2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Coordinated Motion Control of a Robot Arm and a Positioning Table with Arrangement of Multiple Goals

Lounell B. Gueta, Ryosuke Chiba, Jun Ota, Tsuyoshi Ueyama, and Tamio Arai

***Abstract*—The minimum-time motion coordination is an important subject in robotics. In this study, the arrangement of several goals, which is treated as a traveling salesman problem (TSP), is incorporated to this subject. Although TSP has been studied in most previous works; but, solving a TSP that takes into account collision occurrences has not received much attention. This instance arises when a robot arm has to plan a sequence of reaching goals with other moving objects and/or other robot arms. If goals are also moving, then the problem becomes more complex since the end configuration of robot arm is undefined when reaching a goal. In this study, in particular, a 6-DOF robot arm has to reach several goals found in an object while a 1-axis positioning table simultaneously positions the object; thereby changing the goal locations and collision occurrences are inevitable. For the purpose of this study, the TSP is solved effectively with motion coordination and collision avoidance. The collision-free configurations of a robot arm when reaching goals are solved through motion coordination. Collision is avoided by exploiting the redundancy of the system. The above-mentioned solution is verified through a simulation utilizing an object with various numbers of goals and their positions, and is proven effective.**

I. Introduction

THE minimum-time completion of task done by a single robot or several robots is very important in manufacturing to meet the demand for high productivity. When two or more robots share a common workspace, the primary concern is the minimum-time and collision-free motion coordination. When a robot has to reach several goals, the arrangement of these goals is normally formulated as a traveling salesman problem (TSP) to obtain minimum-time motion.

Several previous works studied on trajectory planning of a single robot with initial and final configurations while avoiding collision with static obstacles or other robots [1], [2]. In [3], it is shown that planning a collision-free path for a robot among a set of polyhedral obstacles between two goals is a PSPACE-hard problem. In other research works [4]-[6], the trajectory paths of robot arms are given. In [4] and [5], a robot arm with a positioning table is studied in which the robot arm is required to follow a continuous path. In [6], several robots with required trajectories are coordinated while considering dynamic constraints and collision-free motion is solved through changing of robot start times.

L.B. Gueta, R. Chiba, J. Ota, and T. Arai are with Department of Precision Engineering, Graduate School of Engineering, at the University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan. (email:{gueta, chiba, ota, arai-tamio}@robot.t.u-tokyo.ac.jp).

T. Ueyama is with DENSO WAVE INCORPORATED, 1-1, Showa-cho, Kariya-shi, Aichi, 448-8661, Japan.

A few studies in motion coordination incorporate goal arrangement. In [7], the motion coordination of two robot arms with goal arrangement using simulated annealing is studied; however, no collision avoidance was considered. In [8], a study was done on a single robot arm with goal arrangement and collision avoidance on a static obstacle using probabilistic roadmap planner. In [9], a multi-robotic assembly system is optimized using genetic algorithm with coordination of X-Y placement and 2-DOF delivery machines, ordering of components, and collision detection in a 2D environment. Independently, a TSP with static goals has been the topic of most previous works [10]-[14]; however, only a few studies are done on a TSP with moving goals [15]-[17].

This study addresses the motion coordination, the arrangement of moving goals, as well as collision avoidance while minimizing the working time of a robot arm. A setup with a robot arm and a positioning table, as shown in Figure 1, is considered which is commonly used in spot welding, assembly, and inspection works.

When a robot arm has to reach several goals found in an object while a positioning table simultaneously positions the object, the locations of goals also change and collision occurrences may be inevitable. From the view point of solving a TSP, a collision occurrence is normally not considered, which is mainly addressed in this study. In order to avoid collision, no modification of a straight-line path in the configuration space of robot arm and table is done; but instead, the redundancy of the system is utilized to select a configuration that has a collision-free path among several possible configurations of robot arm and table.

For the remaining parts, Section II formulates the problem in this study. Section III describes the proposed method while Section IV provides the algorithms used in detail. The experiment and result are discussed in Section IV while a conclusion and plan is given in Section V.

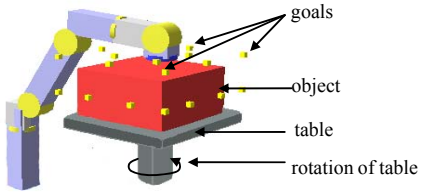


Fig. 1 A setup with a 6-DOF robot arm and a 1-axis positioning table. A tool is held by the end-effector of robot arm while an object is placed on a positioning table. The end-effector moves to several goals which are pre-defined parts of the object while the table positions the object.

II. Problem Formulation

This section describes the problem considered in this study. Let A_o, A_t, and A_r denote the object, table, and robot arm, respectively. Since A_t and A_r compose one redundant system, their configurations at a goal are denoted compactly as $q = \{\theta^0 \ldots \theta^6\}$ where θ^0 is the rotation angle of A_t and $\{\theta^1 \ldots \theta^6\}$ are the joint angles of A_r.

