COLLISION DETECTION FOR ROBOTIC MOTIONS USING CONFIGURATION SPACE

RONG-SHINE LIN

Department of Mechanical Engineering, and The Advanced Institute of Manufacturing with High-Tech Innovations,
National Chung Cheng University, Chia-Yi, Taiwan 62102, R.O.C
E-mail: imelin@ccu.edu.tw

Abstract - Robotic manipulators have become more and more popular in industrial automations. To implement robot motions on automations can be done easily by computer programming. However, the safety issue for robot motions in a workspace is very important in factories. If the robotic arm collides with the peripheral devices, it damages the robot itself and the collided devices. This research proposed a fast detection method to check whether the robot arms collide with the peripheral devices, using Configuration space. The major work for it includes Oriented Bounding Box modeling and Separating Axis Theorem for determining the object intersections. A loop-up table was built and stores 1 and 0 for object-intersected areas and the intersection-free areas, respectively. To check whether the robotic arms collide with obstacles, one just accesses the look-up table with robotic joint angles for the value either 1 or 0. If the value returns 1, the robotic arms collide with an obstacle. On the other hand, if the value returns 0, it's collision-free. The experimental result shows the checking time for collision detection is around 0.01ms in average, which is much less than the sampling time requirement for most of the robotic motion control applications.

Keywords - Collision Detection, Configuration space, Separating Axis Theorem

I. INTRODUCTION

Robotic manipulators have become more and more popular in industrial automations, including automated assembly, welding, painting, and material handling. With high performances for robotic assistances, it improves production, quality, efficiency, and reliability for many applications in industry[1-2]. To implement robot motions on different tasks can be done easily by computer programming. However, the safety issue for robot motions in a workspace is very important in factories. If the robotic arm collides with the peripheral devices, or call obstacles, such as machine tools, material racks, or conveyers, it damages the robot itself and the collided devices. This not only makes the production cost increasing, but makes the production rate decreasing due to the additional repairs.

There are many methods have been developed by researchers for collision detections, such as checking geometric intersection, bounding volume hierarchies, spatial partitioning, and depth imagine[3-7]. Some of them required additional sensing assistances. Some required longer computation time for accurate detection results. This research proposed a fast detection method to check whether the robot arms collide with the peripheral devices, using Configuration space, or called C-space[8-9]. The C-space represents the set of all possible joint configurations of a robot. Therefore, the collision areas of a robot with obstacles in C-space can be identified easily for detections.

The proposed collision detection method starts from the modeling of an object using the Oriented Bounding Box(OBB) model that represents the object in terms of rectangular solids[4-5]. The objects include robotic arms and peripheral devices in workspace for this application. The Separating Axis Theorem(SAT) were used to determine whether two objects intersected with each other[10-11]. The major process for SAT is finding the separating axis from all potential separating axes. If a separating axis exists, these two objects do not intersect. Although the computation load for applying SAT on the object intersection depends on the number of potential separating axes, those are precalculated information, and won't be taking into account for the collision-detection checking time. A C-space map can be built from those information when applying to the collision detection for robotic arms with obstacles. The C-space map shows the object-intersected areas and the intersection-free areas in terms of the robotic joint angles. Finally, a loop-up table stores 1 and 0 for object-intersected areas and the intersection-free areas, respectively. To check whether the robotic arms collide with obstacles, one just accesses the look-up table with robotic joint angles for the value either 1 or 0. If the value returns 1, the robotic arms collide with an obstacle. On the other hand, if the value returns 0, it's collision-free.

II. SEPARATING AXIS THEOREM

This research uses separating axis theorem to determine if two objects intersect or not. The separating axis theorem applies to any convex shapes. The shapes can be 2D polygons or 3D polyhedrons. The statement for the theorem as, two convex polygons do not intersect if and only if there exists a line that completely separates these two polygons. This line is called the separating line, which is perpendicular to the separating axis[10].

The projection method is used to determine if there exists a separating line, by checking the projection areas for these 2 polygons onto the separating axis, which is perpendicular to separating line. If these 2 projection areas do not overlap, these 2 polygons do not intersect. Figure 1 illustrates the polygon projecting onto the separating axis. The projection areas are $\overline{A_{max} \cdot A_{min}}$ and $\overline{B_{max} \cdot B_{min}}$ for polygon A and B, respectively.

