To appear in *Advanced Robotics*
Vol. 00, No. 00, January 2013, 1–17

FULL PAPER

On-line Collision Avoidance System for Two PTP Command-based Manipulators with Distributed Controller

Jianing Zhou[a*] and Yasumichi Aiyama[b]

[a] *Department of Intelligent Interaction Technologies, Graduate School of System and Information Engineering, University of Tsukuba, Tenoudai 1-1-1, Tsukuba, Ibaraki, Japan*;
[b] *Division of Intelligent Interaction Technologies, Faculty of System and Information Engineering, University of Tsukuba, Tenoudai 1-1-1, Tsukuba, Ibaraki, Japan*

(*v1.0 released January 2013*)

This research aims to solve on-line collision avoidance problem of two manipulators with independent controller. Since industrial robot controller is a closed commercial system, trajectory generation part of robot controlling is always proprietary or unknown. Thus, this paper proposes a collision avoidance system of two manipulators which are controlled by Point-to-Point(PTP) commands, in condition that the internal of robot controller is unknown and unchangeable. Based on this condition, collision avoidance is supposed to be realized by on-line scheduling of these PTP controlling commands. This paper proposes the collision avoidance method that assumes the three-dimensional common workspace between two manipulators can be partitioned into many sub-region elements. And with managing these sub-region elements, which are occupied by robot motion, PTP commands are scheduled to adjust execution timing for collision avoidance. A deadlock problem caused by the partition of the workspace is also taken into consideration in the method. And the effectiveness and efficiency of the method have been verified by simulations and experiments.

Keywords: Industrial robots, Collision avoidance, Deadlock avoidance, On-line scheduling, PTP control

1. Introduction

Recently, many intelligent robot systems have been developed to satisfy the diversification of customer's needs and differentiation, such as robot controlling cell production system[14]. In these systems, multiple robots are widely applied to enhance the flexibility and efficiency and the collision avoidance between multiple robots needs sufficient attentions. The robots' motion, like in these systems, once depends on sensor information like vision, it would be difficult to verify all potential collision situations between these robots in advance. Thus, on-line collision avoidance is becoming increasingly more important and necessary.

Among those on-line collision avoidance methods, the concepts that "can detect collision in real time" , and "modifying trajectory in real time" are two most desirable. However limited by the computational capability of controllers and communication delay, factory application of those two concept methods is still difficult. In fact, the robot controller in the market have been produced by many different manufactures and the internal of these robot controllers are always designed as "a black box". So it is hard to program freely inside the robot controller, particularly for the majority of robot users. And this results in difficulty in industrial robot's

*Corresponding author. Email: aiyama@esys.tsukuba.ac.jp

trajectory modification. Therefore, this paper focuses on on-line collision avoidance that can be applied in factory.

Researches about collision avoidance of multiple robotic arms are of large quantity, and generally can be divided into on-line and off-line two categories. Lee et al.[1] suggested a method of collision maps to detect potential collision and then made a time scheduling of all the motion commands to avoid collision. Chang et al.[2] made an extension of Lee's method and proposed a more accurate collision avoidance method with the minimum time delay. Zurawski et al.[3] proposed a method from common workspace's viewpoint and allocated two robot arms with master/slave status to avoid collision. These off-line methods' common feature is to implement all decided motion commands sequentially and they are difficult to deal with the situation when unpredictable commands are input. However, although these methods[1][2][3] have been primarily developed for off-line collision avoidance, it can be also, with some additional conditions, applied for avoiding collision in on-line mode.

On the other hand, many excellent on-line collision avoidance methods have been developed. For instance, [16] addressed an algorithm that provided an on-line and real-time generation of dual-arm collision-free trajectories using virtual road map. [4] presented a method to determine a trajectory of manipulator by potential field to avoid collision in real time, [5] treated robot's trajectory coordination problem as a job-shop scheduling problem and avoid collision and deadlock using task-completion diagram. [6] proposed a method, which is based on reactive paradigm, for calculating collision-free motion of manipulator arms in real time. And in [15], a collision avoidance problem is formulated and solved as position-based force control problem and avoid collision using virtual forces. Although this method is simple and computationally fast, it requires a little modification to the robot control system. The merit of these on-line methods above is able to avoid collision in real time even though there are unplanned robot motion or unpredictable trajectory changing situations. However, because industrial robot is difficult to change the trajectory which is inside the controller and considering the on-line computational burden, these research results can not be easily applied in factory application.

Meanwhile, some researches proposed on-line collision avoidance methods that taking the industrial robot controller into account. Koji[7] proposed a new on-line collision check method by each approach speed among the collision check models of manipulators and showed the effectiveness of practical use on industrial robots. But the method has a condition that implemented robot controller is only can calculate the robot's trajectory modifying in a few ms or more. Afaghani et al.[8] developed the method of collision maps, which proposed by Lee et al. [1], and make an on-line scheduling to avoid collision between two manipulators that controlled by PTP commands. However, although PTP command is widely used in factory, this method needs to know the path generation algorithm of PTP control commands of each robot controller to make collision maps, which is the key of the method. Zhou et al.[9] also used PTP command-based manipulators like Afaghani et al.[8], but since the method is only using flags to interlock common workspace for avoiding collision, it is unnecessary to know the internal of robot controller. It is safe but inefficient. Cheng[13] suggested the collision-free paths are planned using a 2D geometric model and are based on a scheduling concept in consideration of swept areas by the robot arms for parallel execution of independent tasks. From the methods above, this paper argues that having a different condition results a different effect on collision avoidance. To achieve industrial application of collision avoidance methods, industrial robot controllers would be regarded as a significant condition.

The paper is organized as follows. At first, the second chapter explains the premise condition of this research. In the third chapter, it introduces a distributed controlling system taken by the research, and proposed a new collision avoidance system architecture based on it. Then the next chapter explains the algorithm of on-line collision avoidance. In the fifth chapter, this paper evaluates the effectiveness and efficiency of this method via presenting some comparison results of simulation and experiments.

2. Problem Setting

The former chapter has introduced some great online collision avoidance methods. But these methods have seldom been applied in the factory. This paper supposes that the reason might be the difference of industrial robot controllers, which are regarded as premise, between researching and practice. Aiyama et al.[12] categorizes the procedure of robot motion controlling as follows; (1)task planning, (2)motion planning, (3)trajectory generation, and (4)execution. This task planning of (1) is determined by application, so it can be considered apart from robot controller. The rest three stages in these three controlling structures–the structure of traditional industrial manipulator controller, usual controlling structure of research field and three-level controlling structure authorized by sub-working-group inside the NEDO next generation intelligent robot project[10] – summarized in Table 1.

