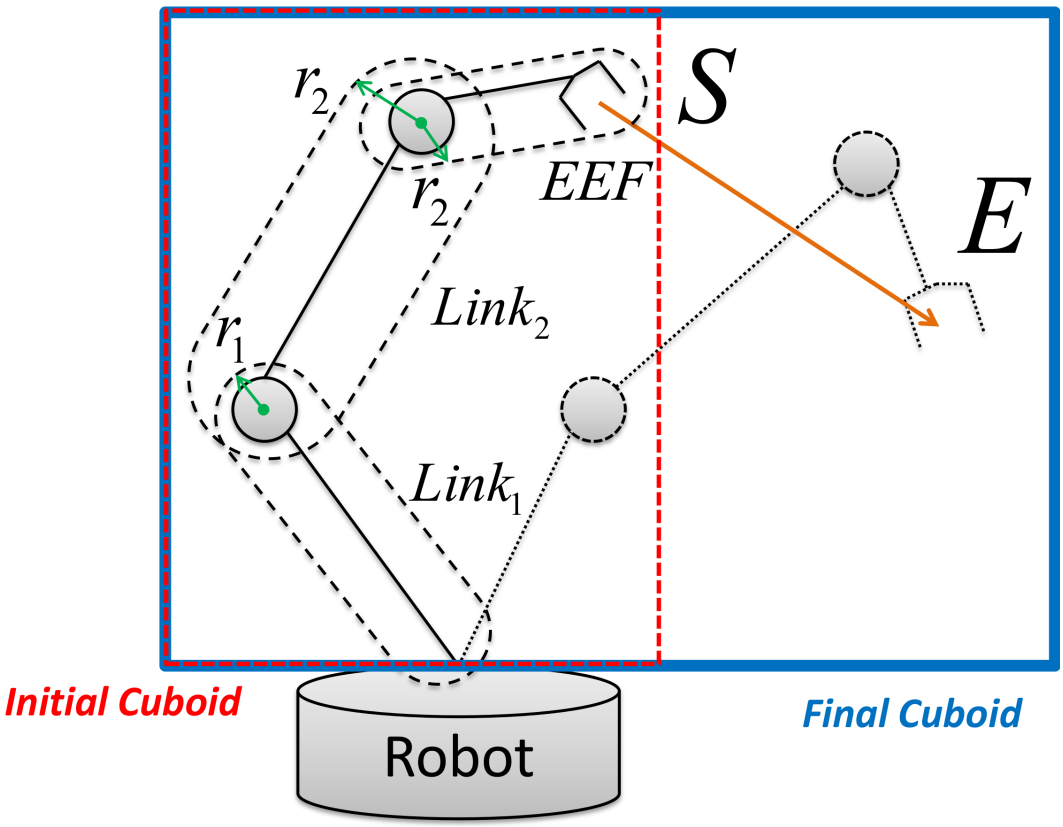


Fig. C.2 Illustrative graph present how to form cuboid (In 2D-sapce)

List of Publications

Journal Papers (Peer Reviewed)

[1] A . Y. Afaghani and Y. Aiyama, “On-line collision avoidance of two command based industrial robotic arms using advanced collision map,” *Int. J. of Robotics and Mechatronics (JRM)*, pp. 321-330, vol.26, no.3, Jun 2014.

Conference Proceedings (Peer Reviewed)

[1] A. Y. Afaghani and Y. Aiyama, “On-line collision avoidance between two robot manipulators using collision map and simple escaping method,” *In Proc. IEEE/SICE Int. Symp. on System Integration (SII)*, pp. 105-110, 15-17 Dec 2013.

[2] A. Y. Afaghani and Y. Aiyama, “Advanced-collision-map-based on-line collision and deadlock avoidance between two robot manipulators with ptp commands,” *In Proc. IEEE Int. Conf. on Automation Science and Engineering (CASE)*, pp. 1244-1251, 18-22 Aug 2014.

