

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900

Open access books available

186,000

International authors and editors

200M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Palletizing Simulator Using Optimized Pattern and Trajectory Generation Algorithm

SungJin Lim, SeungNam Yu, ChangSoo Han and MaingKyu Kang
Hanyang University
South Korea

1. Introduction

Collision avoidance and robot path planning problems have emerged as a potential domain of robotics research of late because of their indispensable requirements in the field of manufacturing vis-à-vis material handling, such as picking and placing an object and loading/unloading a component to/from a machine or storage bins. This chapter focuses on palletization, a form of unitization in which a uniform load is stacked on a wooden pallet using a predetermined case pattern sequence and a given number of layers. In many kinds of proposed C-space construction approaches, several algorithms deal with the boundary of the C-obstacle analytically. Lozano Perez proposed the fundamentals of the C-space approach. When both the robot and obstacles have the shape of convex polygons, the C-obstacle boundary for an n-DOF manipulator is approximated by sets of n-1 dimensional slice planes, which are made from a one-dimensional slice plane. C. Zhao and his colleague proposed an algorithm to describe the C-obstacle as a set of parametric equations formulated from the mapping of the boundaries of the obstacles in a workspace. They use inverse pseudo kinematics to convert the obstacles in a workspace into a C-space. Debanik Roy studied path planning algorithms and their heuristics using the concept of visibility graph, and he presented an overview of the case study of robot path planning in an industrial environment in real time. Xiaojun and his colleague proposed a two-phase approach for C-obstacle construction and the collision detection of manipulators. This method is applicable to manipulators with various types of kinematic structures and geometric shapes. M. Pettersson and his coworkers proposed trajectory optimization method considering fatigue and thermal load of real robot. They also referred that the proposed method could be directly adapted to palletizing system. The issues of these papers, however, are for the operation of real industrial robot, and it is a part of entire palletizing system. Many other latest researches solely oriented towards path planning or modified apparatuses to improve the specified handling task have also been conducted. Studies on the total robot palletizing system, however, which integrates loading pattern optimization, robot OLP simulation, and path optimization, have yet to be systematically conducted.

This study was dedicated to the development of OLP Simulation S/W for a robot palletizing system (Non-vision system), which means that this study prioritized the reflection of the

characteristics of the palletizing task to realize the path generation algorithm. Undoubtedly, the shape of stacked boxes is changed after the unit-stacking step and palletizing robot encounters shape-changed obstacles in every step. This study pays attention to this characteristic of the palletizing task and proposes a simple and efficient methodology to generate a proper path generation algorithm for palletizing robots.

For the practical use of the proposed palletizing OLP Simulation S/W, a user-defined task design is possible in this simulator, and the user can store his own data. This application deals with rectangular boxes only as they make up the vast majority of objects involved in palletizing applications.

This chapter is organized as follows. In section 2, the layout of the task space and its coordinates, which are used consistently throughout this study, are defined. Subsequently, the proposed Fast algorithm and its application are introduced. Section 3 deals with the newly designed 3D robot simulator and its combination, with the application of the Fast algorithm. Section 4 presents the simulation of general C-space and A* algorithm for specified path generation using proposed simulator described in the previous section. Section 5 deals with the simulation of modified C-space to fit it into the real robot system. Based on the approaches of Section 5, Sections 6 is devoted to the newly proposed path generation method (overlap method) as well as to the simulation and its results. Finally section 7 concludes the chapter.

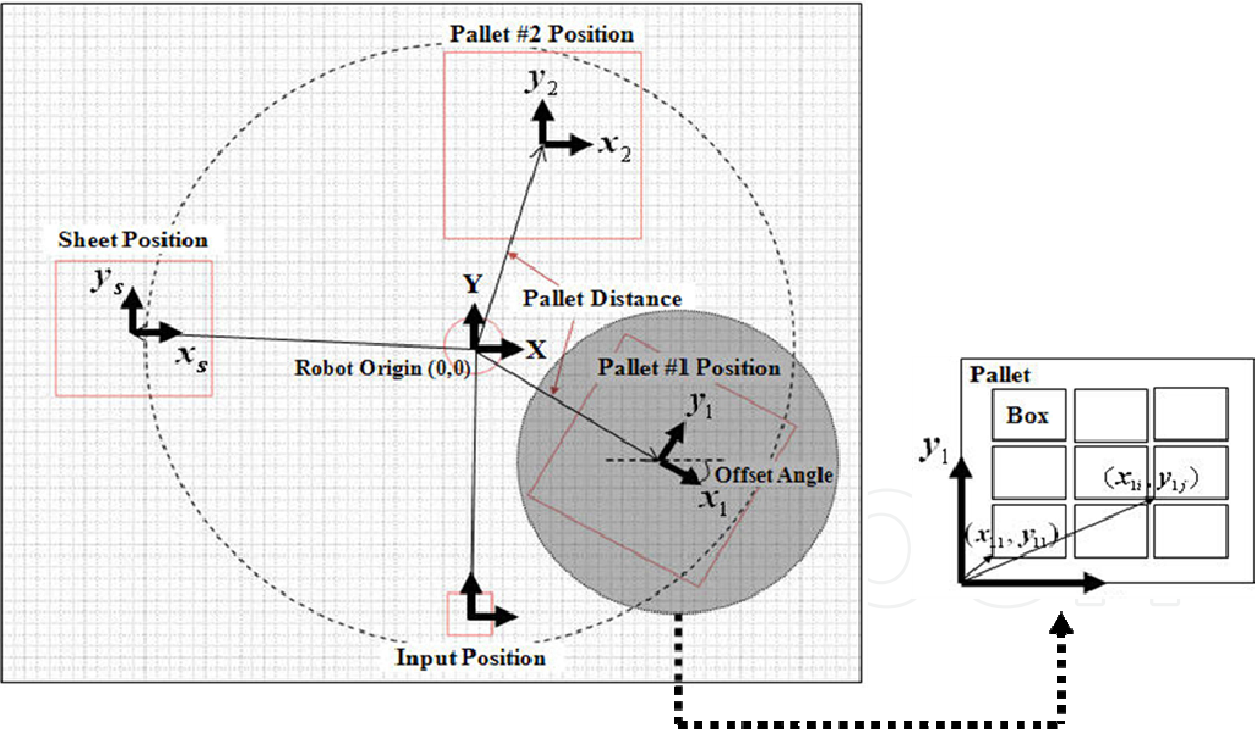


Fig. 1. Definition of task space

2. Fast Algorithm

2.1 Definition of the System Layout

As stated above, this application is a combination of an optimized pallet pattern generation algorithm, an industrial robot simulator, and a modified trajectory optimization algorithm.

To integrate these modules and to define the positions of the boxes, the robot, and its peripherals, the system layout and its coordinates have to be defined.

Fig. 1. describes the system layout and coordinates of the proposed palletizing OLP S/W. The robot is located in an origin of the total system, and other components (pallet, sheet, input facility, etc.) are expressed using this coordinate. Stacked boxes belong to the coordinate of the corresponding pallet. Consequently, the position and rotated angle of the boxes are expressed by the robot origin and its coordinate using relative coordination. By using this coordinate system and layout, the following chapter presents a pair of pattern

2.2 Steudel’s Heuristic Algorithm

The objective of the pallet loading problem is to maximize the number of products that are loaded onto a pallet used for the transportation and storage of products. The distribution and storage costs of the product can be reduced by increasing its pallet utilization. The Fast algorithm presented in this study is an improved version of the 4-block pattern heuristic algorithm proposed by Steudel. The typical pattern of Steudel’s heuristic algorithm is presented in Fig. 2. This heuristic finds the four-block pattern, in which each block is in a homogeneous pattern with the same box orientation. This heuristic consists of two phases. First, an initial solution is made with the combination of L_i and W_i , which maximizes the utilization of all four pallet edges.