Table 1. Robot controller layers

	Motion planning	Trajectory generation	Execution
Traditional robots	Off-line manual work	Inside robot controller	
Typical research robots	On-line/off-line program on PC		PC controller
Intelligent robot project [10]	High-level	Middle-level	Low-level

In general every industrial robot has its own controller, and all the programs (trajectory generation and execution) controlling this robot have been written inside it. And the motion planning part is carried out by operators using off-line teaching method. And robot-controlling structure proposed by the sub-working-group inside the NEDO next generation intelligent robot project[10] is a three level distributed control system and from this paper' viewpoint it may be similar with the industrial controller. The details would be introduced in the third chapter.

Motion planning of traditional industrial robot controllers is always implemented in the off-line mode. It needs in advance empty all the unpredictable space required by motions that might collide. Hence it is safe but inefficient. In the research field, most collision avoidance methods tend to plan the two parts of motion planning and trajectory generation together. Nevertheless in practice, the program of trajectory generation has been equipped inside the controller, where is difficult to access. Therefore if the collision avoidance methods in research field need to be applied in industrial robots, it is necessary to make a large difference on current industrial controller. Controllers proposed by the SWG of NEDO also have the feature of not limiting off-line, and can plan motions in the on-line mode. Moreover with distributed controlling system, the module is highly recyclable. And the change of module makes motions of robots carry out simulation easier in the places like CG. The largest difference between traditional industrial controllers and it is that it can successively generate varieties of commands in the on-line mode instead of needing to store all the programmes in advance like traditional industrial controllers do.

Because this research focuses on on-line collision avoidance of the situation in which robot motion cannot be determined in the off-line mode, and considers practical industrial application, it takes the third controlling structure. Based on this structure, this research proposes collision avoidance methods of multiple manipulators, in condition of inside robot controller is unknown and no changing. Furthermore, to applied in practical industry better, this method has set two premise conditions:

- two robotic arms are controlled by two independent controllers separately;
- using PTP(Point-to-Point) motion commands.

Based on above conditions, the collision avoidance in following applications with two manipulators would be discussed in this research:

(1) bin-pinking application
(2) assembly application

Bin-pinking application is to recognize the position and postures of disorder objects using some sensors such as vision ones and then manipulators will get these recognized objects and put them in order. When multiple manipulators carrying out the application, those arm' motion commands are difficult to be predicted in advance because the recognizing timing, positions and postures of objects are different.

Assembly application in factory usually needs to pick up arranged objects in first, according to some production plans and then carry out assembly operations. But with the coming of multiple variables times, production plans are likely to be modified frequently. In addition, intelligent robot systems in the new generation pursue the flexibility which adapts to multiple productions, so it is important for those systems to develop to reduce cost brought by the change of collision avoidance plan and application environment rebuilding.

Therefore this research aims to propose an effective on-line collision avoidance method of two manipulators whose ranges of movement have an overlap in their workspace. It sets these two applications as assumed operations and two manipulators as assumed number. Figure 1 shows an example of assumed application that need to consider on-line collision avoidance. It is an "automatic candy serving system" with two manipulators and except robots' motion all environments around are prior to be known. These two manipulators are offering different candies which are requested by users, so the robots' motion are unknown in advance and potential collision occurs. In the cases like this, the on-line collision avoidance would be very effective.

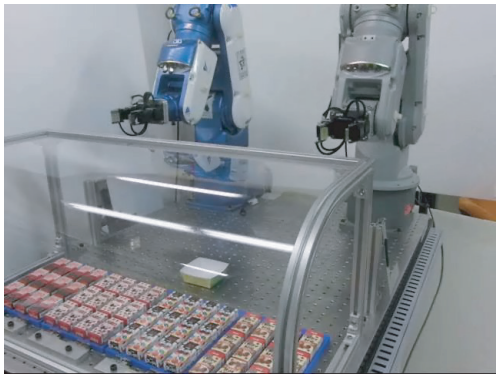


Figure 1. Automatic candy serving system as an example of application

3. System Architecture with Two Manipulators

This chapter introduces the robot controlling system that we used in this research and proposes a new collision avoidance system with two manipulators. In this research, we classify robot's movement in three units: 1)operation, 2)task, 3)motion, and define "motion" as the minimum unit of robot's movement. For example, in the application that is illustrated in Figure 1, "1)operation" like arms are serving with sweets according to users' orders, "2)task" like an arm is taking candy A or an arm is putting B into box, "3)motion" like an arm is moving to position P(x, y, z, roll, pitch, yaw) or close gripper. These robot's movements are controlled by different commands and the commands control "motion" are PTP commands.

3.1 *Distributed Control System*

According to the standard regulation that is proposed by "NEDO intelligent robot system project [10]", robot control system can be divided into three control levels as shown in Table 2.

Low-level controller is directly to access manipulator controller to control its motion in joint coordinates. Middle-level controller is to determine path generation and do the calculation of

Table 2. Three Level Interface for Distributed Robot Control System

Level	Description
Low	Interface refers to be directly controlled by the joint unit commands
Middle	Interface refers to be controlled by PTP commands in Cartesian coordinates
High	Interface refers to execute tasks which are a plurality of motion commands

inverse kinematic of manipulator. It also deals with received motion commands from upper level in Cartesian coordinates. High-level controller is always to describe motion commands' detail according to application. An example of the distributed robot control system, which is using common interface, is illustrated in Figure 2.

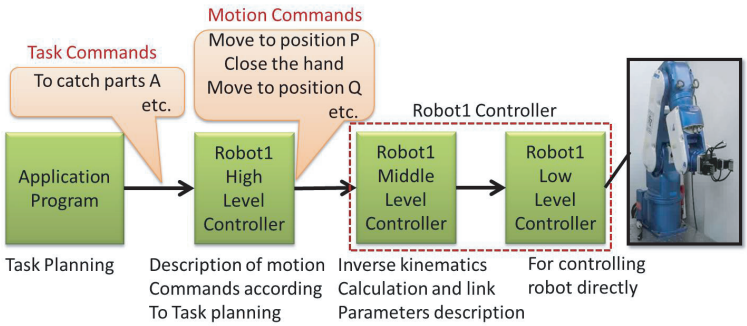


Figure 2. Distributed robot control system

In this system, high-level controller receives task commands from application program and transfer them to a plurality of motion commands, and then these commands are being sent to middle-level controller. Because middle-level and low-level controller in the system is using PTP command controlling method, which is widely used in factory, these two parts can be considered as one industrial robot controller. However, unlike the industrial robot controller always need keep store all the programs in advance, high-level controller may not be off-line like that, on-line commands in this system are possible.