Other Publications (Without Peer Review System)

[1] A. Y. Afaghani and Y. Aiyama, “On-line collision-free motion of two command-based industrial manipulators,” *In JSME Conf. on Robotics and Mechatronics (Robomech)*, no. 12-2, pp. 2A1-W03(1-4), 25-29 May 2014.

[2] Y. Aiyama, Z. Jianing, and A. Y. Afaghani, “Collision avoidance for manipulators which have independent controllers,” *In Proc. SICE Symp. on System Integration (SI)*, pp. 537-541, Dec 2013.

References

[1] J. Zhou, K. Nagase, S. Kimura, and Y. Aiyama, "Collision avoidance of two manipulators using rt-middleware," in *IEEE/SICE Int. Symp. on System Integration.*, pp. 1031–1036, 20-22 Dec 2011.

[2] T. Tsubouchi, S. Kuramochi, and S. Arimoto, "Iterated forecast and planning algorithm to steer and drive a mobile robot in the presence of multiple moving objects," in *Proc. IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 33–38, 8-13 May 1995.

[3] M. Jager and B. Nabel, "Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots," in *Proc. IEEE/RSJ Int. Conf on Intelligent Robots and Systems*, vol. 3, pp. 1213–1219, 29 Oct- 03 Nov 2001.

[4] S.-H. Park and B.-H. Lee, "A new analytical representation to robot path generation with collision avoidance through the use of the collision map," *J. of Control, Automation and Systems*, vol. 4, pp. 77–86, February 2006.

[5] K. Sakurama and K. Nakano, "Online modification of reference trajectories for multiple robots with collision avoidance," in *Proc. IEEE Int. Conf. on Decision and Control*, pp. 2412–2416, 13-15 Dec 2006.

[6] K. Sakurama and K. Nakano, "Online modification of reference trajectories of multiple robots for collision avoidance," in *Proc. IEEE Int. Conf. on Decision and Control*, pp. 1416–1422, 12-14 Dec 2007.

[7] K. Sakurama and K. Nakano, "Deadlock-free path-following control for collision avoidance of multiple robots," in *Proc. IEEE Int. Conf on Decision and Control*, pp. 5673–5678, 15-18 Dec 2009.

[8] A. Sahara, M. Imai, and Y. Anzai, "Cahra: Collision avoidance system for humanoid robot arms with potential field," in *Proc. IEEE Int. Conf. on Systems, Man and Cybernetics*, vol. 3, pp. 2889–2895, 10-13 Oct 2004.

[9] H. Taubig, B. Bauml, and U. Frese, "Real-time swept volume and distance computation for self collision detection," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems(IROS)*, pp. 1585–1592, 25-30 Sep 2011.

[10] B. H. Lee and C. S. G. Lee, "Collision-free motion planning of two robots," *IEEE Trans. Systems, Man and Cybernetics*, vol. 17, pp. 21–32, Jan 1987.

[11] C. Chang, M. J. Chung, and B. H. Lee, "Collision avoidance of two general robot manipulators by minimum delay time," *IEEE Trans. Systems, Man and Cybernetics*, vol. 24, pp. 517–522, Mar 1994.

[12] J. Roach and M. Boaz, "Coordinating the motions of robot arms in a common workspace," *Proc. IEEE Int. J. of Robotics and Automation*, vol. 3, pp. 437–444, Oct 1987.

[13] Z. Bien and J. Lee, "A minimum-time trajectory planning method for two robots," *IEEE Trans. Robotics and Automation*, vol. 8, pp. 414–418, June 1992.

[14] J. Lee, "A dynamic programming approach to near minimum-time trajectory planning for two robots," *IEEE Trans. Robotics and Automation*, vol. 11, pp. 160–164, Feb 1995.

[15] Z. Shiller and S. Dubowsky, "On computing the global time-optimal motions of robotic manipulators in the presence of obstacles," *IEEE Trans. Robotics and Automation*, vol. 7, pp. 785–797, Dec 1991.