A. Input parameters

1) *The goals $p_1 \ldots p_N$ where N is the number of goals:* A goal is a vector in which its position is its location relative to the object and its orientation describes the required orientation of end-effector when it reaches that goal. Note that $p_1 \ldots p_N$ are detached from A_o (See Fig. 1).

2) *The start and end configurations of both A_r and A_t before and after all the goals are reached:* For the purpose of this study, these configurations are just the same which is referred as q_0 or equivalently, where A_r is solely said to be at goal 0, p_0.

B. Assumptions

1) The robot arm has large reduction ratios that its dynamics can be neglected [10].

2) The object can be bounded by an imaginary rectangular box that excludes goals; effectively, the goals are outside the bounding box.

3) The motion of A_r, A_t, and A_o are sampled at a sufficient time interval so that no collision occurs during movement from one goal to another goal.

C. Constraints

1) *Kinematic constraints:* The joint and velocity limits of robot arm must be satisfied.

2) *Position constraint:* The end-effector must satisfy the required position and orientation when it reaches a goal.

3) *Calculation time constraint:* For practicality, the solution must be obtained at a reasonable amount of time, which is set at 20 minutes. Several other aspects in optimization can be considered as future research works (e.g. base placement, trajectory motion planning); thus, minimizing the calculation time is desired at this stage of study.

D. Problem description and definition

Initially, A_r and A_t are at q_0. Then, A_r moves to all goals; concurrently, A_t rotates and positions A_o. Afterwards, A_r and A_t moves back to q_0.

Let $t_j(a,b)$ denotes the motion time of joint $j \in \{0 \ldots 6\}$ in moving from goals a to b. In addition, let $\pi = (i_1 \ldots i_N)$, which is a permutation of $(1 \ldots N)$, denotes the goal order. The problem is defined as: determine (1) the order π, and (2) the configuration of A_r and A_t, q_i for every $p_i \in (p_{i1} \ldots p_{iN})$ in order to minimize the working time T such that:

$$T = \sum_{k=1}^{N} c(p_{i_{k-1}}, p_{i_k}) + c(p_{i_N}, p_0) \quad (1)$$

where $p_{i0} = p_0$ and $c(a,b) = \max_{j \in \{0..6\}} [t_j(a,b)]$ since the slowest joint in a multi-joint system dictates its motion time. The motions of A_r and A_t are coordinated by finding q_i for every goal $p_i \in (p_{i1} \ldots p_{iN})$ satisfying the constraints and no collision occurs among A_o, A_t, and A_r as A_r moves from one goal to another goal. The above formulation implicitly imposes a point-to-point motion of end-effector, which is appropriate for applications that require the robot arm to stop at every goal to perform a task.

III. Problem Analysis and a Summary of the Proposed Solution

In this section, the problem is analyzed and the proposed solution is presented.

A. Problem Analysis

For concreteness, consider Figure 2. The p_0, as described by the position of end-effector in Fig. 2(a), is located above A_o and is aligned on the rotation axis of A_t. Fig. 2(b) to Fig. 2(f) shows q_i or the configuration of A_r and A_t for $p_i \in \{p_{i1} \ldots p_{iN}\}$. From p_0, A_r moves to p_{i1} as shown from Fig. 2(a) to Fig. 2(b). As for A_t, the direction of motion can either be clockwise or counter-clockwise resulting into several possible locations for p_{i1}, which is described by the dotted circle in Fig. 2(b). Similarly, these locations are the possible end-effector positions of A_r when it plans to reach p_{i1}. Note that this has been possible due to the additional DOF in A_t. This scenario holds true for the remaining goals.

In this system, the travelling distance of A_r is based on the joint space of A_r and A_t. As A_o is rotated by A_t, the joint configuration of A_r is also changed to satisfy the position constraint. For one instance, A_t moves in one particular direction while A_r moves to a direction that shortens its travelling distance between goals. In some possible instance, A_t is stationary while A_r moves, or vice versa, depending on which instance a collision-free motion is possible. In another instance, the travelling distance of end-effector may be longer to avoid collision. In summary, there are three unique aspects of TSP in this study: (a) the travelling distance which is based on the joint configuration space, (b) the goals can be

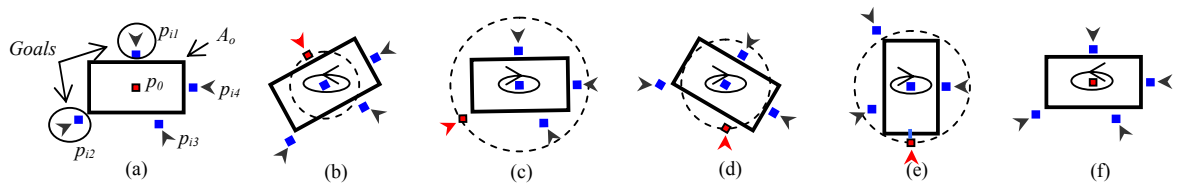


Fig. 2 Some possible configurations of A_r and A_t with 4 goals $\{p_{i1} p_{i4}\}$ (Top view). A_o is the rectangular box. The end-effector of A_r is marked as a red dot while a goal is marked as a blue dot, which signifies its position, and an adjacent arrow, which signifies its orientation. The color of a goal is changed to red when A_r reaches that goal. The bent arrow denotes the direction of movement of A_t. Initially, the end-effector is at p_0 as shown in (a) and after doing a task, it returns to p_0 as shown in (f). The dashed circle traces the possible position of a goal as A_t positions A_o.