3.2 *System Architecture with Two Manipulators*

In general, as shown in Figure 3, two manipulators' controlling system is simply combined by two individual-manipulator controlling systems. But it is highly possible that potential collision

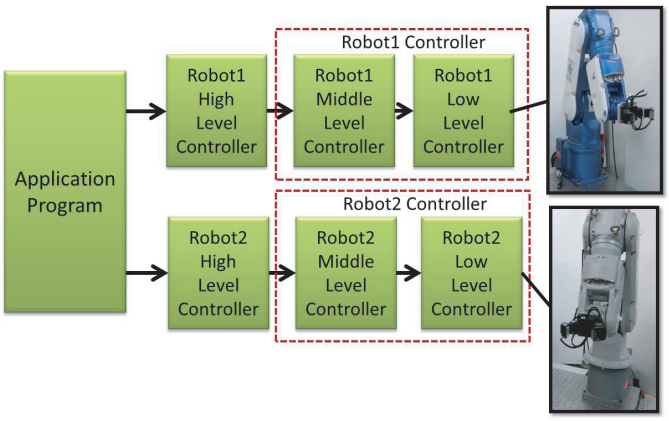


Figure 3. Two robot control system

occurs because no connection exits between two individual-manipulator systems. So, the collision avoidance between two individual manipulators is usually arranged in the application program in advance. Especially in factory, two manipulators' commands are decided in the sequence to avoid collision and they repeatedly move in order according to specific operation. However, in this system it is difficult and time consuming for application programmer to redevelop collision avoidance algorithm when operation planning is changing. In addition, since two high-level controllers are independent, the timing of motion commands from two high-level is difficult to adjust, such as the operation that robot motion commands are based on on-line vision sensors.

Therefore, as illustrated in Figure 4, a new collision avoidance system for two manipulators is proposed in this paper. Without making any changes to the original system, the new system is incorporated another module called collision avoidance planner between the high-level and the middle-level. Through the collision avoidance planner, the new system is able to arrange PTP motion commands from two high-level controllers and send them to two middle-level controllers. Moreover, in proposal system architecture, in application program re-planning collision-free motion is unnecessary when operation changing, so it is possible to reduce the burden on the application programmers.

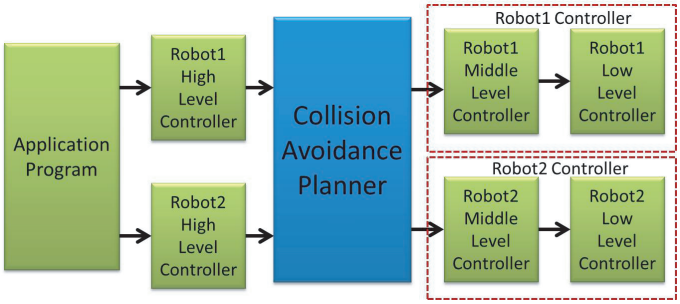


Figure 4. Proposed system architecture for two robots

3.3 *Collision Avoidance Planner*

Collision avoidance algorithm is implemented in this collision avoidance planner module. The main function of it is to receive on-line(or off-line) motion commands from high-level and adjust the command executing timing to avoid collision by proposed collision avoidance algorithm. Every received motion command is firstly stored in the queue and then they will be collision checked for determining if the current command will be executed or waited. According to the premise condition in this paper, since motion commands which have been sent to middle-level are unchangeable, the concept that check the collision before commands executing and adjust timing of executing is effective and practical. Figure 5 shows the simple structure of inside collision avoidance planner.

3.4 *Operation Priority and Collision Avoidance*

The collision avoidance system proposed by this paper is constructed on the foundation of two paralleled manipulator system. So there is no priority between them two. But in practice, it is necessary to think about not only collision avoidance of manipulators but also operation priority problems in some applications. The problem of operation priority is always considered in the application programs. Due to the integration of collision avoidance planner to the system, executing timing of some commands is adjusted to avoid collisions, which makes upper modules difficult master the starting and ending timing of the whole task(a series of commands).

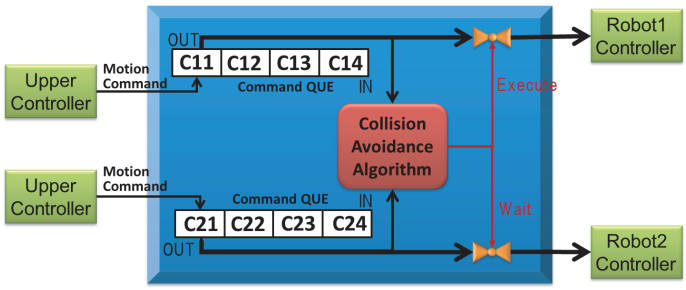


Figure 5. Brief structure of collision avoidance planner

This system manages the operation priority via getting feedback of flags which indicate finishing of task from collision acoidance planner. Then when upper level modules receiving the flag, it would prepare the next task command. By this approach application program in the system manages operations with priority properly. For instance, if when assembling is carried out,

(1) task A: Robot 1 install the part A
(2) task B: Robot 2 install the part B

If task A is prior to task B, in the application program, after receiving flag (A) representing task A finished, the command of task B would be sent.

4. Collision Avoidance Algorithm

In this paper, under the system architecture for two manipulators which is described in the last chapter, two collision avoidance methods are proposed. The first one is called "common workspace interlock (Method 1)" and the other is "partitioned workspace allocation (Method 2)" in this paper. In the interference space of two manipulators workable area, there is a high possibility that collision occurs. So in the algorithms, the whole workspace is partitioned into two regions, common workspace(CWS) and manipulators' respective external workspace(EWS). The collision is assumed to be occurred only in CWS between two arms. In this chapter, a simple method[9] which is using flags to arrange CWS for collision avoidance will be introduced firstly. And then a new collision avoidance algorithm will be proposed in order to improve collision avoidance efficiency.