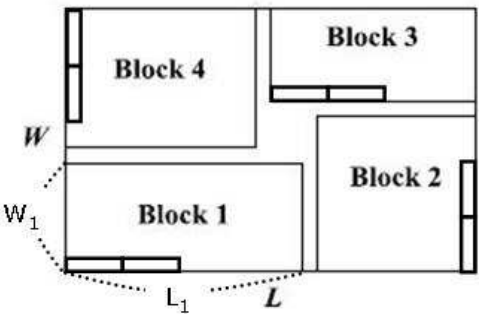


Fig. 2. Steudel’s heuristic algorithm

Dynamic programming is applied to find the combination, and the initial solution has one of the four patterns shown in Fig. 3. The initial solution has a central hole that is sufficiently large to load more than one box in the case of P1 and P3, or an infeasible pattern, such as P2 and P4, due to the overlapped area.

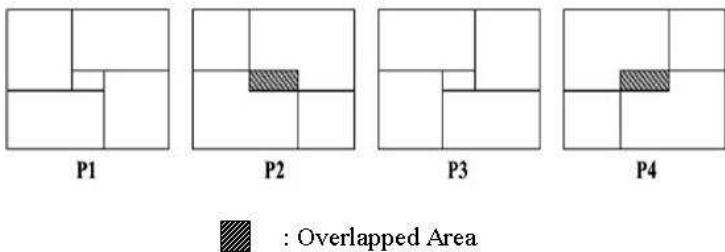


Fig. 3. Four patterns of the initial solution

This case involves the second phase. In the second phase, Treatment 1 fixes (L_3, W_3) and (L_4, W_4) and resizes L_1 and L_2 , and Treatment 2 fixes (L_1, W_1) and (L_4, W_4) and resizes W_2 and W_3 . Then, the first and second methods are compared and the better solution is chosen (Fig. 4.).

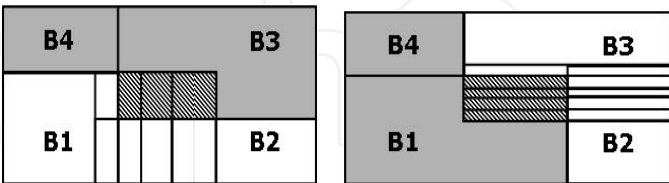


Fig. 4. Treatment of Steudel’s algorithm

2.3 The Fast Algorithm

2.3.1 Definition

The Fast algorithm has similar processes with which to generate the initial four solution patterns. In addition, Treatment 3 is adapted to apply the heuristic recursively to the central hole in the following three methods so as to remove the overlapped area (Fig. 5.).

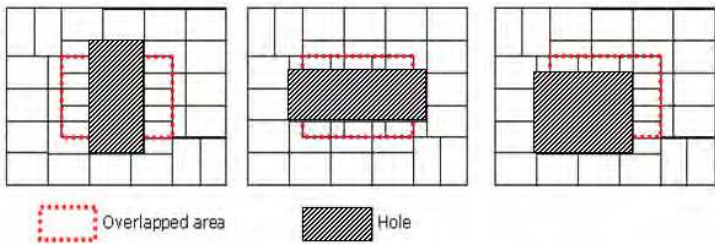


Fig. 5. Treatment of the Fast algorithm

- (1) In the first method, the boxes are cut by the two horizontal edges of the overlapped area.
- (2) In the second method, the boxes are cut by the two vertical edges.
- (3) In the third method, the boxes are cut by the left vertical edge and the lower horizontal edge.

2.3.2 Schematic Diagram of the Fast Algorithm

As this algorithm does not consider all block sizes, it has a more rapid calculation time. The initial solutions of the first phase find the combination rather than using DP, and define the four parameters (Fig. 6.).

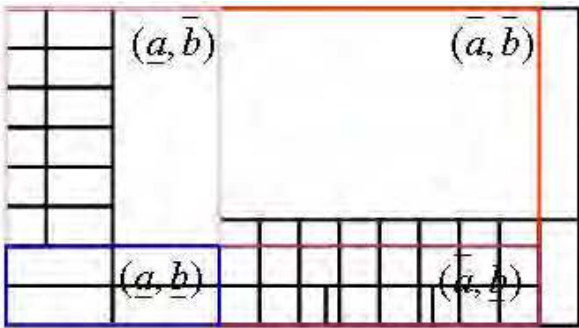


Fig. 6. Parameters of the Fast algorithm

- \bar{a} : When maximizing the length of the block and disposing the boxes lengthwise, the maximal possible number of boxes = $5l$.
- \underline{a} : When maximizing the length of the block and disposing the boxes lengthwise, the minimal possible number of boxes = $2l$.
- \bar{b} : When maximizing the width of the block and disposing the boxes lengthwise, the maximal possible number of boxes = $8w$.
- \underline{b} : When maximizing the width of the block and disposing the boxes lengthwise, the minimal possible number of boxes = $2w$.

In the first phase, (L_1, W_1) , such as (\bar{a}, \bar{b}) , (\bar{a}, \underline{b}) , (\underline{a}, \bar{b}) , and $(\underline{a}, \underline{b})$, are combined, and (L_1, W_1) , the width and length of the other blocks, can be determined.

$$(L_2, W_2) = (L_4, W_4) = \left(\left\lceil \frac{L - L_1}{w} \right\rceil w, \left\lceil \frac{W - W_1}{l} \right\rceil l \right) \tag{1}$$

$$(L_3, W_3) = (L_1, W_1) \tag{2}$$

After obtaining the four initial solutions in the first phase, these solutions are redefined by applying the three treatments in the second phase.

Procedure FindBlockLayout(L,W,depth)
bestSolution \leftarrow 0
Find \bar{a} , \underline{a} , \bar{b} , and \underline{b}
Make four initial Solutions.

S_i (i=1,2,3, and 4), using them

```

For all  $S_i$  ( $i=1,2,3$ , and 4)
 $S'_i \leftarrow$  Number of boxes after the first treatment
 $S''_i \leftarrow$  Number of boxes after the second treatment
If  $\max\{S'_i, S''_i\} > bestSolution$ , then
     $bestSolution \leftarrow \max\{S'_i, S''_i\}$ 
End If
If  $depth >> MaxDepth$  then
    Return  $bestSolution$ 
End If
For all central holes
     $S'''_i \leftarrow$  Number of boxes in the area
    excluding central hole
    Let  $(L_h, W_h)$  = size of central hole
     $S'''_i \leftarrow S'''_i + FindBlockLayout(L_h, W_h, depth+1)$ 
    If  $S'''_i > bestSolution$ , then
         $bestSolution \leftarrow S'''_i$ 
    End If
End For
End For
Return  $bestSolution$ 
End Procedure

Algorithm  $SolvePLP(L, W, l, w)$ 
 $bestSolution \leftarrow 0$ 
For all  $(L_1, W_1, l, w)$  that satisfy the inequality (2) or (3) and  $L_1 \in C_1, W_1 \in C_w$ 
    Calculate all size of the five blocks
    Call  $FindBlockLayout(L_1, W_1, 0)$  for all
         $i=1,2,3,4$  and 5
    If  $\sum_{i=1}^5 n(B_i) > bestSolution$  then
         $bestSolution \leftarrow \sum_{i=1}^5 n(B_i)$ 
    End If
End For
End Algorithm

```

Fig. 7. The Fast algorithm

2.3.3 Computing Experience

The proposed algorithm was implemented in Visual C++ 6.0 and was compiled with the maximized-speed option. This algorithm test generated a 2D pattern of boxes and its calculation speed. As a hypothesis, the load balancing of a box and its stability were not considered.