positioned by the table resulting into several possible configurations, and (c) the motion from one goal to another can result into collision between a robot arm and a table or an object. Solving this type of TSP is indeed a challenge as there is no identical work that has been done in the past. The movement of goals becomes more complex if an additional DOF is added to A_t that will allow for additional movement, rotational or translational, of goals.

In addition, as the number of goals increases, the number of possible orders increases exponentially and exploring all possible orders becomes impractical. Another complexity is the high-dimensionality of possible positions of $\{p_{i1}\ldots p_{iN}\}$, which heavily loads the calculation together with collision detection in a 3D environment.

B. Proposed Solution

The problem is divided into three sub-problems. For each sub-problem, a solution method is proposed and is analyzed to find opportunities to improve its performance. The three sub-problems are: (1) TSP or goal arrangement, (2) coordinated motion control of A_r and A_t, and (3) collision detection. The cycle to minimize the working time, as shown in Figure 3, is as follows: For N input goals, a possible order is determined in (1). A clustering method is employed in order to reduce the time spent in testing several possible goal orders. In (2), given a goal order, the q_i for every goal is solved. Sub-problem (3) detects if there is collision that occurs in A_r, A_t and A_o when finding a solution to sub-problem (2). As a result, the working time T is calculated. If the calculation time is still within the calculation time constraint, then a new order is considered and the same cycle as described above is done. Otherwise, the cycle is terminated and the solution that has the least working time is considered as the best solution.

Note that an order is derived first before a coordinated motion is obtained since q_i is very dependent on the goal order. Ideally, an algorithm that solves simultaneously the goal order and q_i for every goal is desired. However, with the aforementioned complexity, a practical solution is proposed in this study.

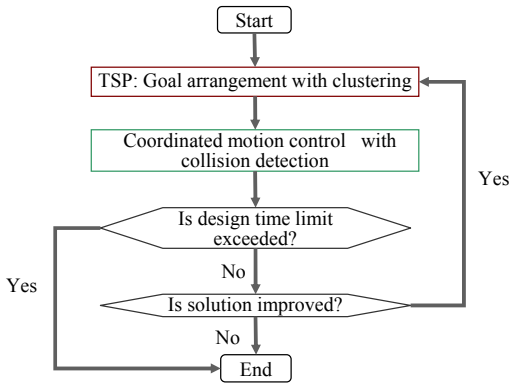


Fig. 3 The Proposed Solution

IV. Details of The Proposed Solution

This section provides the details of the algorithms employed in the solving the TSP, the motion coordination, and the collision detection.

A. TSP: Goal arrangement with clustering

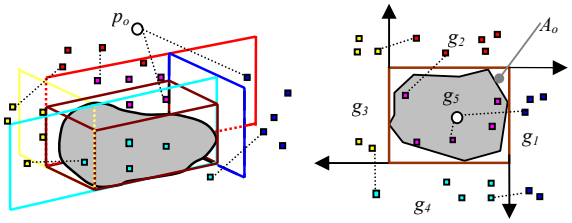
1) *Overview:* The TSP is a combinatorial problem. With N goals, there are $N!$ orders and considering all these orders can take enormous calculation time. For this reason, the goals are clustered into groups.

For a large scale TSP, a mean vector, which is the average position of goals in one cluster, is first determined [18]. These mean vectors are the basis for determining the order of clusters, the solution of which is referred as a *reference TSP*. After deriving a reference TSP, two cities for every cluster is selected that will serve as the connecting goals between clusters. Then, the goal order in every cluster is solved. The direct application of utilizing the mean vectors is, however, inappropriate in this study since it may lead to impossibility in deriving a reference TSP due to collision. The succeeding parts describe the algorithm for solving TSP with clustering.

2) *TSP algorithm:* With clustering, the TSP algorithm involves solving the order of clusters and the goal order for every cluster. In this study, an *ad hoc* algorithm is proposed and its steps are discussed below.

a) *Assign a cluster for every goal*. The goals are clustered based on their topological locations on A_o as shown in Figure 4. This employs the assumption that A_o can be modelled as an approximate box and the box does not enclose any goals. Every goal is associated to a cluster $\{g_1...g_5\}$; each cluster corresponds to the face in the geometric shape of A_o. Since the goals in each cluster are nearer to each other compared to other goals from different clusters, deriving a goal order from the same cluster is more practical than with goals from various clusters. In addition, the motion of A_r between goals in the same cluster has a greater chance of not encountering collision than goals from different clusters.

b) *Determine the order of clusters.* For simplicity, the order of clusters is determined through 2-opt algorithm which exchanges the order of two clusters. The stopping condition for exchanging clusters is when it did not result to an improvement in the solution or when the calculation time limit is already reached.