4.1 *Common Workspace Interlock Method[9]*

The method adopts the simplest concept that using flags to manage the proprietary of the CWS to avoid collision. In the first as shown in Figure 6, depending on the relationship between arm's moving position and the CWS, the arm's motion commands are classified into three types:

(1) Internal command: finally stop inside the CWS.
(2) Pass command: pass through the CWS and finally stop inside the EWS.
(3) External command: move completely outside the CWS.

Internal and pass command are required to compete for the flag which only one arm is permitted to enter the CWS and External command can be executed freely. So, there are five flags to manage the CWS; "Open","Arm1","Arm2","Arm1-Pass","Arm2-Pass." They would be switched by executing different type commands. "Open" represents the CWS is empty and "Arm1" or "Arm2" represents the CWS is occupied by arm1 or arm2. "Arm1-Pass" or "Arm2-Pass" represents the arm occupying the CWS is outside of it when the command is finished. With managing the flags according to executed command, when one arm is occupying the CWS, another one keeps waiting outside the CWS to avoid collision. The algorithm of the method as

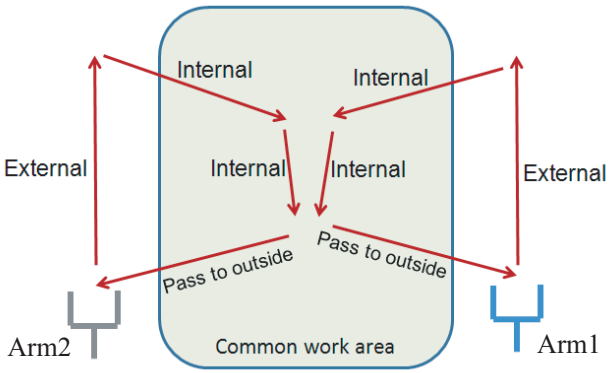
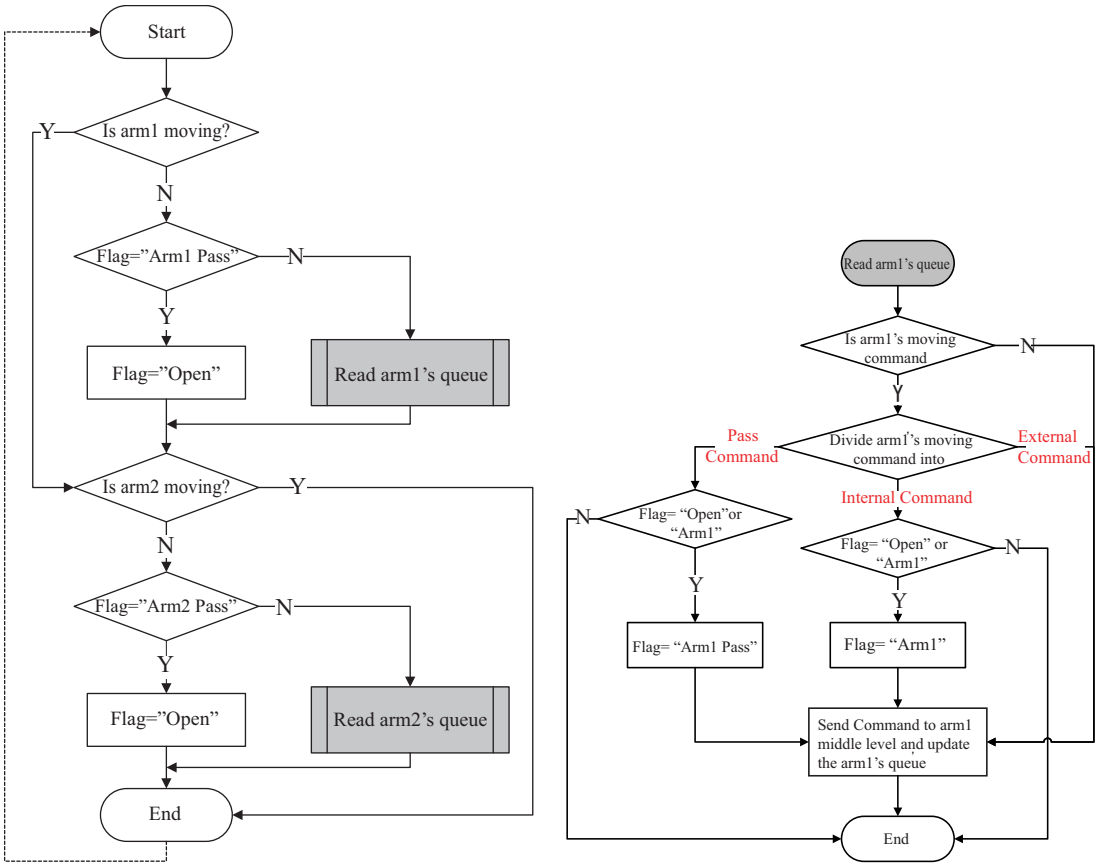


Figure 6. Classification of motion command[9]



(a) Algorithm of collision avoidance planner (b) Algorithm while read arm1's queue

Figure 7. Algorithm Flowchart[9]

shown in Figure 7(a) and (b) is impletemented in collision avoidance planner. The program of arm1 and arm2 is completely the same.

4.2 *Partitioned Workspace Allocation Method*

In the method mentioned in the last section, the whole common workspace is regarded as one region in which collision is possible, so it is simply and safe but not efficient. In order to increase the efficiency of collision avoidance, this paper assumes common workspace would be divided into

many sub-region elements. The more efficient method would be proposed through completing for acquiring necessary sub-region elements where manipulators need to go through in common workspace. Besides, in this paper collision avoidance of the whole manipulator including link and joint is considered.

4.2.1 Robot Modeling

In order to check the whole manipulator's potential collision, robot's modeling method is very important. It is determined by different requirement from collision check/avoidance method, such as precise, mathematical simplicity or computational complexity[17][18]. In this paper, considering on-line collision check and avoidance's computational burden, a spherical modeling method is adopted. As illustrated in Figure 8, the whole robot including link and joint is modeling with several spheres, which is named as Link Modeling Sphere (LMS). This spherical modeling has three parameters: the number of sphere, position of sphere center and radius of sphere. Modeling the different manipulators depends on changes of these parameters[11].