[16] H. Onda, T. Hasegawa, and T. Matsui, "Collision avoidance for a multiple-dof manipulator based on empty space analysis of the 3-d real world," *Int. J. of Robotics and Mechatronics*, vol. 4, no. 5, pp. 430–436, 1992.

[17] H. Onda, T. Hasegawa, and T. Matsui, "Collision avoidance for a 6-dof manipulator based on empty space analysis of the 3-d real world," in *Proc. IEEE Int. Workshop on Intelligent Robot and System(IROS)*, pp. 583–589, 3-6 Jul 1990.

[18] R. Zurawski and S. Phang, "Path planning for robot arms operating in a common workspace," in *Proc. IEEE Int. Conf on Industrial Electronics, Control, Instrumentation, and Automation. Power Electronics and Motion Control*, pp. 618–623, 9-13 Nov 1992.

[19] C. Seshadri and A. Ghosh, "Minimum-time trajectory planning for two robots," in *Annual Conf. of IEEE on Industrial Electroincs Society*, vol. 1, pp. 676–681, 27-30 Nov 1990.

[20] C. Seshadri and A. Ghosh, "Optimum path planning for robot manipualtors amid static and dynamic obstacles," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 23, pp. 576–584, Apr 1993.

[21] A. Mohri, M. Yamamoto, and S. Marushima, "Collision-free trajectory planning for two manipulators using virtual coordination space," in *Proc. IEEE Int. Conf on Robotics and Automation(ICRA)*, vol. 2, pp. 674–679, 2-6 May 1993.

[22] S. W. Lee, Y. S. Nam, K. D. Lee, and B. H. Lee, "A safety arc based collision avoidance algorithm of a two-arm robot manipulator," in *Proc. 35th SICE Annual Conf. Int. Session Papers*, pp. 1167–1172, 1996.

[23] F. Schwarzer, M. Saha, and J.-C. Latombe, "Adaptive dynamic collision checking for single and multiple articulated robots in complex environments," *IEEE Trans. Robotics*, vol. 21, pp. 338–353, June 2005.

[24] N. Asakawa and Y. Kanjo, "Collision avoidance of a welding robot for a large structure(application of human experience)," *Int. J. of Automation Technology*, vol. 7, no. 1, pp. 88–94, 2013.

[25] N. Asakawa and Y. Kanjo, "Collision avoidance of a welding robot for a large structure(application of potential field)," *Int. J. of Automation Technology*, vol. 7, no. 2, pp. 190–195, 2013.

[26] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proc. IEEE Int. Conf. on Robotics and Automation(ICRA)*, vol. 2, pp. 500–505, Mar 1985.

[27] S. Kalaycioglu, M. Tandirci, and D. Nesculescu, "Real-time collision avoidance of robot manipulators for unstructured environments," in *Proc. IEEE Int. Conf. on Robotics and Automation*, vol. 1, pp. 44–51, 2-6 May 1993.

[28] E. Freund and H. Hoyer, "Pathfinding in multi-robot systems: Solution and applications," in *Proc. IEEE Int. Conf. on Robotics and Automation(ICRA)*, vol. 3, pp. 103–111, Apr 1986.

[29] Y. P. Chien and A. J. Koivo, "On-line generation of collision-free trajectories for multiple robots," in *Proc. IEEE Int. Conf. on Robotics and Automation(ICRA)*, vol. 1, pp. 209–214, 24-29 Apr 1988.

[30] P. A. O'Donnell and T. Lozano-Periz, "Deadlock-free and collision-free coordination of two robot manipulators," in *Proc. IEEE Int. Conf on Robotics and Automation(ICRA)*, vol. 1, pp. 484–489, 14-19 May 1989.

[31] K. Sun and V. J. Lumelsky, "Motion planning for three-link robot arm manipulators operating in an unknown three-dimensional environment," in *Proc. IEEE Int. Conf on Decision and Control*, vol. 1, pp. 1019–1026, 11-13 Dec 1991.