(a) Isometric view (b) Top view

Fig. 4 Clustering of goals. The planar boundaries in (a), represented as arrows in (b), define the limits in assigning a cluster to every goal. The clusters $\{g_1..g_4\}$ correspond to the sides of A_o while g_5 to its top. If the cluster order is g_1, g_4, g_3 , g_2 and g_5, then the dotted lines show the links connecting a goal from one cluster to a goal of a next cluster.

c) *Connect clusters.* After a cluster order is derived, the clusters are linked by finding their *connecting goals*. There are two connecting goals for every cluster. If a cluster is either the first or the last in a cluster order, then one connecting goal is connected to p_0. This connecting goal is determined by selecting the nearest goal to p_0. For two consecutive clusters, the connecting goals are determined based on the mean vector, which is the mean of goal positions (*x, y,* and *z* values) of a cluster. Consider two consecutive clusters, g_{from} and g_{to}, where g_{from} precedes g_{to}. First, the mean vector of g_{to}, m_{to}, is determined. Then, n_{from}, the goal in g_{from} that is nearest to m_{to}, is selected. Afterwards, n_{to}, the goal in g_{to} that is nearest to n_{from} is determined. The pair (n_{from}, n_{to}) is the connecting goals for g_{from} and g_{to}. Note that these connecting points depend on the cluster order.

d) *Determine the goal order in every cluster.* After connecting clusters, the goal order is determined in each cluster. As can be seen in Fig. 4, the goals of a cluster that are the connecting goals of its preceding and succeeding clusters must be the start and end goals, respectively, when a goal order within that cluster is determined. Therefore, the goal order in a cluster is solved as a TSP with the aforementioned precedence constraints.

In this study, it is conjectured that the goal order that is based on Euclidean distance is a reasonable estimate of their order in the configuration space. If the order is based in the configuration space of A_r and A_t, deriving the goal order becomes very complex since as A_t repositions A_o, the configuration space is changed. If the order is based on the Euclidian distance, then it can be derived using an existing TSP algorithm.

One of the most effective methods in solving a TSP in a relatively short calculation time is the LK heuristic [14]. In this study, the implementation of LK heuristic by Helsgaun is adopted and modified to accommodate the precedence constraints. The reader is referred to [14] for a detailed discussion of LK heuristic.

If the goal order in every cluster is obtained, then the order of goals is completely determined. Note that the goal order is based on the Euclidean distance but the working time T is based on the joint configuration space. In essence, the goal order acts as a guide for A_t on which goal it will reach and the T is calculated based on the actual motion time of A_r and A_t.

B. Coordinated motion control

Given a goal order, the aim is to determine q_i for every $p_i \in \{p_{i1} \ldots p_{iN}\}$. In this study, the possible positions of p_i are determined by θ^0 which is parameterized into discrete values:

$$\boldsymbol{\theta^0}(l) = \theta^{0,min} + l*k \quad (2)$$

where, $l \in \{0,1,\ldots l_{max}\}$, $l_{max}=int(\,(\theta^{0,max}-\theta^{0,min})/k+0.5)\}$, *int()* gives the nearest integer value, k is the discrete step size of θ^0, $\theta^{0,min}$, and $\theta^{0,max} \in \{-180° \ldots 180°\}$ are the minimum and maximum values of θ^0. By discretization, the solutions are inherently just approximates. Nonetheless, an approximate solution that has short calculation time is preferred over a high-quality solution that demands an enormous amount of calculation time.

The position of A_t and p_i are defined by a single θ^0 value. By assigning a value to θ^0, the inverse kinematics of A_t is derived, which completely determines q_i. For every p_i, there are $l_{max}+1$ possible positions and for every order of $\{p_1 \ldots p_N\}$, there are N x ($l_{max}+1$) positions; hence, with the large size of possible positions, it is practical to represent them as a graph. Consequently, the q_i for every goal is solved as a graph search. A detailed discussion is provided below.

1) *Generate search space: A* directed graph $\boldsymbol{G}=(\boldsymbol{V},\boldsymbol{E})$ is created in which the vertices $\boldsymbol{V}$ are sets of q_i. The weighted edges $\boldsymbol{E}$ are the lines connecting the vertices of p_i to p_{i+1}. For every p_i, the possible positions, called *candidate positions*, are determined using (2).

2) *Check validity of candidate positions:* For every candidate positions, $\{\theta^1 \ldots \theta^6\}$ are determined through inverse kinematics (IK) of A_r. If an IK solution exists, then that goal position is said to be valid; otherwise, invalid.

The validity check ensures that the position of goal is reachable by A_r but does not guarantee that a collision-free path exists in moving from a previous goal. By employing first the validity check, the computational burden of checking every pair of vertices is avoided. In addition, the goal position can still be regarded as valid even if a straight line path is not possible since the path can be modified to avoid collision.