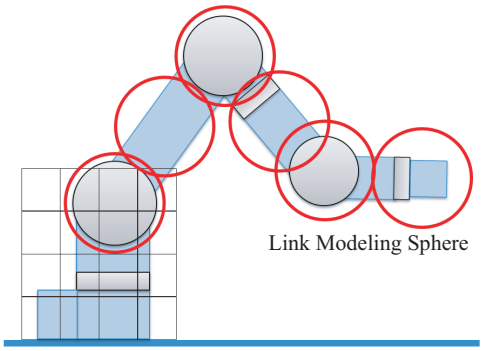


Figure 8. Spherical Modeling of The Whole Robot

4.2.2 Partition of Common Workspace

According to this method, in the first step, the 3D workspace between two manipulators is treated as a cubic region and it is partitioned into many discrete cubic sub-regions. Approximately their circumscribed sphere, which is named as "Workspace Cube Modeling Sphere(WCMS)" could express these sub-regions. Then take these WCMS as common elements, so that the calculation of workspace occupying space is simplified. Through the comparison of the distance between LMS and WCMS with the summation of two radius of LMS and WCMS, the occupation of common elements would be determined, as shown in Figure 9.

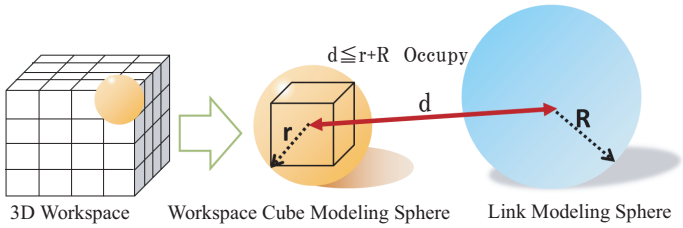


Figure 9. Occupancy Method of Common Elements

4.2.3 Algorithm Overview

The flow chart of collision avoidance process is shown in Figure 10. Collision avoidance process is executed periodically. Every motion command is stored in the queue firstly and then collision checked by four main processes before they are allowed to be executed The collision avoidance's program processes of arm1 and arm2 are similar.

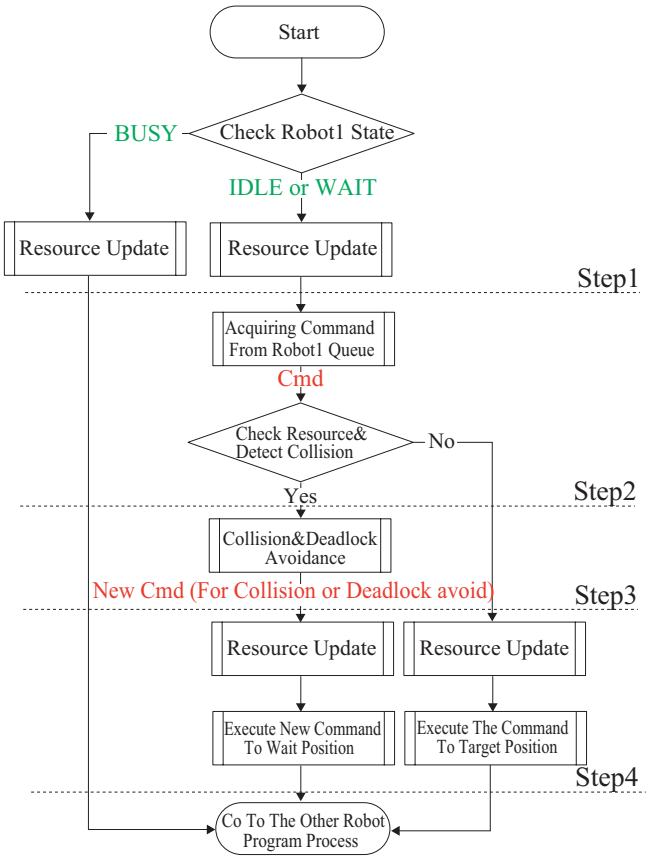


Figure 10. Algorithm Overview

Step1: "The State of Robot and The Update of Common Elements"

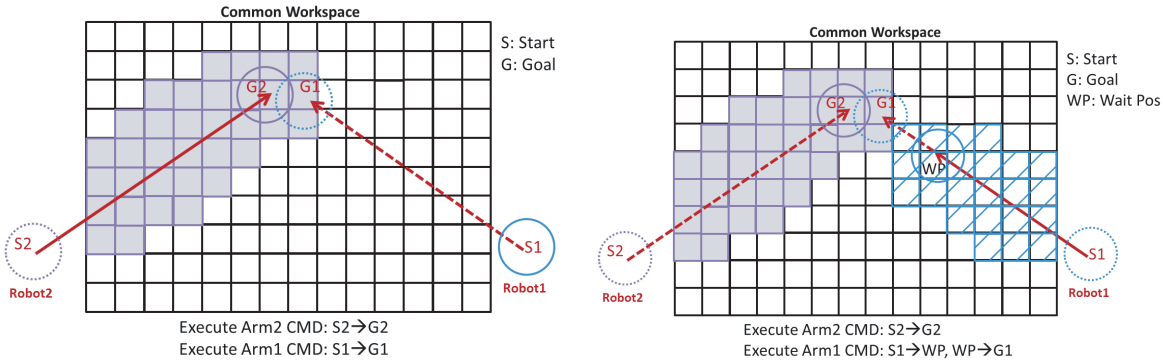
This algorithm defines three states below to express the manipulator movement. These manipulators' states could be switched into others according to the algorithm.

- IDLE: No commands is coming and manipulator is standby.
- BUSY: Commands are executing and manipulator is moving.
- WAIT: Commands are waiting to be executed as lack of common elements, and manipulator is standby.

Common elements occupied by two manipulators are periodically being updated in the algorithm. When manipulator is standby (IDLE and WAIT), common elements occupied by the whole manipulators could be calculated according to the distance between LMS and WCMS. When manipulator is executing commands(BUSY), the moving paths of end effectors is treated as straight line. This path line is assumed combined by a plurality of points and the position and posture of manipulators on these points can be calculated using inverse kinematics. So, the common elements on every point can be calculated in the same way when manipulator is standby. Combining all the common elements needed on all points from the path line could deduce common elements needed by the motion command. Moreover, with the moving of manipulators, spare space would be verified periodically and occupied common elements would be updated.