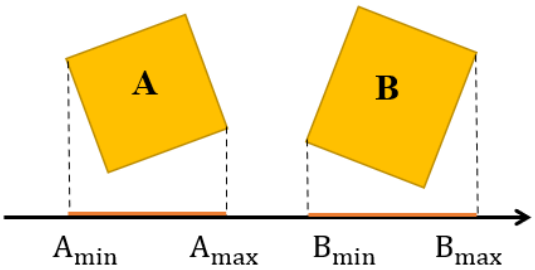


Figure 1: The projections of polygon A and B on separating axis

The Separating axis theorem can be extended into 3D shape objects, which become to polyhedrons. This research uses OBB model, which is a rectangular solid, for both robotic arms and detecting objects. Each rectangle solid is represented by 8 vertices and 3 surface normal vectors, as shown in Fig. 2. Solid A has 8 vertices $(a_1, a_2, a_3, a_4, a_5, a_6, a_7, and\ a_8,)$ and 3 surface normal vectors $(\vec{u}_1, \vec{v}_1, and\ \vec{w}_1)$, likewise for Solid B. There are 15 potential separating axes for determining if two rectangular solids are going to be checked for intersecting.

The 15 potential separating axes are follows,

6 surface normal vectors:

$$\vec{u}_1, \vec{v}_1, \vec{w}_1, \vec{u}_2, \vec{v}_2, \vec{w}_2$$

9 cross product vectors of surface normal vector associated with the 2 solids.

$$(\vec{u}_1 \text{ x } \vec{u}_2), \quad (\vec{u}_1 \text{ x } \vec{v}_2), (\vec{u}_1 \text{ x } \vec{w}_2),$$
$$(\vec{v}_1 \text{ x } \vec{u}_2), (\vec{v}_1 \text{ x } \vec{v}_2), (\vec{v}_1 \text{ x } \vec{w}_2),$$
$$(\vec{w}_1 \text{ x } \vec{u}_2), (\vec{w}_1 \text{ x } \vec{v}_2), (\vec{w}_1 \text{ x } \vec{w}_2)$$

where x is the cross product operator

A potential separating axis becomes a separating axis of two collision-free solids if and only if the projections of these two solids onto the potential separating axis do not overlap.

The algorithm for collision detection of 2 solids:

1. Project the vertices of A and B solids onto the potential separating axis, using inner product.
2. Find the min. and max. projection values for both solid A and B, (A_{min}, A_{max}) and (B_{min}, B_{max}).
3. If (A_{max} is less than B_{min}) or (A_{min} is greater than B_{max}), the projection areas for solid A and B do not overlap, which implies this potential axis is a separating axis, Then, the separating line separates solid A and B clearly.
4. If (A_{min} is less than B_{max} and A_{min} is greater than B_{mix}), or if (B_{min} is less than A_{max} and B_{min} is greater than B_{min}), the projection areas for solid A and B overlap, which implies this potential axis is not a separating axis.

The above algorithm for collision detection terminates if step 3 is true. This result shows solid A and B do not intersect. If Step 3 is false, all the potential separating axes need to be checked by step 4. If none of the separating axis can be found, Solid A and B intersect.

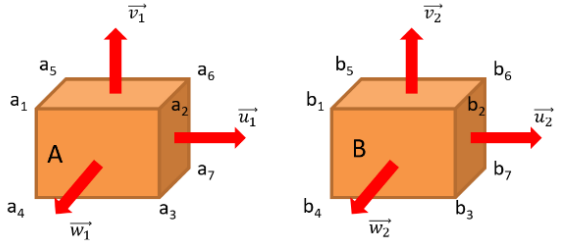


Figure2: Rectangular solid for OBB model

III. CONFIGURATION SPACE ANALYSES

Configuration space, or called C-space, for robotic arm motion describes the robot end-effector positions in terms of the robotic arm-joint coordinates. For a two-joint and two-link robot arm, the 2D C-space shows the robot end-effector positions in terms of joint-1 and joint-2 angles for the horizontal axis and vertical axis, respectively. Figure 3 illustrates the joint angles, joint-1=75^o and joint-2=-125^o, for an end-point location in working space and the configuration space. This example assumes that both 2 joints can rotate freely from -180^o to $+180^o$.