[32] C. Czarnecki, "Collision free motion planning for two robots operating in a common workspace," in *Int. Conf. on Control (Control'94)*, vol. 2, pp. 1006–1011, 21-24 Mar 1994.

[33] H. Hoyer, M. Gerke, and U. Borgolte, "Online collision avoidance for industrial robots with six degrees of freedom," in *Proc. IEEE Int. Conf. on Robotics and Automation(ICRA)*, vol. 2, pp. 1258–1265, 8-13 May 1994.

[34] U. Borgolte, H. Hoyer, and F. Wrosch, "Online collision avoidance for two robots in 3d-space," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robot and Systems (IROS)*, pp. 1919–1926, 26-30 Jul 1993.

[35] X. Cheng, "On-line collision-free path planning for service and assembly tasks by a two-arm robot," in *Proc. IEEE Int. Conf. on Robotics and Automation*, vol. 2, pp. 1523–1528, 21-27 May 1995.

[36] L. Tsai-Yen and J.-C. Latombe, "On-line manipulation planning for two robot arms in a dynamic environment," in *Proc. IEEE Conf. on Robotics and Automation*, vol. 1, pp. 1048–1055, 21-27 May 1995.

[37] R. Z. Lise Cellier, Pierre Dauchez and M. Uchiyama, "Collision avoidance for a two-arm robot by reflex actions: Simulations and experimentations," *J. of Intelligent and Robotic Systems*, vol. 2, no. 14, pp. 219–238, 1995.

[38] T. Sugie, K. Fujimoto, and Y. Kito, "Obstacle avoidance of manipulators with rate constraints," *IEEE Trans. Robotics and Automation*, vol. 19, pp. 168–174, Feb 2003.

[39] K. Fujimoto and T. Sugie, "Freedom in coordinate transformation for exact linearization and its application to transient behavior improvement," *Automatica*, vol. 37, pp. 137–144, 2001.

[40] K.-S. Hwang and M.-D. Tsai, "On-line collision-avoidance trajectory planning of two planar robots based on geometric modeling," *J. of Information Science and Engineering*, vol. 15, pp. 131–152, 1999.

[41] K.-S. Hwang, M.-Y. Ju, and Y.-J. Chen, "Speed alteration strategy for multijoint robots in co-working environment," *IEEE Trans. Industrial Electronics*, vol. 50, pp. 385–393, Apr 2003.

[42] E. Freund and J. Rossman, "The basic ideas of a proven dynamic collision avoidance approach for multi-robot manipulator systems," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, vol. 2, pp. 1173–1177, 27-31 Oct 2003.

[43] J. Rossmann, "On-line collision avoidance for multi-robot systems: a new solution considering the robots' dynamics," in *Proc. IEE/SICE/RSJ Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems*, pp. 249–256, 8-11 Dec 1996.

[44] J.-G. Juang, "Application of repulsive force and genetic algorithm to multi-manipulator collision avoidance," in *Proc. Asian Control Conference*, vol. 2, pp. 971–976, 20-23 Jul 2004.

[45] J.-H. Chuang, C. C. Lin, J. H. Kao, and C. T. Hsieh, "A potential-based path planning of articulated robots with 2-dof joints," in *Proc. IEEE Int. Conf. on Robotics and Automation(ICRA)*, pp. 1815–1820, 18-22 Apr 2005.

[46] S. Leonard, E. A. Croft, and J. J. Little, "Planning collision-free and occlusion-free paths for industrial manipulators with eye-to-hand configuration," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems(IROS)*, pp. 5083–5088, 10-15 Oct 2009.

[47] J.-C. L. L. E. Kavraki, P. Svestka and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration space," *IEEE Trans. on Robotics and Automation*, vol. 12, pp. 566–580, Aug 1996.

[48] J. Cascio, M. Karpenko, Q. Gong, P. Sekhavat, and I. Ross, "Smooth proximity computation for collision-free optimal control of multiple robotic manipulators," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems(IROS)*, pp. 2452–2457, 10-15 Oct 2009.