Step2: "The Collision Check of Motion Commands"

Every motion command controlling manipulator move to its target position is stored in the queue firstly and need to be collision checked before they are executed. Motion commands' potential collision is checked by the intersection of common elements, which needed by each manipulator or the commands of each manipulator. Before every motion command is executed, required common elements of this command need to be reserved. If required common elements are occupied by other manipulators, the algorithmth judges this command



(a) An example of collision detection using common elements

(b) An example of decision of best standby position

Figure 11. Collision detection and avoidance in 2D workspace

would have a possibility of collision. If required common elements are all ensured, it would be judged no potential collision occurs and executed immediately. Figure 11(a) shows an example of collision detcetion using common elements in the 2D workspace. The command from S1 to G1, which would be executed by arm1, is detected potential collision because the commom elements around G1 have already been occupied by arm2's command(S2 to G2).

Step3: "Standby Position Determination for Collision Avoidance"

After a potential collision is detected in step 2, the command that controls manipulator to move to target position is waiting to be executed until required common elements of this command are released. And before that a new standby position need to be decided. In the algorithm, a concept that standby position approaches the target position as close as possible could reduce the waiting time is adopted. Therefore, in step 3, the manipulator would not to be controlled to move to its own target position directly, instead it is searching a most optimal standby position, which named "Waiting Position(WP)". The most optimal standby position is calculated in the path towards the target position, which required common elements are not occupied by others, and is the closest position to its target. The manipulator is waiting in WP until its required common elements are released by the other one. Figure 11(b) shows an example of the decision of best standby postion(WP) in the 2D workspace. Since the command from S1 to G1 has a potential collision, it is unable to be executed directly. Instead, an another command from S1 to WP, which is the best standby position, is firstly executed. And then the command from WP to G1 would not be executed until required common elements is free.

Step4: "Command Execution"

Besides commands, which are stored in the command queue, other commands, for example those are controlling manipulator moving to standby position, also need to be collision checked. In other words, before every motion commands executed, common elements which are required in the command path must be ensured in advance to avoid potential collision. Meantime with the moving of manipulators, all common elements in the workspace would be updated and the common elements that have passed through would be released.

The feature of this method is that manipulators are allowed to move freely within the range of common elements which are reserved in advance so that collision avoidance efficiency would be improved. The amount of calculation is not so much for on-line collision avoidance by the use of spherical approximation modeling. Furthermore, the method's advantage is that although manipulator's position acquisition during manipulator moving depends on the communication between middle-level and upper level, there is no collision even if communication delay occurs. Because the common elements needed from current position to target position are always keeping

reserved. In addition, since the real-time process and real-time communication are not required in any time in factory, this method is very practical and efficient.

However, since in this method the priority of two manipulators is not determined, there is a possibility that different robots' motion order could occur even though completely same motion commands are executed. Besides, because this method is not only one workspace region, a new deadlock problem needs to be considered. In the next chapter, a deadlock avoidance method is proposed.

5. Deadlock Avoidance

If two manipulators are allowed to come into the workspace simultaneously, it may occurs that two manipulators cannot move neither because lack of common elements. It is called as deadlock. With the manipulator motion states mentioned in the last chapter, the following two situations would lead to deadlock:

(1) Arm1 and arm2 are both in the WAIT;
(2) Arm1 (arm2) is in the IDLE and the other is in the WAIT;

In the situation 1, since two manipulators are both waiting the release of common elements, they are not able to move neither. Thus deadlock happens. In the situation 2, the manipulator in the state IDLE has not receive a coming command so that common elements required by the other manipulator in the state WAIT cannot be released.

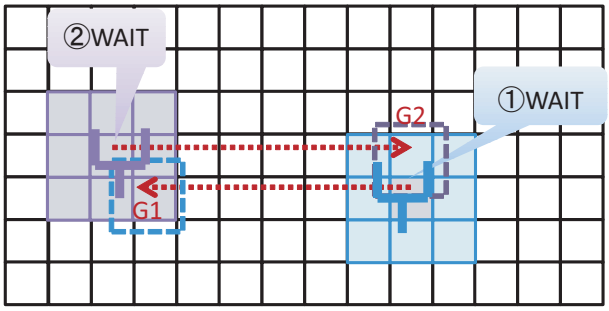


Figure 12. Deadlock example

In this method, when deadlock occurs, one manipulator intends to steps aside to release the common elements needed by the other manipulator. This is the principle of deadlock avoidance. But this stepping aside motion has not been considered in the original motion planning, so it is necessary to assume obstacles in the workspace are pre-known.

In the situation 1, in order to decide which manipulator steps aside, this paper researches several determining methods including the manipulator that comes later and that has a close avoidance distance. In the situation 2, the manipulator in the WAIT has not completed the whole motion, so its priority is set to be higher. Therefore the manipulator in the state IDLE is set to implement avoidance motion.

In these two situation, the stepping aside motion's directions of manipulator are selected from six axis directions of WS coordinate (+x,+y,+z,-x,-y,-z). From these six directions, manipulators calculate the direction in which release common elements fastest to avoid deadlock. In addition after completing avoidance motion, the manipulator does not need to move back to former standby position. Instead, when required common elements available, it directly moves to target position.

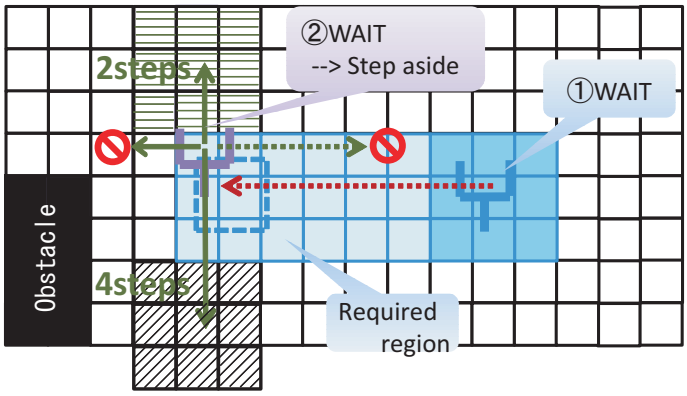


Figure 13. Motion candidate to avoid deadlock

6. Simulation Results

Collision avoidance algorithm is implemented by the collision avoidance system we proposed in chapter 3 and the effectiveness of collision avoidance methods have been verified with a simulator as shown in Figure 14. The simulator can emulate robot motions by giving joint angles with every sampling time. Moreover, in order to investigate the efficiency of the proposal method, a previous collision avoidance method is compared.