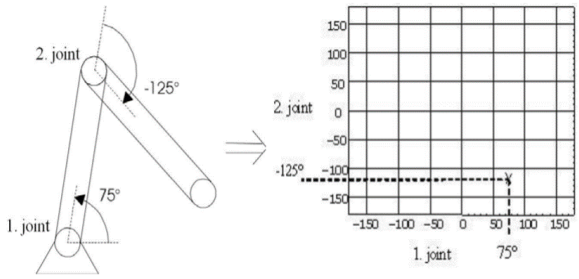


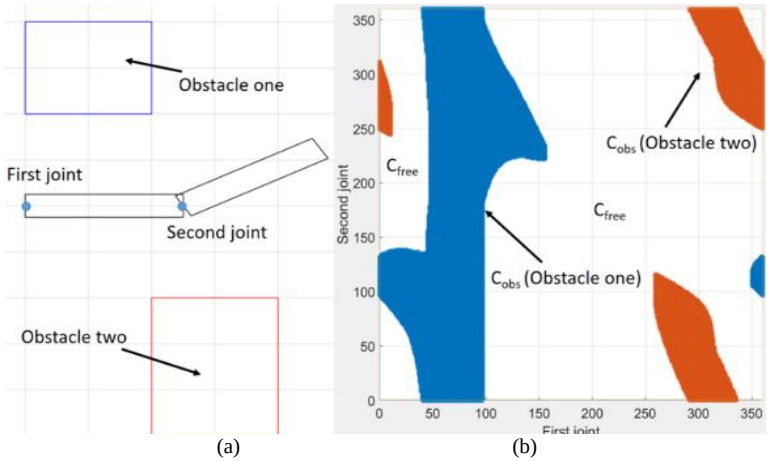
Figure 3: (a) the working pace, (b) configuration space

Applying previous collision detection algorithm to this 2 degree-of-freedom (DOF) robotic arm, the robot moving paths can be checked whether the robotic arms are colliding with an obstacle. If the checking result for a particular robot arm position, with 2-joint angles, collides with the obstacle, mark 1 for this position in C-space. The entire robot-arm motions, moving from -180^o to $+180^o$ for both joint-1 and joint-2, can be checked for collision detection with obstacles, and produce the completed 2D C-space map with collided areas. Figure 4 shows (a) a 2 DOF robotic arm and 2 rectangular obstacles in a workspace, and (b) the collided areas in C-space map. The blue areas and orange areas in C-space represent the obstacle-1 and obstacle-2, respectively.

A look-up table is created for the collided areas according to the C-space, blue and orange, which

stores “1” as collided. Likewise, the white areas in C-space represent the collision-free, which stores “0” in the look-up table. This look-up table will be used later for collision detection easily by checking the joint angles for the robot arm. If the look-up table value shows 1 for the checking joint angles, the robotic arms collide with an obstacle. On the other hand, if the look-up table value shows 0for the checking joint angles, the arm does not collide with any obstacles. This look-up table with 0 and 1 can be used for a fast check whether the robot collides with an obstacle. The user inputs with the joint angles and accesses the look-up table value, either 1 or 0. If the value returns 1, the robot arm collides with an obstacle. Likewise, the returned 0 value means the collision free. The previous calculating load for the separating axes and projections are precalculated information, which do not count into the collision-checking time.

The C-space map and look-up table can be extended into 3D multi-axes robotic arm motion that detects whether the arms collide with the obstacles.



(a) (b)

Fig. 4 (a) A 2DOF robot arm with 2 obstacles, (b) the C-space and collided areas, blue and orange

IV. EXPERIMENTAL RESULTS

The C-space map and look-up table can be extended into 3D multi-axes robot motions that detects whether the robotic arms collide with the obstacles. This research implemented the C-space collision detection method on a Hiwin Robot-Arm, RT610-1869-GB, as shown in Figure 5. The OBB model was used for this robot-arm modeling. The robotic arms have 6 portions, link-1, -2, -3, -4, -5, and -6. The number of rectangular solid for each link is shown in Table1. The total number of rectangular solid for the OBB model is 14that presents this robotic arms. Figure 6 shows the Robot-arm OBB model in terms of rectangular solids, which includes a base at the bottom.