(L,W,l,w)	Amount of boxes loaded
(1000,1000,205,159)	30
(1000,1000,200,150)	33
(22,16,5,3)	23
(30,22,7,4)	23
(14,10,3,2)	23
(53,51,9,7)	42
(34,23,5,4)	38
(87,47,7,6)	97
(1200,800,176,135)	38

(L: Length of Pallet, W: Width of Pallet, l: Length of Box, w: Width of Box)
Table 1. Test results of The Fast Algorithm (2D)

The above results were acquired by a computer with a K6-350-MHz CPU and 64MB RAM. All problems were calculated within 1 s and resulted in optimal solution. To use this algorithm practically, one dimension of height is applied additionally, and the 3D pallet loading simulator is realized, as shown in Fig. 8.

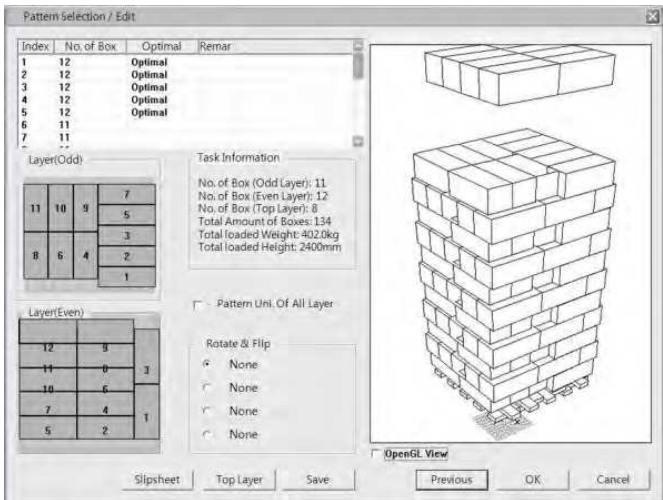


Fig. 8. pattern generation S/W

3. Development of the 3D Robot Simulator

Several methods have been introduced to make industrial robots perform the palletizing task. The first involved an online tutorial for the robot, which used a teach pendant to enable the robot to mimic and memorize the worker’s motion. The second method is an offline method that generates task data using a computer, and that downloads it onto the robot controller. This chapter focused on offline task generation and simulation using a

robot simulator. In this phase, the 3D robot simulator is presented based on the dimensional data of a real target machine, the HX300, which is a six-axis industrial robot of Hyundai Heavy Industrial Co. This robot model was realized by a commercial CAD modeler, and the GUI was developed using OpenGL® and MFC of Microsoft Visual C++®. To solve and analyze the forward and inverse kinematics equations, a general D-H parameter and the Lagrangian dynamic equation were used. With this simulator, it was possible to compute and display the joint torque, angle, and angular acceleration simultaneously. Fig. 9. shows the realized 3D robot simulator that was developed using Microsoft Visual Studio® and OpenGL®. It was possible to functionally calculate the velocity and acceleration of the gripper and to simulate the user-defined motion. The coordinates, which are generated by the pattern of loaded boxes on the pallet and the initial position of the box coming through an in-feeder, are passed to the simulator, and using these coordinates, it was possible to simulate the specified motion.



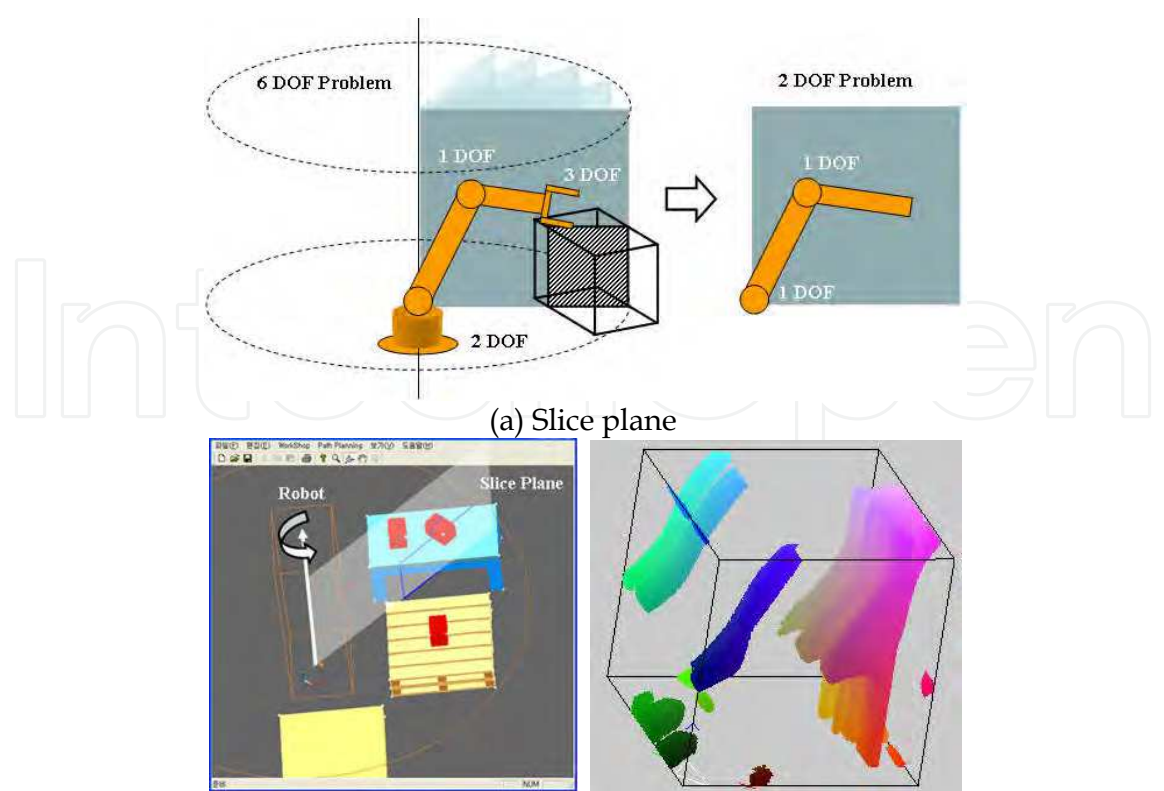
Fig. 9. Robot simulator for a palletizing task

4. C-Space and A* Algorithm for Trajectory Generation

4.1 C-Space Mapping of Obstacles

The palletizing task is generally composed of several palletizing components. These are auxiliary but are nevertheless obstacles for the palletizing robot. The important part of this study was to find the optimal path, considering the obstacles; hence, the concept of C-space (Configuration Space) to solve this problem was applied. The configuration defined the variables that exactly express the position and direction of an object, and the C-space represented all of the spaces where configurations may be acquired. Using this concept, a coordinate for each configuration was defined. In this coordinate, each point that was approached by the robot gripper was expressed by joint angles (configuration, posture) of the palletizing robot.

Fig.10. shows an example of the generation of the configuration space. First, on the basis of the joint of the base frame, the imaginary plane was rotated 360 degrees like Fig.10.(a).