3) *Calculate motion time as edge weights:* The motion time in reaching a valid goal position is calculated. For simplicity, it is assumed that a joint achieves its maximum velocity in a very short time; thus, the motion time from goal a to goal b, $t_j(a,b)$, is calculated as:

$$t_j(a, b) = |\theta^j \text{ in } a - \theta^j \text{ in } b\,| / v^{j,max} \quad (3)$$

where $v^{j,max}$ is the maximum velocity of joint j.

4) *Find shortest collision-free path:* A shortest collision-free path is determined by choosing a valid q_i with the least $c(p_i, p_{i+1})$ and then checking if a path has no collision. In this study, the trajectory of A_r is based on a straight-line path in the configuration space of A_r and A_t. No modification of a straight-line path is done when collision exists. The collision is avoided by selecting a valid q_i with a collision-free straight-line path. In a worst-case scenario, no valid q_i that has a collision-free path is possible and therefore, a modification of straight-line paths like roadmap methods becomes inevitable.

In finding the shortest collision-free path, there are three search methods tested and compared in terms of the quality of solution and the calculation time. The definition of $c(p_i, p_{i+1})$ in (1) is extended where i is changed to s to mean a stage in a graph. Let $c_l^s\ (p_s, p_{s+1})$ denotes the motion time at stage $s \in \{0 \ldots N+1\}$ of a valid q_i with collision-free path when $\theta^0 = \boldsymbol{\theta^0}(l)$, C_l^s is the motion time from stage 0 to s and C^s denotes the least motion time at stage s.

a) *Greedy Nearest Neighbor method (GM)*: The selection of q_i is based on only two consecutive goals. At every stage s, C^s is calculated and hence q_i is determined. The calculation involved in GM is shown below.

$$C^1 = \min_{l \in \{0 \ldots l_{max}\}} \{c_l^1(p_0, p_1)\},$$

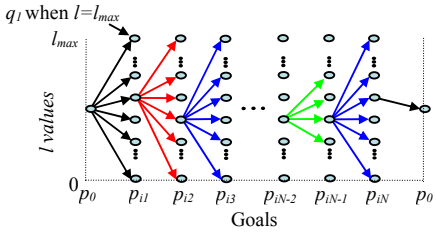


Fig. 5 Search space for the coordinated motion control of A_r and A_t. For N goals, there are N+2 stages with the first and last stages corresponding to p_o.The objective is to select a set of vertices that results to the shortest collision-free path in moving from p_0 and moving back to it.

$$C^2 = \min_{l\in\{0\ldots l_{\max}\}} \{c_l^2(p_1,p_2)\} + C^1$$

$$C^s = \min_{l\in\{0\ldots l_{\max}\}} \{c_l^s(p_s,p_{s+1})\} + C^{s-1},$$

$$T = c^{N+1}(p_N,p_0) + C^N.$$

Fig. 5 shows the graphical representation of GM. The vertices from where the edges emanate correspond to the goal positions with the least motion time at their corresponding stages (i.e. $C^1 \ldots C^s$).

b) *Dijkstra method (DM)*: The DM, a widely-used algorithm, calculates the cost at every vertex with valid q_i and collision-free path. At stage s, the cost of every vertex c_l^s is calculated based on the cost in moving from stage s-1 to s and on the derived minimum cost from stage 0 to s-1. The calculation involved in DM is shown below.

$$C_l^1 = c_l^1(p_0,p_1),$$

$$C_l^2 = \min_{n\in\{0\ldots l_{\max}\}} \{c_l^2(p_1,p_2) + C_n^1\}$$

$$C_l^s = \min_{n\in\{0\ldots l_{\max}\}} \{c_l^s(p_s,p_{s+1}) + C_n^{s-1}\},$$

$$T = \min_{n\in\{0\ldots l_{\max}\}} \{c^{N+1}(p_N,p_0) + C_n^N\}$$

After T is calculated, the search graph is backtracked to find the q_i which is part of the solution.

c) *Rough-to-smooth method (RS)*: The RS method, which is the proposed method, is a dual-stage search method that is based from the idea of solving an approximate solution at a short amount of time and then later improving the solution. An approximate solution is derived using a large k value (i.e., rough table motion) in a large search space (i.e. large $\theta^{0,min}$ and $\theta^{0,max}$ values) and afterwards a solution is improved by using a small k value(i.e., smooth table motion) at a narrow search space (i.e. small $\theta^{0,min}$ and $\theta^{0,max}$ values) near the approximate solution.

One problem with RS is on selecting k values for the two stages. With a large k value at the approximation stage, the calculation time is reduced but may result to a low-quality solution. At the improvement stage, although using a small k value can improve a solution; but the vertices which are parts of a good solution may have already been left out at the approximate stage. In the implementation of RS, GM and DM are combined: (1) *RS_GM* and (2) *RS_DM*. In both combinations, the GM is utilized at the approximation stage due to its speed in finding a solution. At the improvement stage, the GM is utilized in RS_GM while DM is used in RS_DM.