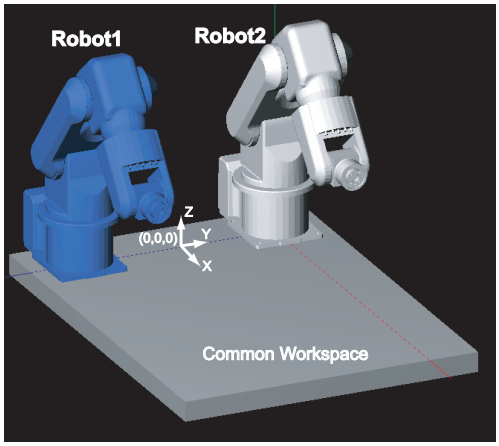


Figure 14. Simulator

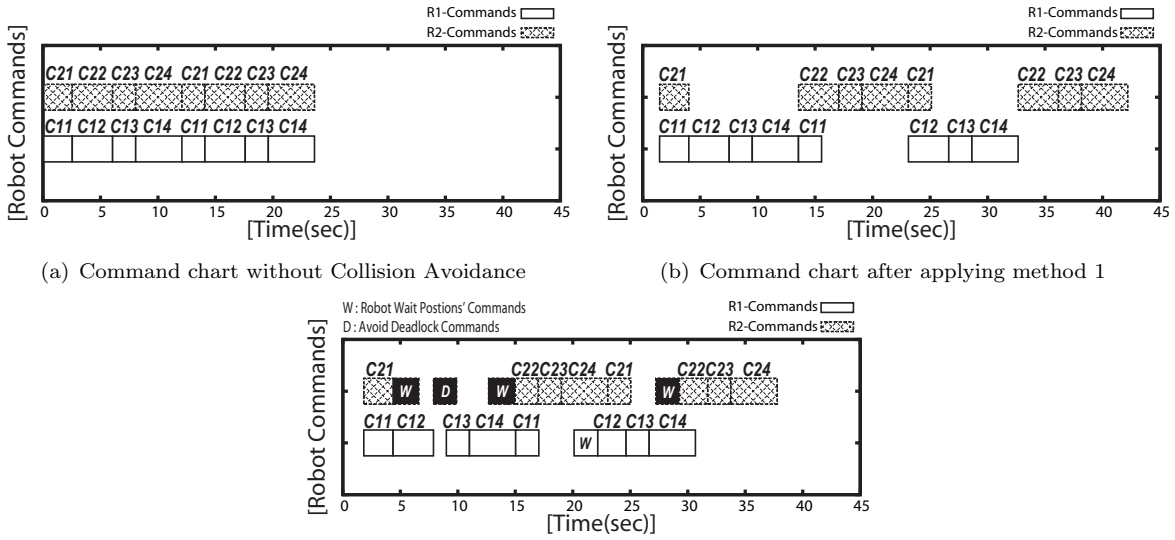
In this simulation, the simulator is used to imitate two Yaskawa MOTOMAN robots (HP3J and UPJ) and there is 500mm distance between central axis of the two robots base. The workspace coordinate is set in the middle between both robot bases. The speed of robot end-effector is 100mm/s, the highest acceleration time is set as 1s. Besides, in the proposal method, WS is partitioned in to cubes with 25mm length of side, the radius of LMS which is used to model the robots is set 50mm. From the straight line of robot motion commands, points with interval of 25mm are taken to make inverse kinematics.

In the simulation experiment, let collision avoidance system including collision avoidance planner perform a series of completely same motion commands and compare robot's motion completion time with two collision avoidance methods. Also, let a system without any collision perform these commands and record the completion time to compare. In this situation, motion commands are sent to middle-level directly without collision avoidance planner. The test motion commands sent to arm1 and arm2 are shown in Table 3 and they are executed twice repeatedly.

The results of simulation is illustrated by three command charts in Figure 15(a), (b) and (c). The horizontal axis is the time and square-shaped represents executed commands by arm1

Table 3. The motion commands sent to arm1 and arm2

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) Command chart without Collision Avoidance

(b) Command chart after applying method 1

(c) Command chart after applying method 2

Figure 15. Simulation results

and arm2. Figure 15(a) illustrates the chart when commands in Table 3 are executed without collision avoidance. Figure 15(b) illustrates the chart after applying collision avoidance method 1 and Figure 15(c) illustrates the chart after applying collision avoidance method 2. In the Figure 15(c) command chart, besides the commands from Table 3, commands that controls robot to move to WP for avoiding collision(attached W) or controls robot to step aside for avoiding deadlock(attached D) are generated. Table 4 shows the completion motion time of each method.

Table 4. Completion time of each method

	Collision avoidance method	Time spend to execute the commands
Without Collision Avoidance	None	T1=23.6[s]
Proposed Method 1	Common workspace interlock	T2=40.7[s]
Proposed Method 2	Workspace partitioning	T3=35.9[s]

From the motion completion time in the Table 4, T2 $\geq$ T3 $\geq$ T1 is clearly shown. Since the robots' motion without collision avoidance is the fastest, it is considered that the more close to T1 the evaluation value of collision avoidance efficiency is higher. For this reason, comparison between the evaluation value of method 2 and method 1 can be expressed by the following formula. The result of this simulation is 28%, which represents proposed method efficiency is

28% higher than the previous method.

$$Q = \frac{(T2 - T1) - (T3 - T1)}{T2 - T1} = 0.28 \tag{1}$$

In addition, the deadlock situation is sucessufully avoid by some steppting aside commands. But it leads to more extra time to complete the whole motions. Thus, in the situation that deadlock does not occur, higher efficiency of the proposed method is expected.

In order to further confirm the effectiveness of the proposed method, simulation experiments not only investigated the efficiency of simple motion commands, and also evaluated several assumed applications. These applications are categorized into two types: one is with work priority and the other is without work priority. Every application are evaluated by proposed method and previous method. An simple picking up example with work priority is shown in Figure 16.