Figure 5: Hiwin Robot-Arm, RT610-1869-GB

Link	No. of rectangular solids
1	6
2	1
3	2
4	2
5	2
6	1

Table 1 The number of rectangular solid for each arm

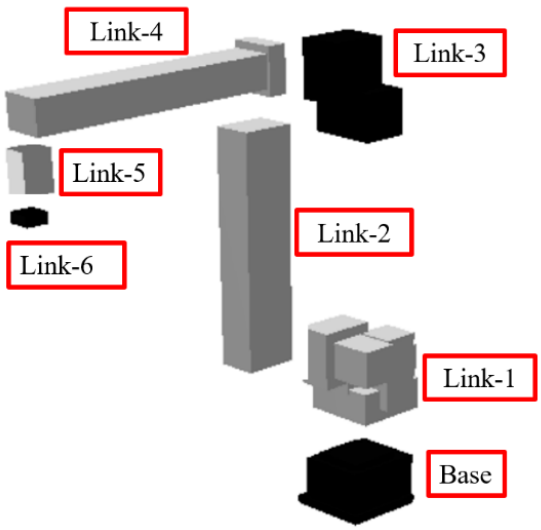


Figure 6: Rectangular solids for Robotic arms

The obstacles to be detected during robot moving are a machine tool and a rack. The numbers of rectangular solids in OBB models are 6 and 5, for the machine tool and rack, respectively. Table 2 shows the rectangular solid numbers for the obstacle OBB model. The machine tool and rack are shown in Fig. 7.

Obstacle	No. of rectangular solids
Machine tool	6
rack	5

Table 2 The number of rectangular solid for the machine tool and the rack

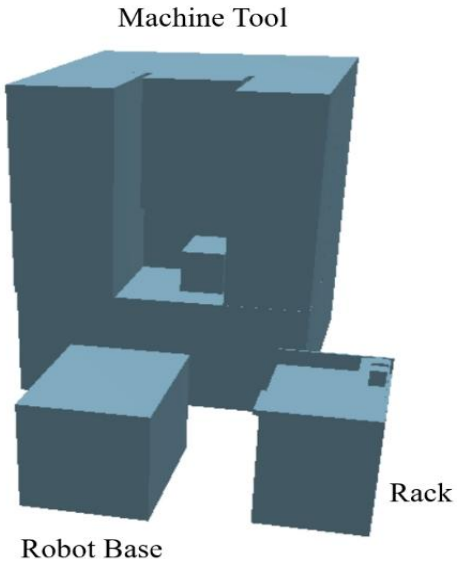


Figure 7: Rectangular solids for the machine tool and rack

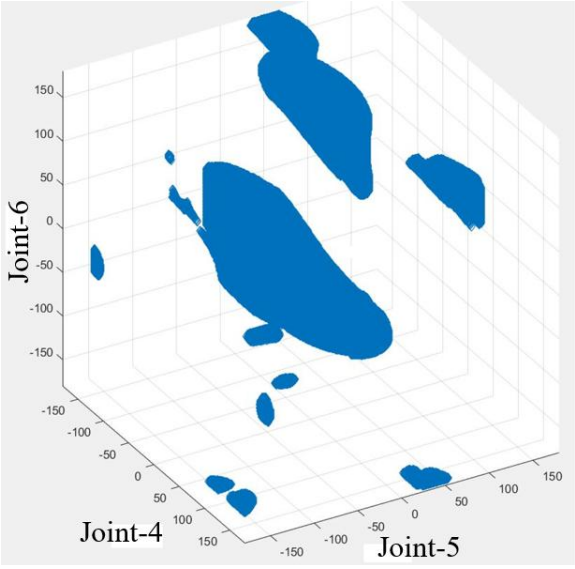


Figure 8: 3D C-space map with collided areas

3D C-Space

In order to build the 3D C-space, the first 3 joints of the 6-joint robot arm are fixed in a position, but rest of the joints, joint-4, -5, and -6, can rotate freely from -180^{o} to $+180^{o}$. By using Separating Axis Theorem and the same process of 2 DOF robotic collision detection, a 3D C-space can be built with collided areas and collision-free areas. Figure 8 shows a 3D C-space map for joint-4, -5, and -6 where the Joint-1 = 4°, Joint-2 = -12°, and J3 = -46°. The blue areas represent the robot-obstacle collision, which store 1 in the look-up table with 3 joint angles. On the other hand, the blank areas are collision free, and store 0 in the look-up table. With the same process, the 3D C-space map for joint-1, -2, and -3 can be obtained with fixed joint angles for Join-4, -5, and -6.