(b) Apply the slice plane to the workspace to generate the C-space.
Fig. 10. Obstacles expressed in C-space

Step	Task Layout	C-space	Enlarged Image	Elapsed Time (sec)
1				3.132
2				0.384

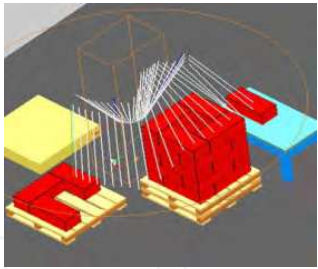
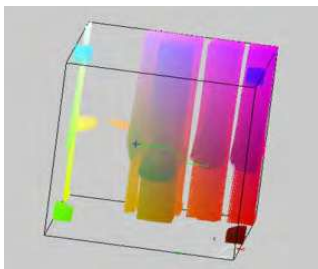
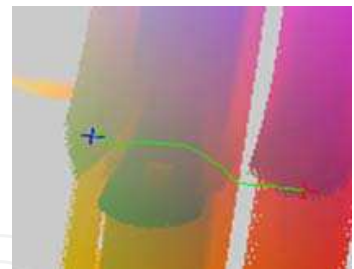
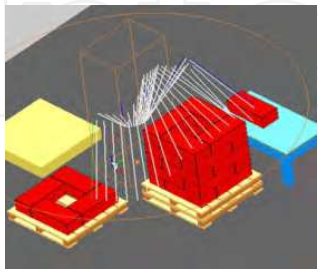
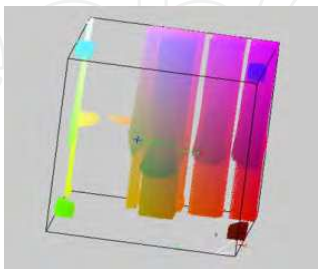
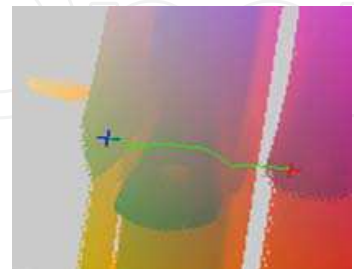
3				12.267
4				9.734

Table 2. Palletizing Task Simulation and Generation of Optimal Trajectory using A* Algorithm

In this progress plane, the objects surrounding the robot were scanned and the outline of a section was generated. The left side of the Fig.10.(b) describes the specified palletizing task layout. The outline, including its interior, could be considered an obstacle. In this study, the outline was acquired by using an end effector of the robot, and the free-movement and obstacle zones in the C-space were generated as shown at the right side of Fig.10. To help distinguish the 3D shape of C-space, various brightness and color are used. This figure is necessary to generate the optimal path using the A* algorithm described in the next chapter.

4.2 Application of the A* Algorithm for Trajectory Generation

The A* method is a thorough, robust planning technique that determines either the minimum cost path or whether no safe path exists. By exploring a map, the A* algorithm generates nodes that are used to recode the current status. This technique is used to find the optimal path between the gripping point (starting point) and the place’s down point (end point). The original A* technique is outlined below. To begin, a 2D rectangular grid was produced in which the cells were either safe or forbidden. The planning began at the starting point, and the cells adjacent to this cell were probed. On the basis of a cost function, the cell with the minimum cost was explored next. The cost function refers to the summation of costs, which required one to move from the starting node to the current node, and the “estimated” cost, which required one to move from the current node to the goal (a lineal distance). Based on this algorithm, palletizing simulation is performed in the 3D space and Table.2 is the results of the simulation.

5. Consideration of the Real Size of the Robot for Trajectory Generation

5.1 Modified Slice Plane (with horizontal thickness)

One of the disadvantages of the A* algorithm is the required computing time. The aforementioned approach considers the robot arm as a bar. Hence, the computing time load is relatively low. A real industrial robot, however, has an original volume, and these factors

have to be applied to the A* algorithm. The next step was to consider the real volume of the robot when it scans obstacles and generates C-obstacles.

To do this, the slice planes were redefined because it was assumed that the original slice plane had no thickness but that the modified slice plane had a thickness and that the factor that changed the scanning point of an obstacle of each angle was a group of both sides of the boundary of the modified slice plane (Fig. 11.). The thickness of the plane was determined individually by the thickness of the robot arm, including its gripper and load.

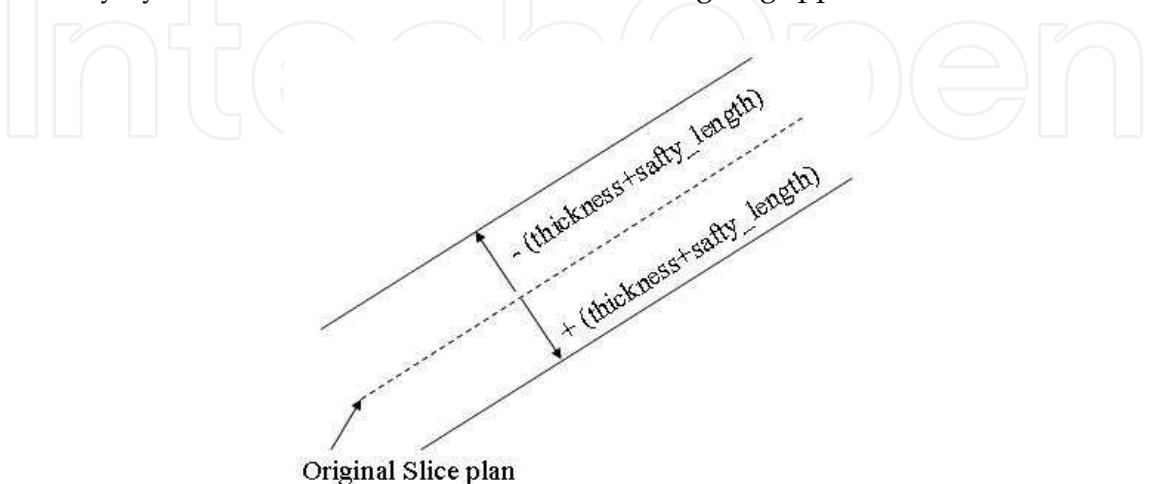


Fig. 11. Modified slice plane

5.2 Convex List and Graham’s Algorithm

Fig. 12. shows the scanning points that used the modified slice plane. The proposed system used factors of convex list points of objects and the sum of half of the thickness and a safe distance. As shown in Fig.12, the convex list was generated using the inside apexes of objects and intersection points. If the number of intersection points was less than two, the slice plane is regarded as meeting with one apex or edge.

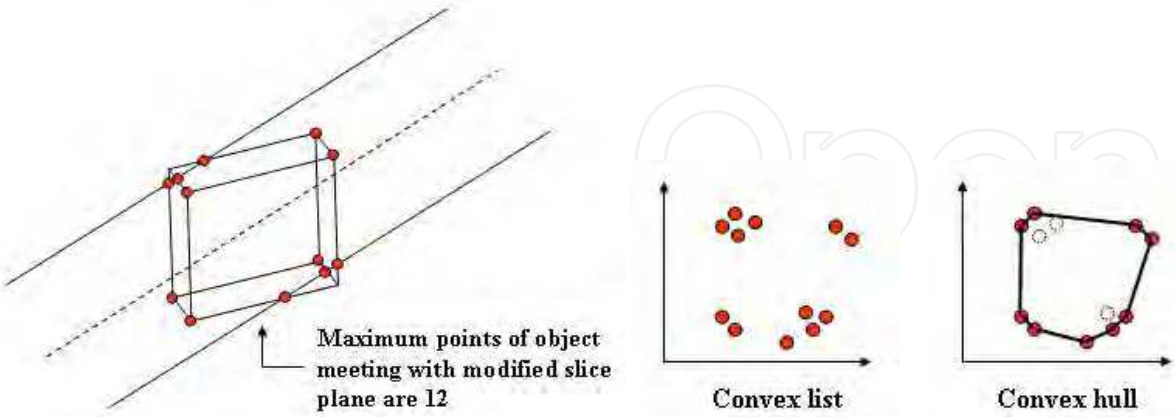


Fig. 12. Convex list generation

Finally, Graham’s algorithm was used to generate the convex hull. This hull was used as the new boundary of the object when the modified slice plane was applied.