An exhaustive search method to solve the shortest path has to consider $(l+1)^N{}_{max}$ path which makes it impractical. The GM searches at most Nx(l_{max}+1) paths while DM searches at most Nx(l_{max}+1)^2 paths. The DM has relatively longer calculation time than GM but due to its method of finding solution, DM is expected to obtain a solution better than GM.

C. Collision detection

The A_r, A_t, and A_o are modeled as approximate boxes called oriented bounding boxes (OBBs) [19], which is practical for modeling their rectangular shapes. The OBBs are defined by its size and orientation. To detect collision is to check the overlaps of two OBBs and a naive way of doing it is to perform 144 edge-face tests (i.e., 12 edges of one box times 12 edges of the other box). The collision detection is done through the *separating axis theorem* which states that two disjoint convex polytopes in 3D-space can always be separated by a plane which is parallel to the face of either polytope or orthogonal to an edge from each polytope. By employing this theorem, there are 15 potential separating axes for two OBBs (i.e., 3 faces from one box, 3 faces from the other box, and 9 pair-wise combinations of edges). A test of being disjointed is to project the two OBBs onto a potential separating axis. If their projections do not overlap, then OBBs are not colliding. If their projections do overlap, other potential separating axes are subsequently used to check if they again overlap or not. In short, the worst case scenario for detecting collision is to perform 15 tests when two OBBs are overlapping. A detailed discussion is given in [19].

V. Simulation And Discussion

This section describes the simulation done as well as the result and discussion.

A. Parameters

A simulation with different parameter settings is done to assess empirically the performance of various search methods. Table I shows the parameters used in the experiment. The velocities, v^1-v^6, correspond to joint velocities of robot arm from its base up to the end-effector.

There are six A_o considered; the goal positions are different

TABLE I
Simulation Settings

Parameter	Notation	Setting
A_r *base position*(mm)		x=-540, y=0, z=300
Initial end-effector position	p_0	x=0, y=0, z= 700
Joint velocity (deg/s)	v^1~v^6	250, 250, 250, 410, 410, 660
A_t *position:*		x=0, y=0, z= 0
A_t *velocity:* (deg/s)	v^0	360^0
A_t *size:* lxwxh (mm)		400x400x400
A_o *size :* lxwxh (mm)		400x400x150
A_o *type*	Obj_1, Obj_2	N= 25
	Obj_3, Obj_4	N= 50
	Obj_5, Obj_6	N= 75

TABLE II
SETTINGS FOR VARIOUS SEARCH METHODS

Search method	Table rotation step	Table rotation limits
GM	k=5°	$\theta^{0,max}$=180°, $\theta^{0,min}$=-180°
DM	k=5°	$\theta^{0,max}$=180°, $\theta^{0,min}$=-180°
RS_GM1	k_1= 30°, k_2= 5°	$\theta_1^{0,max}$= 180°, $\theta_1^{0,min}$=-180°, $\theta_2^{0,max}$= 90°, $\theta_2^{0,min}$ =-90°
RS_DM1	k_1= 30°, k_2= 5°	$\theta_1^{0,max}$= 180°, $\theta_1^{0,min}$=-180°, $\theta_2^{0,max}$= 90°, $\theta_2^{0,min}$ =-90°
RS_GM2	k_1= 60°, k_2= 5°	$\theta_1^{0,max}$= 180°, $\theta_1^{0,min}$=-180°, $\theta_2^{0,max}$= 90°, $\theta_2^{0,min}$ =-90°
RS_DM2	k_1= 60°, k_2= 5°	$\theta_1^{0,max}$= 180°, $\theta_1^{0,min}$=-180°, $\theta_2^{0,max}$= 90°, $\theta_2^{0,min}$ =-90°

Note: k_1 is the rotation step of a table at the approximation stage while k_2 is the rotation step of a table at the improvement stage in RS method.

in A_o with equal number of goals. The distribution of goals on the object is randomly generated. The sizes of clusters are uniform but clusters with non-uniform sizes can be dealt with by the clustering method employed in this study. In addition, the number of clusters is 5 for all objects. Since the size of object is fixed, the distribution of points in each cluster becomes dense as N is varied from 25 to 75.

In the given positions of A_r and A_t, some of the goals cannot be reached by end-effector due to collision; hence, it is impractical to compare the result of this simulation, which has coordinated motion control of A_r and A_t, to a setup with static A_t or to a setup which does not consider collision. In finding the shortest collision-free path, the settings of various search methods are shown in Table II. There are several possible settings for RS method by choosing values for k, $\theta^{0,max}$, and $\theta^{0,min}$; although only two settings are chosen as references that give good solutions with no substantial amount of time incurred in calculation.