Six DOF Robot

The above collision detection method was implemented on the Hiwin 6 DOF robot arm, RT610-1869-GB, for 6-joint detections. The OBB model for the robotic arm with 6 links is shown in Figure 9. It's difficult to show the 6 DOF C-space map, but the look-up table for either 0 or 1 can be built by the same method. The workspace includes a robot, a machine tool, and a rack, as shown in Fig. 10. An end-effector with a cubical part has been added on Link-6, which has 4 rectangular solids. This makes the number of rectangular solid for Link-6 becoming to 5. A motion path has been checked whether the robot arm collide with the obstacles, machine tool and rack, using the C-space method. The input to the collision detection are 6-joint angles. Table 3 shows the joint angles for the demonstrated motion paths. The result for the collision detection for this path is no collision. Figure 11 shows the workspace with the robot, machine tool, and rack. The red path is the demonstration path for Table 3.

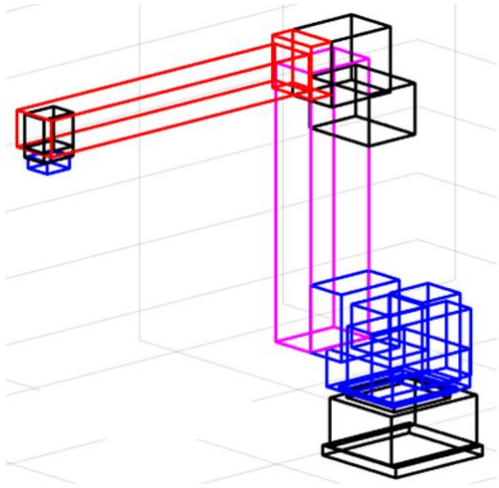


Figure 9: The6 DOF robotic arm

Joint Angle (degree)	J-1	J-2	J-3	J-4	J-5	J-6
Starting point	4	-12	-46	4	58	-2
By point	83	33	-20	2	46	-1
End point	90	-24	-8	0	36	0