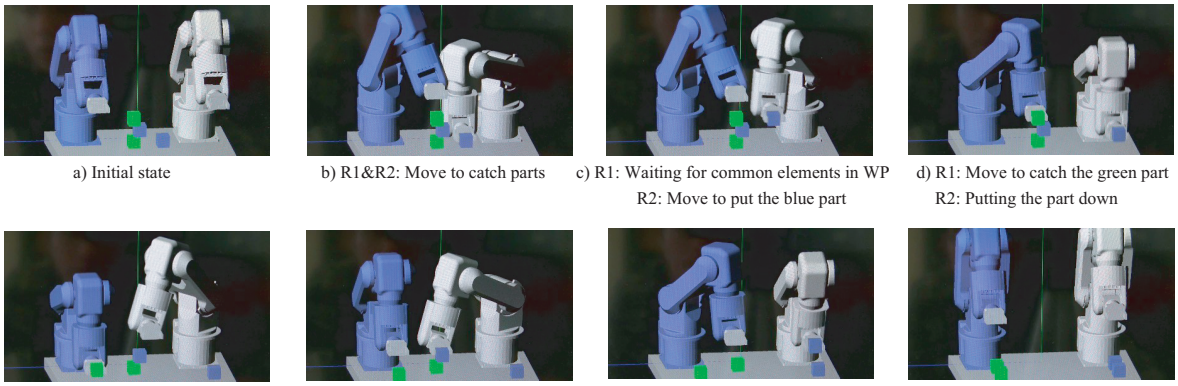


Figure 16. Robots motion in an example of simulation application

Table 5 shows the evaluation value Q of each application by the same evaluation method of the formula 1. By simple average calculation from Table 5, the average evaluation value is 37%, which further verified that the proposed method is more efficient than the previous method.

Table 5. Evaluation value of each example

	No work priority				Work priority exist			
	Ex.1	Ex.2	Ex.3	Ex.4	Ex.5	Ex.6	Ex.7	Ex.8
Q	0.46	0.38	0.39	0.41	0.37	0.28	0.30	0.39

However from Table 5, it is evident that applications with work priority are less efficient in terms of collision avoidance than those without work order. It results from the fact that manipulators are always waiting at the former position, when they need to deal with ordered tasks, instead of moving to optimal standby position like in the case of collision avoidance.

7. Conclusion

This paper proposes that on-line collision avoidance methods of two robots with independent command-based controller. In the applications like bin-picking, since the motion of manipulators depends on some on-line sensors like vision, it is difficult to be collision checked in the off-line mode. Thus collision avoidance among several manipulators becomes more complex. Nevertheless conditions of collision avoidance methods in the researching fields is quite different with

controlling patterns of manipulator controllers in the industry, so to apply these research results directly seems to be difficult relatively. Therefore this paper adopts a new controller that can receive on-line(or off-line) PTP commands and that is similar with industrial manipulator controllers. Based on this controller, this paper proposed two collision avoidance methods. Moreover this research compares these two methods and evaluates their efficiency by experiment and simulation. Consequently it is concluded that method 2 has higher collision avoidance efficiency than method 1. Meantime because the considering of deadlock problem and work priority, different applications of method 2 would lead to different effects. In this research, the collision avoidance system and the methods both target industry manipulators. Hence it is expectable that those manipulator system and collision avoidance technique being equipped in the practical industry field.

References

[1] B. H. Lee and C. S. G. Lee, "Collision-free motion planning of two robots", in Systems, Man and Cybernetics, IEEE Transactions on, vol.17, pp.21-32, Jan 1987.

[2] C. Chang, M. J. Chung, and B.H.Lee, "Collision Avoidance of Two General Robot Manipulators by Minimum Delay Time", In System, Man and Cybernetics, IEEE Transactions on, volume 24, pp.517-522,Mar 1994.

[3] R. Zurawski and S. Phang, "Path planning for robot arms operating in a common workspace", Industrial Electronics,Control,Instrumentation and Automation, Power Electronics and Motion Control,Proc.of the International Conference on, Vol.2, pp.618-623, Nov 1992.

[4] H. Hirukawa and S. Kitamura, "A collision avoidance method for robot manipulators based on the safety first algorithm and potential function", Advanced Robotics,Vol.4,No.1,pp.43-57,1989.

[5] P. A. O'Dnnel and T. Lozano-Perez, "Deadlock-Freeand Collision-Free Coordination of Two Robot Manipulators", Proc. of the IEEE Int. Conf. on Robotics and Automation, pp.484-489, 1989.

[6] P. Bosscher, D. Hendman, H. Corporation, and P. Bay, "Real-Time Collision Avoidance Algorithm for Robotic Manipulators", In Technologies for Practical Robot Applications, TePRA. IEEE International Conference on, pp.113-122, Nov 2009.

[7] Koji Shiratsuchi, "The Development of the Collision Check Mothod for FA Robots by using each Approach Speed", Journal of the Robotics Society of Japan, Vol.31, No.7, pp.697-702, 2013.(in Japanese)

[8] A. Y. Afaghani and Y. Aiyama, "On-Line Collision Avoidance between Two Robot Manipulators Using Collision Map and Simple Escaping Method", in System Integration (SII), IEEE/SICE International Symposium on, Dec 2013.

[9] J. Zhou, K. Nagase, and S. Kimura and Y. Aiyama, "Collision avoidance of two manipulators using rt-middleware", in System Integration (SII), IEEE/SICE International Symposium on, pp.1031-1036, 20-22 Dec 2011.

[10] ACT Middle Level Common Interface specification document of "robot Intelligence Technology for Next-Generation Development Project", 2010.

[11] M. Fuji, H. Murakami, K. Kosuge and F. Seto, "Collision detection method for manipulators using approximated model by spheres", the Robotics and Mechatronics Conference, the Japan Society of Mechanical Engineers, 1A1-C03, 2007.

[12] Y. Aiyama, J. Zhou and A. Y. Afaghani, "Collision Avoidance for Manipulators which have Independent Controllers", in System Integration (SI) division, SICE conference, Dec 2013.

[13] X. Cheng, and Ing. R. Dillmann, "Executing Elementary Assembly Operations by a Two-Arm Robot" IEEE Internat. Conf. on Robotics and Automation. 1993.

[14] Akio Noda*, "Manufacturing Intelligence to Evolve Industrial Robot", Journal of the Robotics Society of Japan, Vol.31, No.1, pp.10-13, 2013.

[15] Homayoun Seraji, Fellow, IEEE, and Bruce Bon, "Real-Time Collision Avoidance for Position-Controlled Manipulators", IEEE Transactions on Robotics and Automation, Vol.15, No.4, August 1999.

[16] S. Lee, H. Moradi and C. Yi, "A Real-Time Dual-Arm Collision Avoidance Algorithm for Assembly", Proceedings of the 1997 IEEE International Symposium on Assembly and Task Planning Marina del Rey, August 1997.

[17] E. G. Gilbert, D. W. Johnson and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space", Robotics and Automation, IEEE Journal of 4.2 (1988): 193-203.

[18] E. Larsen, S. Gottschalk, C. M. Lin and D. Manocha, Fast proximity queries with swept sphere volumes. Technical Report TR99-018, Department of Computer Science, University of North Carolina, 1999.