[49] P. Bosscher, D. Hendman, H. Corporation, and P. Bay, "Real-time collision avoidance algorithm for robotic manipulators," in *Proc. IEEE Int. Conf. on Technologies for Practical Robot Applications (TePRA 2009)*, pp. 113–122, 9-10 Nov 2009.

[50] H. Chung and S. Jeon, "Collision-free trajectory generation of robotic manipulators using receding horizon strategy," in *Proc. American Control Conference*, pp. 1692–1997, 2011.

[51] R. G. Beaumont and R. M. Crowder, "Real-time collision avoidance in two-armed robotic systems," *J. of Computer-Aided Engineering*, vol. 8, pp. 233–240, Dec 1991.

[52] C. A. Shaffer and G. M. Herb, "A real-time robot arm collision avoidance system," *IEEE Trans. Robotics and Automation*, vol. 8, pp. 149–160, Apr 1992.

[53] V. J. Lumelsky and E. Cheung, "Real-time collision avoidance in teleoperated whole-sensitive robot arm manipulators," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 23, pp. 194–203, Feb 1993.

[54] B. Bon and H. Seraji, "On-line collision avoidance for the ranger telerobotic flight experiment," in *Proc. IEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 2041–2048, Apr 1996.

[55] A. Spencer, M. Pryor, C. Kapoor, and D. Tesar, "Collision avoidance techniques for tele-operated and autonomous manipulators in overlapping workspaces," in *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 2910–2915, 19-23 May 2008.

[56] J. Zhou and Y. Aiyama, "Efficient collision avoidance method of two command-based manipulators using partitioned workspace," in *31th Annual Conf. of the Robotics Society of Japan (RSJ 2013)*, pp. 2–5, 04-06 Sep 2013(In Japanese).

[57] T. Lozano-Perez, "Spatial planning: A configuration space approach," *IEEE Trans. Computers*, vol. C-32, pp. 108–120, Feb 1983.

[58] M. Perez-Francisco, A. P. del Pobil, and B. Martinez, "Fast collision detection for realistic multiple moving robots," in *Proc. IEEE Int. Conf. on Advanced Robotics(ICAR)*, pp. 187–192, 7-9 Jul 1997.

[59] M. C. L. E. Larsen, S. Gottschalk and D. Manocha, "Fast proximity queries with swept sphere volumes," tech. rep., Department of Computer Science, UNC Chapel Hill, 1999.

[60] E. Larsenand S. Gottschalk, M. Lin, and D. Manocha, "Fast distance queries with rectangular swept sphere volumes," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, vol. 4, pp. 3719–3726, 2000.

[61] J. Klosowski, *Efficient Collision Detection for Interactive 3D Graphics and Virtual Environments*. PhD thesis, State University of New York, 1998.

[62] S. Gottschalk, *Collision Queries using Oriented Bounding Boxes*. PhD thesis, Dept. of Computer Science , University of North Carolina, 2000.

[63] G. Bradshaw, *Bounding Volume Hierarchies for Level-of-Detail Collision Handling*. PhD thesis, Trinity College , University of Dublin, 2002.

[64] B. Mirtich, "V-clip: Fast and robust polyhedral collision detection," vol. 17, no. 3, pp. 177–208, 1998.

[65] GD Robotics. http://www.gdrobot.com/Upload/Bg/gd_robotics_1362012161126446.jpg. Accessed: 2015-01-30.

[66] Fanuc Corporation. http://www.fanuc.co.jp/en/product/robot/baradumi.html. Accessed: 2015-01-30.

[67] RobotWorx. http://www.robots.com/articles/viewing/manufacturing-or-replicating. Accessed: 2015-01-30.

[68] Global Robots LTD. http://www.robotsltd.co.uk/product.aspx?product=22188. Accessed: 2015-01-30.

[69] D. Sunday, "Distance between 3d lines and segments." http://geomalgorithms.com/a07-_distance.html, 2012. Accessed: 2015-01-01.