Table 3 Joint angles for robotic paths

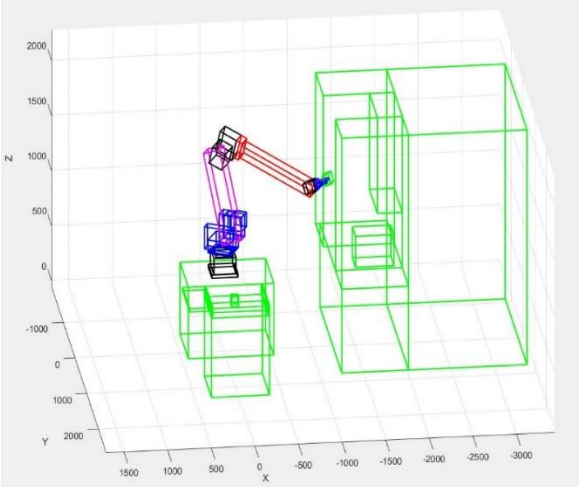


Figure 10: The workspace with a robot, machine tool, and rack

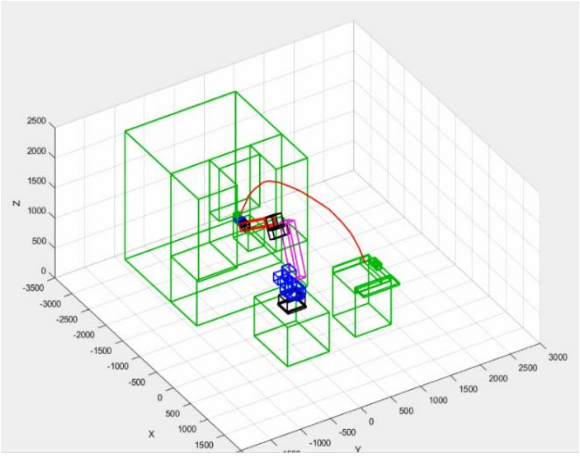


Figure 11: The robotic motion path in a workspace

This experiment shows whether the robot arms collide with the obstacle in C-space. All the major required calculating for collision detections, such as finding separating axis and object projections, were done before building the look-up table, either 0 or 1. The in-plant implements for collision detection processing is checking only the value in the look-up table, providing with robotic 6-joint angles. Therefore, the time required for collision detecting is very short. In this example, the 6 DOF Hiwin Robotic arm was modeled by 18 rectangular solids. The obstacle was modeled by 11 rectangular solids, which includes a machine tool and a rack. The average collision-detection checking time is 0.01ms for one movement. The average time was counted by taking the average of 100 different checks. This collision checking time is much less than the sampling time requirement for most of the robotic motion control applications. The computation was under an Intel(R) Core i7-8750H CPU, 2.2 GHz, with 24 GB RAM, and 64 bit processor.

V. CONCLUSION

This research proposed a fast detection method to check whether the robotic arms collide with the peripheral devices, using Configuration space. The major work for it includes OBB model and Separating Axis Theorem. A loop-up table was built and stores 1 and 0 for object-intersected areas and the intersection-free areas, respectively. To check whether the robotic arms collide with obstacles, one just accesses the look-up table with robotic joint angles for the value either 1 or 0. If the value returns 1, the robotic arms collide with an obstacle. On the other hand, if the value returns 0, it's collision-free. The experimental result shows the checking time for collision detection is around 0.01ms in average, which is much less than the sampling time requirement for most of the robotic motion control applications.

REFERENCES

[1] Al-Turaiki, I., & Ali, A., "Robotics technology in the industrial automation: An overview of present and future," Journal of Advanced Research in Dynamical and Control Systems, 10(12), pp.1607-1620, 2018.

[2] Y. Xu, Y. Zhang, B. Xie, X. Zhang, and W. Chen, "An overview of industrial robot applications and benefits," International Journal of Advanced Manufacturing Technology, 107(9-10), pp.3189-3205, 2020.

[3] Jack Ritter, "An Efficient Bounding Sphere," In Graphics Gems, A. Glassner, Ed. Academic Press, 301–303, 1990.

[4] Y. Shen, and X. Sun, "Research and Improvement of Collision Detection Based on Oriented Bounding Box in Physics Engine," in 2011 IEEE 3rd International Conference on Communication Software and Networks: IEEE, pp. 73-76, 2011.

[5] Jung-Woo Chang, Wenping Wang, Myung-Soo Kim, "Efficient collision detection using a dual OBB-sphere bounding volume hierarchy," Computer-Aided Design 42.1, pp.50-57, 2010.

[6] Markus Fischer, Dominik Henrich, "3D Collision detection for industrial robots and unknown obstacles using multiple depth images," Advances in Robotics Research, Springer Berlin Heidelberg, pp.111-122, 2009.

[7] L. Mathe, A. Caverzasi, F. Saravia, and G. Gomez, J. Pedroni, "Detection of human-robot collision using kinetic," IEEE Latin Am. Trans, Vol.11, pp.143-148, 2013.

[8] L´eonard Jaillet, Juan Cort´es, and Thierry Sim´eon, "Sampling-Based Path Planning on Configuration-Space Costmaps," IEEE TRANSACTIONS ON ROBOTICS, VOL. 26, NO. 4, pp.635-646, 2010.

[9] Ariyan M. Kabir, Alec Kanyuck, Rishi K. Malhan, Aniruddha V. Shembekar, Shantanu Thakar, and Brual C. Shah, "Generation of Synchronized Configuration Space Trajectories of Multi-Robot Systems," 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, pp.8683-8690, 2019.

[10] C. Gino and I. Mahoney, "Implementing the separating axis theorem for collision detection," IEEE Computer Graphics and Applications, 30(6), pp.34-43, 2010.

[11] Cheng Liang and Xiaojian Liu, "The Research of Collision Detection Algorithm Based on Separating axis Theorem," International Journal of Science, Vol.2 No.10, pp.110-114, 2015.

★★★