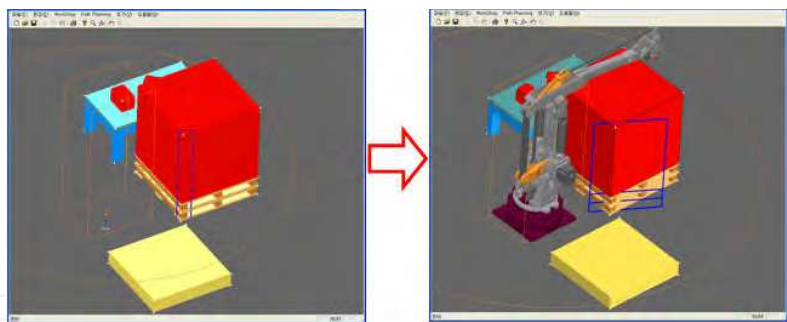
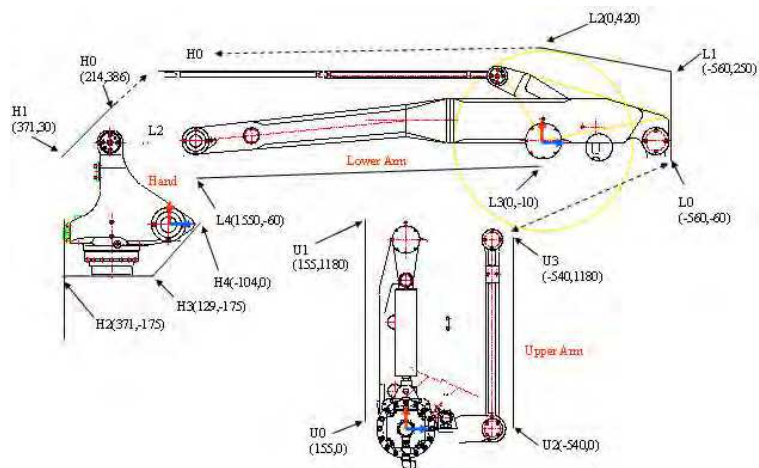


Fig. 13. Modified slice plane

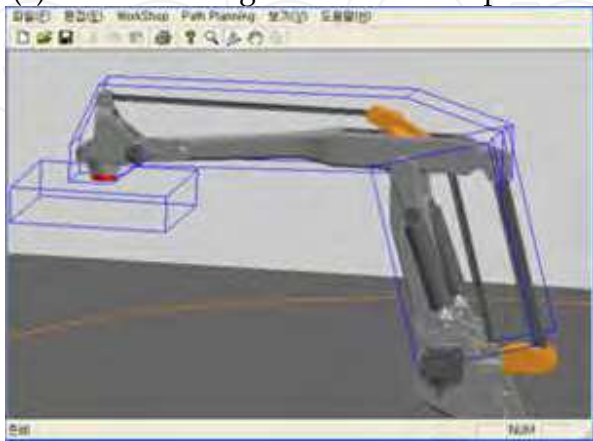
Fig. 13 describes the effect of the modified slice plane. As shown in the figure, the slice plane became larger.

5.3 Consideration of Vertical Thickness

The previous chapter showed the horizontal thickness of a real robot and proposed the modified slice plane that was used to generate the obstacle area of an object. As a next step, the vertical thickness of the robot was considered. Fig. 14. illustrates outlined margin of robot manipulator and its realization on the proposed simulator.



(a) Outlined Margin of Robot Manipulator



(b) Robot Model Realization

Fig. 14. Boundary line of the target robot system

These assumptions of the boundary of the gripper and its load (box) consider the total volume of the robot, including the robot arm, the gripper, and its load. Hence, when the modified slice plane (vertical thickness of the robot, gripper, and its load) is applied, the designed simulator is considered the vertical thickness of the robot arm, including the gripper and its load, simultaneously.

5.4 Consideration of the Performance of the A* Algorithm Using the Modified Slice Plane

If the robot body is a line, the computing time is very short and is therefore not an issue. When the modified slice plane was applied, however, the computing time was substantially increased. The possible explanation for this could be that the results were duplicated at the intersection points in each step and were added to the computation load of Graham's algorithm for the generation of the convex list. Fig. 15. shows an illustration of this simulation.

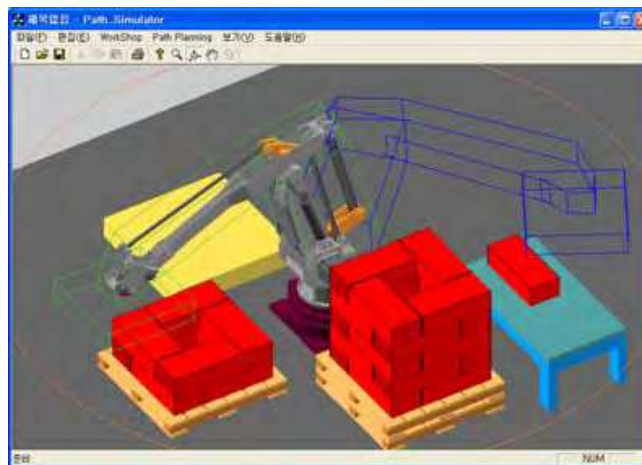


Fig. 15. Simulation of the A* algorithm using the modified slice plane

6. The Overlap Method to Generate the Palletizing Trajectory

The computing load is a critical problem in the area of software development. The purpose of this study, as described in the introduction, was to develop an OLP (offline programming) simulator specific to palletizing automation. As shown in Table 2, if the real size of a palletizing robot is considered to generate the optimized trajectory, an A* algorithm is a relatively expensive method. To use this algorithm, the C-space has to be generated, but this requires a large amount of computing load.

To focus on the characteristics of the palletizing task, a new strategy devoted to the generation of the set of boundaries (convex) of the obstacles was proposed. As shown in Fig. 16., the proposed method overlaps the scanned images of each box at one plane and obtains the outer line of the overlapped image. This method used the total traveling distance from the pickup point of the boxes to the place-down point via the outer line of the overlapped area.

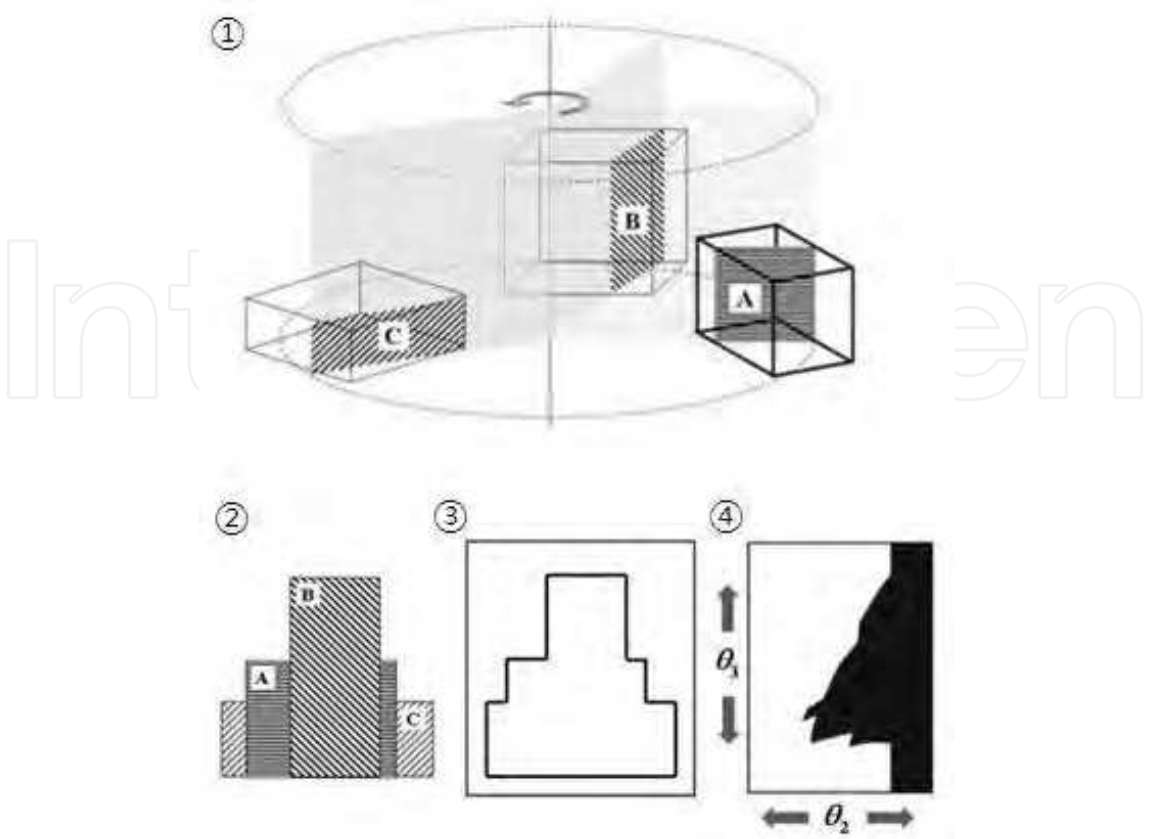


Fig. 16. Procedure of Overlap method

The following equation was used in this study to optimize this distance:

$$T_{opt} = A[abs\{(P_{via} - P_{pick-up})_{\theta_2}\} + abs\{(P_{place-down} - P_{via})_{\theta_2}\}] + B[abs\{(P_{via} - P_{pick-up})_{\theta_3}\} + abs\{(P_{place-down} - P_{via})_{\theta_3}\}] \tag{3}$$

where T is the distance that the robot must negotiate to palletize one box, which is the absolute summation of the distance from the pickup point to the outer line of the overlapped area and the distance from the outer line to the place-down point (Fig. 17.).

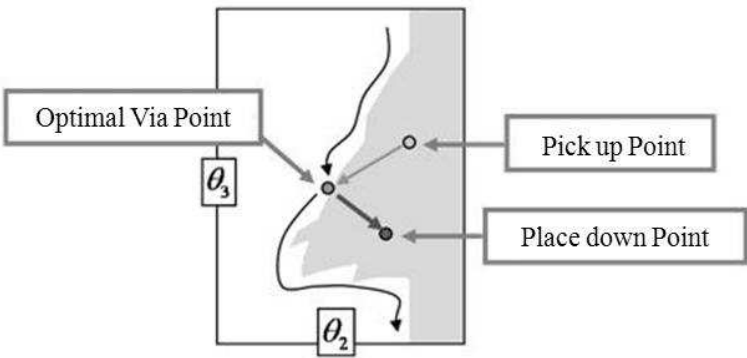


Fig. 17. Determination of θ_2 and θ_3 to generate trajectory

The robot path, however, is not composed of 3 points only (a place-down point, an optimal via point, and a place-down point). Therefore, this algorithm is expended to find an extra via point that would travel the whole path, from the start to the end point. (Fig.18.)

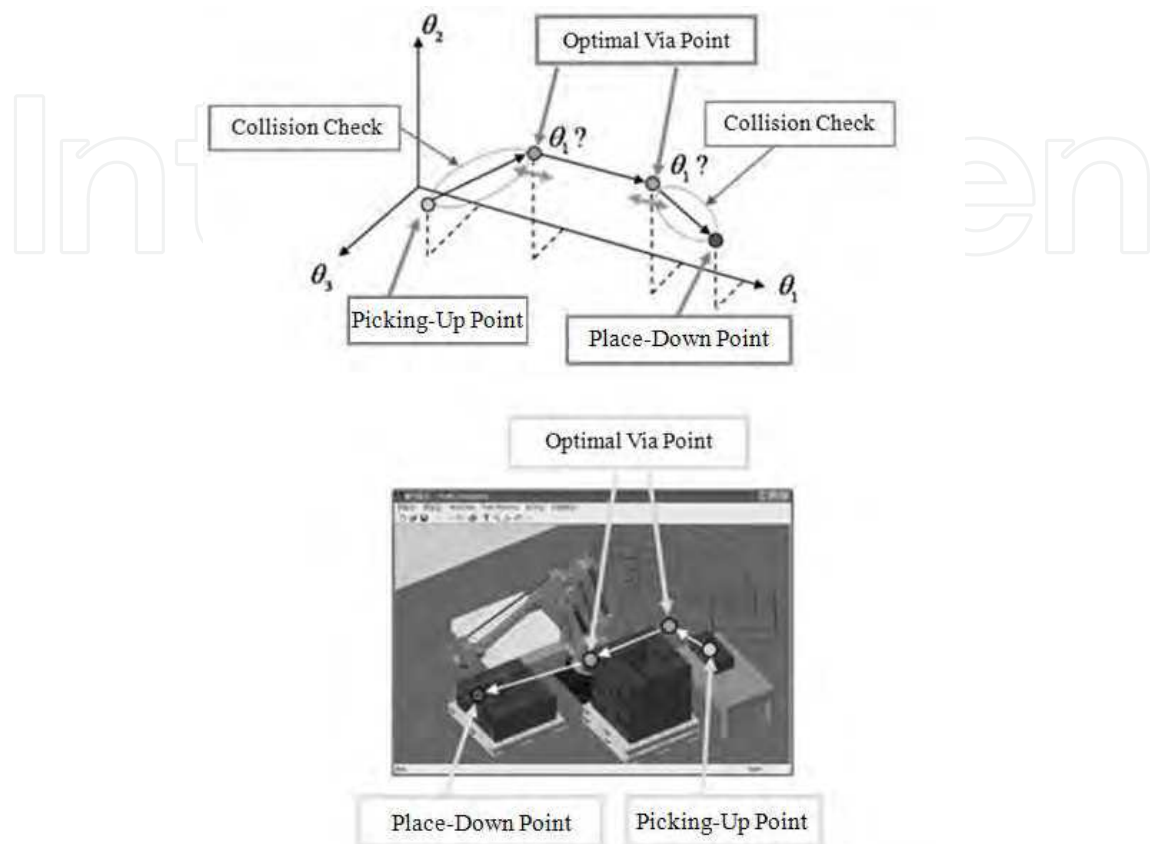


Fig. 18. Determination of θ_1 to generate the optimal via point

To do this, the aforementioned optimal 1 via point is used as a 1st optimal via point. If the gripper of the robot reaches this point, a collision between the gripper and the obstacle can be avoided by changing θ_1 . The definition of the collision or gap between the robot and the obstacle is decided beforehand (user-defined setting of the designed OLP S/W - "safe distance"). Through this treatment, the intermediate via points are decided. Finally, the total travel points are composed of [picking-up point] \rightarrow 1st optimal via point, [via($\theta_{11}, \theta_{12}, \theta_{13}$)] \rightarrow [...] \rightarrow nth via point, [via($\theta_{n1}, \theta_{n2}, \theta_{n3}$)] \rightarrow final optimal via point, [via($\theta_{f1}, \theta_{f2}, \theta_{f3}$)] \rightarrow [place-down point]. Here, i and j of θ_{ij} means ith generated via point of jth joint of robot manipulator.