B. Results and Discussion

The results of the simulation for Obj_1 to Obj_6 are shown in Figure 6 and Figure 7. Fig. 6 shows the derived best working time for every search method while Fig. 7 shows the calculation time spent. For all objects, as shown in Fig. 6, the DM consistently obtains shorter working time than GM. However, as shown in Fig. 7, the GM has shorter calculation time than DM, has the longest calculation time for all objects. In particular, for Obj_1 and Obj_2 where N=25, the improvement of solution (i.e. reduction in the derived working time) in DM against GM are 10% and 16%, respectively whereas the calculation times in DM take 30 and 50 times longer than GM. Overall, as shown in Fig. 7, the GM obtains a solution at a shorter design time. However, the solution obtained using GM has longer working time due to its short-sighted search method, which selects configuration based only on the possible configurations of the next goal. In effect, the GM misses out vertices in the search graph that may be part of a good, if not, the optimal solution. In contrast, the DM considers the incremental updates in every stage and then selects whichever has the least cost.

The performance of RS is determined by the method used in the improvement stage. As for RS_GM that uses the GM in the improvement stage: the RS_GM1 has better working time than GM for all objects while RS_GM2 has working time that is worse than GM in Obj_5 and Obj_6. This scenario can be explained by a larger k value used in RS_GM2 than in RS_GM1 which resulted to a longer working time. The increase in calculation time observed in RS_GM1 and RS_GM2 with respect to GM is expected: at the approximation stage of RS_GM1 and RS_GM2, the value of k_1 is large, which means that the number of vertices in a search graph is fewer that resulted to quicker calculation time; on the other hand, at the improvement stage, due to small k_2 value and wide rotation limits, the calculation time still takes a long calculation time. Choosing the value of the rotation limit is a trade off between the working time and the calculation time: a narrow rotation limit may results into a poor solution (i.e. longer working time); on the other hand, a wider rotation limit may results into a longer calculation time yet a better solution.

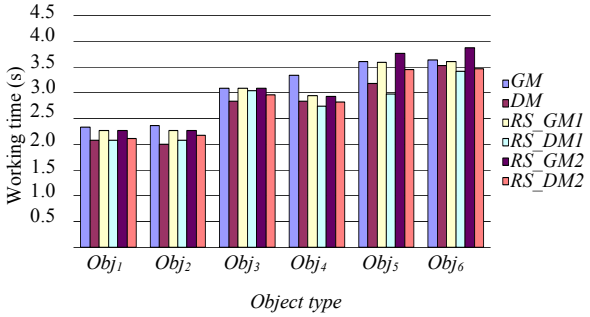


Fig. 6 A comparison of the derived working time of various search methods in Obj_1 to Obj_6.

The RS_DM1 and RS_DM2 take advantage of the speed achieved by employing GM at the approximation stage and of deriving a good solution by using DM at the improvement stage. The ideal case for RS_DM1 and RS_DM2 is when the search space of DM encompassed the vertices that result into a short working time. Note that in RS_DM1 and RS_DM2, the search space of DM is determined after a solution of GM is found at the approximation stage. For RS_DM1 and RS_DM2, this case holds true for Obj_1, Obj_2, Obj_4, Obj_5, and Obj_6 which have solutions comparable to, if not, better than, DM. In all object, RS_DM1 and RS_DM2 has better solution that GM. In terms of calculation time, RS_DM1 and RS_DM2 are faster than DM by a factor of 4. Comparing RS_DM1 and RS_DM2, the former obtains better solution than the latter, simply because the k_1 value of the former is smaller - meaning, higher search resolution at approximation stage, than that of the latter.

Note that RS_DM1 and RS_DM2 have better solution (i.e. shorter working time) than DM in Obj_1, Obj_4, and Obj_5. This can be explained by the goal arrangement, which uses 2-opt exchange. The DM takes a long calculation time just to find a solution for one goal order; whereas the RS_DM1 and RS_DM2 have short calculation times such that other possible orders are searched before the calculation time constraint is reached.

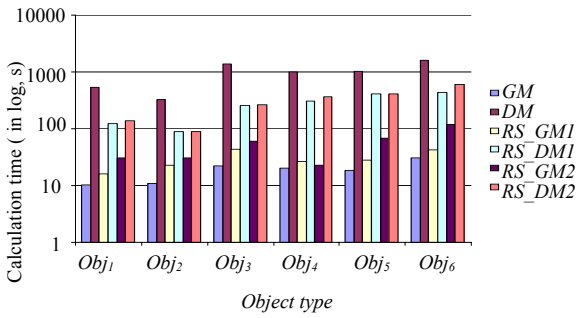


Fig. 7 A comparative performance of various search methods in Obj_1~Obj_6.

Based from the above results, the following can be concluded about the search methods:

- The GM gives a solution at a short amount of time which is practical for finding solutions for highly time-constrained optimizations.
- The DM provides a high-quality solution with a large demand of calculation time.
- The RS, particularly RS_DM, can potentially give a better solution than that of GM and DM, with short calculation time. The performance of RS is however dependent on the parameter settings of the approximation and improvement stages.

In regards to the actual motion of robot arm and table, the joint values selected by the DM, including RS_DM1 and RS_DM2, allows few joint motions for robot arm. In Obj_1, for instance, (See accompanying video), the robot arm links are stretched out utilizing, most of the time, the joints near the end-effector, which are faster than the joints near the robot arm base. In effect, the robot arm moves to only few regions in order to reach all goals. While in GM, several robot arm motions are spent on changing the location of end-effector making the robot arm links to be initially stretched out and then bent inward.

VI. Conclusion

In this study, a minimum-time and collision-free motion coordination with arrangement of several goals is described. A 6-DOF robot arm with a single-axis positioning table is utilized as the setup. The goal arrangement is determined through a TSP with clustering based on the geometric shape of object. The motion of robot arm and table is based on a straight-line path in the configuration space of robot arm and table. No modification of this path is done to avoid collision; instead, the redundancy of the system is utilized to select a configuration of robot arm and table with a collision-free path.

In addition, a two-stage search method is proposed and is shown effective in deriving minimum-time collision-free configurations of robot arm and table. It employs first the greedy nearest neighbor method to find an approximate solution and then the Dijkstra method to improve the solution. With an imposed calculation time limit, a simulation is done on objects with various numbers of goals and with different goal positions.

For the purpose of this study, a simple 2-opt exchange is utilized to derive the order of clusters. Other algorithms may be used such as simulated annealing, which can probably obtain a better solution, but may take considerable amount of calculation time.

The study can be extended by changing the base placement of robot arm relative to the table position. Furthermore, adding more DOF in the positioning table can be a good venue for future research work.

References

[1] J.-C. Latombe, *Robot Motion Planning*. Norwell, MA: Kluwer Academic Publishers, 1991.

[2] D. Hsu, J.-C. Latombe, and S. Sorkin, "Placing a robot manipulator amid obstacles for optimized execution," in *IEEE Intl. Symp. on assembly and task planning*, pp. 280-285, 1999.

[3] J.E. Hopcroft, J. T. Schwartz, and M. Sharir, "On the complexity of motion planning for multiple independent objectd: PSPACE-hardness of the *warehouseman's problem*," *Intl. J. of Robotics Research*, vol. 3., no. 4, pp. 76-88, 1984.

[4] S. Ahmad and S. Luo, "Coordinated Motion Control of Multiple Robotics Devices for Welding and Redundancy Coordination through Constrained Optimization in Cartesian Space", *IEEE Trans. On Robotics and Automation*, vol. 5, no. 4, pp. 409-417, 1989.

[5] L. Wu, K. Cui and S.B. Chen, "Redundancy coordination of multiple robotic devices for welding through genetic algorithm," *Robotica*, vol. 18, pp. 669-676, 1999.

[6] S. Akella and S. Hutchinson, "Coordinating the motions of multiple robots with specified trajectories," in *Intl. Conf. on Robotics and Automation*, pp. 624-631, 2002.

[7] B. Cao, G. I. Dodds and G. Irwin, "Time-suboptimal inspection task sequence planning for two cooperative robot arms using mixed optimization algorithms," in *IEEE Intl. Conf. on Robotics and Automation*, pp. 2103-2107,1997.

[8] M. Saha, T. Roughgarden, and J.-C. Latombe. "Planning tours of robotic arms among partitioned goals," *The Intl. J. of Robotics Research*, vol. 25, no. 3, pp. 207-224, 2006.

[9] M. Bonert, L.H. Shu, B. Benhabib, Motion planning for multirobot assembly systems. *ASME Design Engineering Technical Conference*, Sept. 1999.

[10] Y. Edan, T. Flash, et. al. " Near-minimum-time task planning for fruit picking robots," Trans. on Automation and Control, vol. 7, no. 1, pp. 48-56, February 1991.

[11] T. L. DeFazio and D. E. Whitney, Simplified generation of all mechanical sequences," IEEE Journal of Robotics and Automation, vol. RA-3, no. 6, 1987.

[12] S.Kirkpatrick, C. D.Gelatt, and M. P.Vecchi, "Optimization by simulated annealing," Science, 220, pp. 671-680, 1983.

[13] S. Lin and B.W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem", Ops. Res. 21, pp. 498-516, 1973.

[14] K. Helsgaun,"An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic", *European J. of Operations Research,* vol. 126, no. 1, pp. 106-130, 2000.

[15] A. Zhou, L. Kang, and Z. Yan, "Solving dynamic TSP with evolutionary approach in real time," in *Congress on Evolutionary Computation* , pp. 951-957,2003.

[16] C. S. Helvig, G. Robins, and A. Zelikovsky, "Moving-Target TSP and Related Problems," Journal of Algorithms, vol. 49, pp 153-174, 2003.

[17] Y. Asahiro, T. Horiyama, K. Makino, et. al. "How to collect balls moving in the Euclidian plane," *Electronic Notes in Theoretical Computer Science*, vol. 91, pp. 229-245, 2004.

[18] K. Kobayashi, " Introducing a clustering technique into RNN for solving large scale TSP," in *Intl. Conf. on Artificial Neural Network*, vol. 2, pp. 935-940, 1998.

[19] S. Gottschalk, "OBB-tree: A hierarchical structure for rapid interference detection," *Comp. Graphics*, pp. 171-180, 1996.