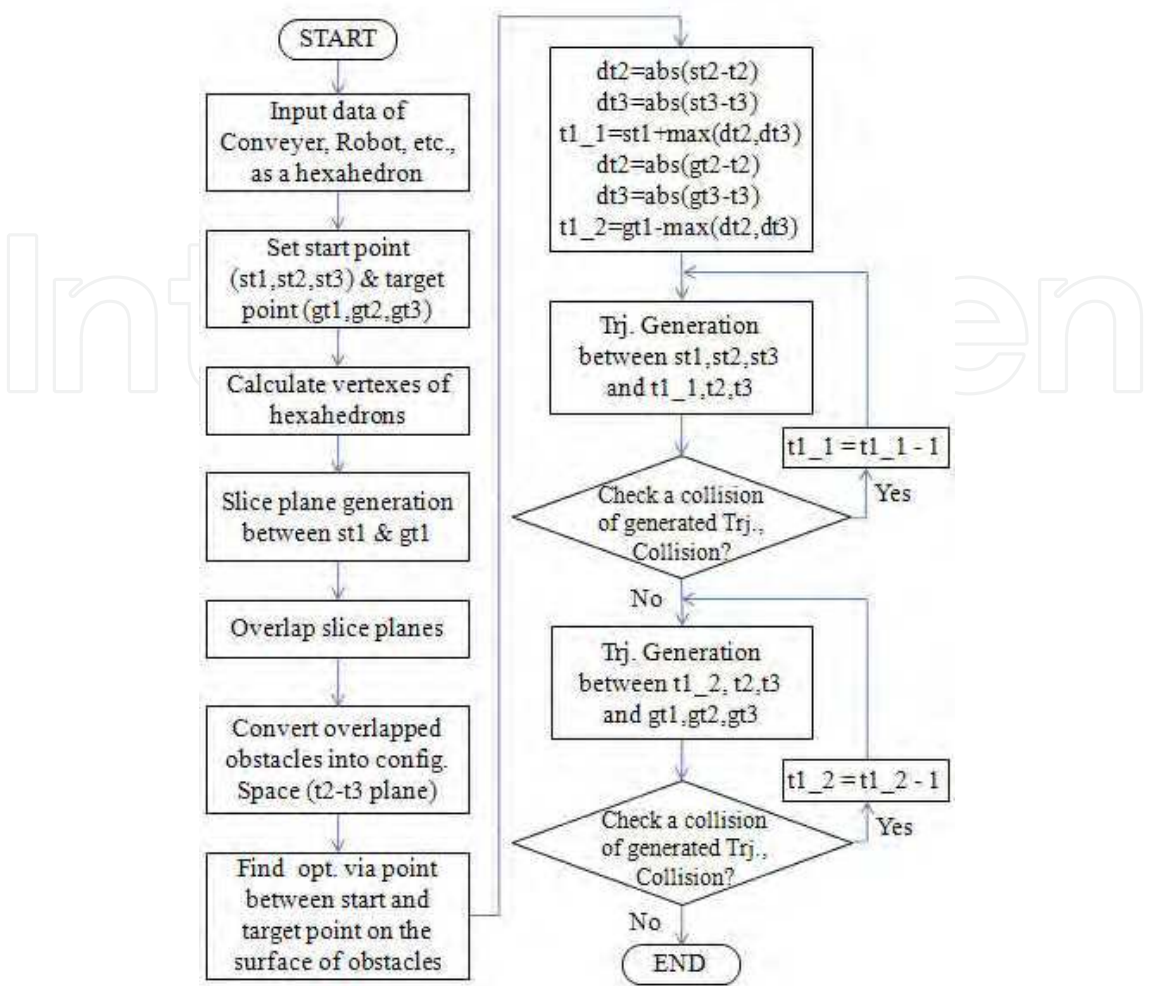


Fig. 19. Basic algorithm of the overlap method

- $st1, st2, st3$: , , of the starting point
- $gt1, gt2, gt3$: , , of a goal point
- $t2, t3$: , of an optimal path point
- $t1_{(i)}$: of an optimal path point (ith iteration)

This method deals with every surrounding obstacle of the robot in every unit step of the process (“unit step” means one cycle of pick-and-place task). As the shapes of the obstacles that surround the robot are changed at every step, this approach has the advantage of being able to calculate the pick-and-place path. Fig. 19. shows the detailed algorithm of the overlap method.

7. Conclusions and Considerations

To prove the efficiency of the proposed methodology, all type of trajectory generation method described in this chapter is simulated and its results are compared.

Step	Time (Sec.)		
	Line(A*)	Volume(A*)	Overlap Method
1	0.014051	2.380194	0.412106
2	0.01543	2.066007	0.427587
3	0.01558	1.857863	0.415959
4	0.017952	2.318304	0.43101
5	0.014286	2.265576	0.411975
6	0.016003	2.213547	0.422834
7	0.013669	1.360996	0.443131
8	0.014206	2.20403	0.440602
9	0.016555	1.328561	0.454023
10	0.015094	1.298407	0.438623
11	0.017387	1.660548	0.466195
12	0.01523	1.298854	0.4273
13	0.01889	0.886562	0.46002
14	0.01344	1.159015	0.428835
15	0.016348	1.091212	0.43728
16	0.01413	1.301314	0.424192
17	0.017107	0.386479	0.464915
18	0.016205	0.429178	0.47078
19	0.017836	0.361664	0.484123
20	0.014301	0.389356	0.439179
21	0.020215	0.278059	0.491373
22	0.014119	0.408476	0.441098
23	0.018028	0.288906	0.466881
24	0.014902	0.295928	0.439116

Table 3. Elapsed Time of Each Method (Box, 24ea)

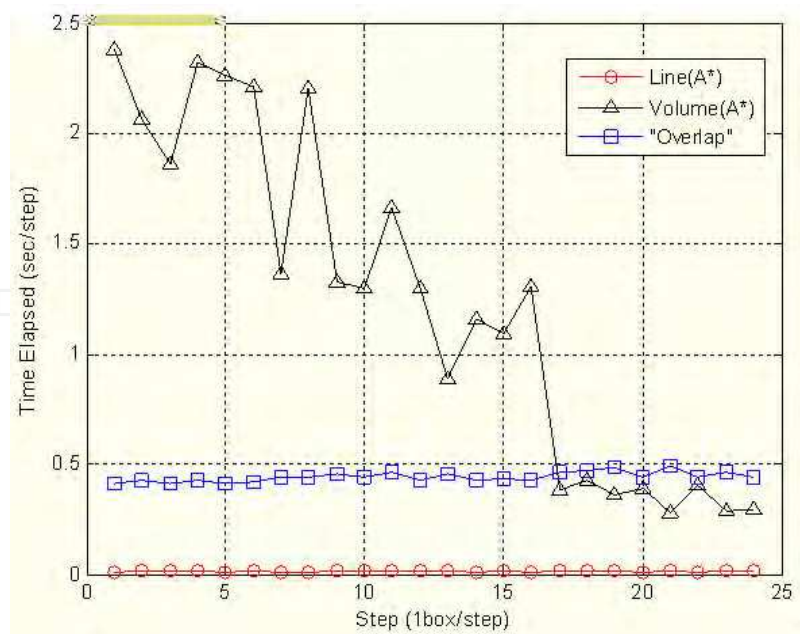


Fig. 20. Elapsed time of the 1-step palletizing task

As shown in the third column of Table 3 and Fig. 20., with the A* algorithm that considered the volume of the robot, the computing time of each step was remarkably different depending on the situation which is encountered at every step. The overlap method, however, produces fast and stable computation results, regardless of the place-down position and configurations of surrounding obstacles.

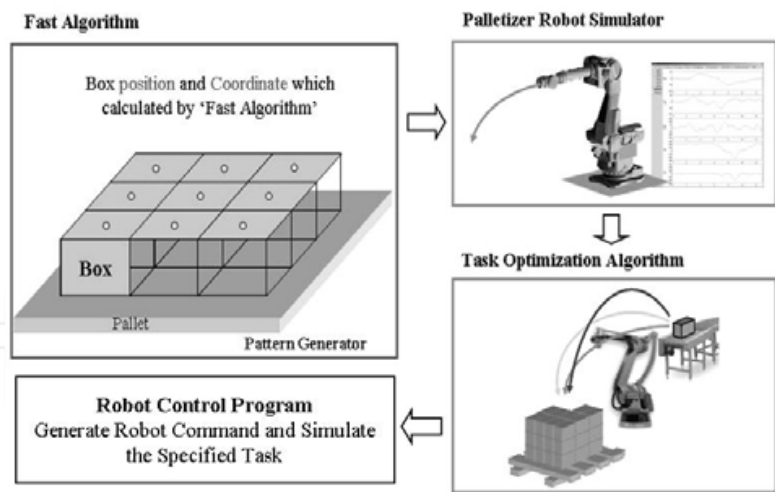


Fig. 21. Schematic diagram of the proposed palletizing OLP Simulation S/W

This chapter focuses on the realization of palletizing OLP Simulation S/W, which is a newly adapted method of treating and generating the path of the robot palletizing task efficiently. Fig. 21. shows the organization of this study. First, the box-loading patterns are generated on the pallet. This pattern was generated through the Fast algorithm that is handled in this chapter. The generated boxes were given a unique position value indicating the location and posture on the pallet where each box would be placed (i.e., the place-down point). Finally,

the robot moves the loads from pick-up points to place-down points through the trajectory generated by the overlap method.

Finally, Fig. 22. shows the total simulator that uses the proposed algorithms. The user defines the size of the box, the pallet, and the position of the working component on the workspace, and the simulator generates the optimized box-loading pattern, motion simulation of palletizing robot through the optimized trajectory and its task result as shown in enlarged image of Fig. 22.

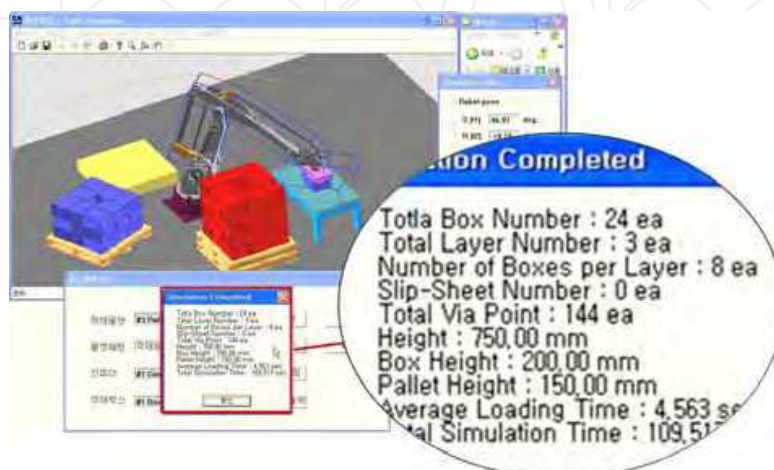


Fig. 22. Task results of the proposed OLP Simulation S/W

For application in the real robot system, however, the accuracy problem involving the synchronization of the robot simulator with the real robot, or the “calibration” problem, is an absolutely important issue. The calibration of a proposed palletizing S/W is suggested for future research. Provisionally, “three-point calibration” was considered to check and compensate for the errors between the workspace and the installed robot system.

8. References

- Debanik Roy (2005). Study on the Configuration Space Based Algorithmic Path Planning of Industrial Robots in an Unstructured Congested Three-Dimensional Space: An Approach Using Visibility Map, *Journal of Intelligent and Robotics Systems*, Vol. 43, No. 2-4, Aug 2005, pp. 111-145, 0921-0296
- Harold J. Steudel (1979). Generating pallet loading patterns: A special case of the two-dimensional cutting stock problem, *Management Science*, Vol. 25, No. 10, Oct 1979, pp. 997-1004
- J. H. Kim; J. S. Choi; H. Y. Kang; D. W. Kim & S. M. Yang (1994). Collision-Free Path Planning of Articulated Robot using Configuration Space, *Transactions of the KSAE*, Vol. 2, No. 6, Nov 1994, pp. 57-65, 1225-6382
- John J. Craig (2004). *Introduction to Robotics – Mechanics and Control*, 3rd Edition, Pearson Education Int., 978-0201543612
- Michael A. Hernan I (2000). *An Introduction to Automated Palletizing*, Anderson Technical Services, Inc.

- Pettersson, M.; Olvander, J. & Andersson, H. (2007). Application Adapted Performance Optimization for Industrial Robots. *IEEE International Symposium on Industrial Electronics*, pp. 2047-2052, 978-1-4244-0755-2, Jun 2007
- Ronald Graham (1972). An Efficient Algorithm for Determining the Convex Hull of a Finite Point Set, *Info. Proc. Letters* 1, pp. 132-133, Jan 1972
- Seung-Nam Yu; Heu-Kwon Yoon; Sung-Jin Lim; Young-Hoon Song & Chang-Soo Han (2005). The development of Robot Palletizing S/W using Fast Algorithm and 3-D Robot Simulator, *Proceedings of Korean Society of Mechanical Engineers*, pp. 1663-1668
- T. Lozano-Perez (1987). A Simple Motion Planning Algorithm for General Robot Manipulators, *IEEE Journal of Robotics and Automation*, Vol. RA-3, No. 3, Jun 1987, pp. 224-238, 0882-4967
- Warren, C.W. (1993). Fast Path Planning Using Modified A* Method, *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, pp. 662-667, 0-8186-3450-2, Atlanta, GA, USA, May 1993, IEEE Comput. Soc. Press
- Xiaojun Wu; Qing Li & Heng, K.H. (2005). A New Algorithm for Construction of Discretized Configuration Space Obstacle and Collision Detection of Manipulators, *Proceedings of 12th Int. Conf. on Advanced Robotics*, pp. 90-95, 0-7803-9178-0, Jul 2005
- Young-Gun G & Maing-Kyu Kang (2001). A fast algorithm for two-dimensional pallet loading problems of large size, *European Journal of Operational Research*, Vol. 134, No. 1, pp. 193-202,
- Zhao, C.S.; Farooq, M. & Bayoumi, M.M. (1995). Analytical solution for configuration space obstacle computation and representation, *Proceedings of the 1995 IEEE IECON 21st International Conference*, pp. 1278-1283, 0-7803-3026-9, Orlando, FL, USA, Nov 1995, IEEE

IntechOpen



Mechatronic Systems Applications

Edited by Annalisa Milella Donato Di Paola and Grazia Cicirelli

ISBN 978-953-307-040-7

Hard cover, 352 pages

Publisher InTech

Published online 01, March, 2010

Published in print edition March, 2010

Mechatronics, the synergistic blend of mechanics, electronics, and computer science, has evolved over the past twenty five years, leading to a novel stage of engineering design. By integrating the best design practices with the most advanced technologies, mechatronics aims at realizing high-quality products, guaranteeing at the same time a substantial reduction of time and costs of manufacturing. Mechatronic systems are manifold and range from machine components, motion generators, and power producing machines to more complex devices, such as robotic systems and transportation vehicles. With its twenty chapters, which collect contributions from many researchers worldwide, this book provides an excellent survey of recent work in the field of mechatronics with applications in various fields, like robotics, medical and assistive technology, human-machine interaction, unmanned vehicles, manufacturing, and education. We would like to thank all the authors who have invested a great deal of time to write such interesting chapters, which we are sure will be valuable to the readers. Chapters 1 to 6 deal with applications of mechatronics for the development of robotic systems. Medical and assistive technologies and human-machine interaction systems are the topic of chapters 7 to 13. Chapters 14 and 15 concern mechatronic systems for autonomous vehicles. Chapters 16-19 deal with mechatronics in manufacturing contexts. Chapter 20 concludes the book, describing a method for the installation of mechatronics education in schools.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

SungJin Lim, SeungNam Yu, ChangSoo Han and MaingKyu Kang (2010). Palletizing Simulator Using Optimized Pattern and Trajectory Generation Algorithm, *Mechatronic Systems Applications*, Annalisa Milella Donato Di Paola and Grazia Cicirelli (Ed.), ISBN: 978-953-307-040-7, InTech, Available from: <http://www.intechopen.com/books/mechatronic-systems-applications/palletizing-simulator-using-optimized-pattern-and-trajectory-generation-algorithm>

INTECH
open science | open minds

InTech Europe

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447

InTech China

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820

www.intechopen.com

Fax: +385 (51) 686 166
www.intechopen.com

Fax: +86-21-62489821

IntechOpen

IntechOpen

© 2010 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the [Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License](https://creativecommons.org/licenses/by-nc-sa/3.0/), which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen