Indonesian Journal of Electrical Engineering and Computer Science
Vol. 1, No. 1, January 2016, pp. 109 ~ 118
DOI: 10.11591/ijeecs.v1.i1.pp109-118

Collision Detection and Trajectory Planning for Palletizing Robots Based OBB

Aal-Hasan Ali Joodi Jasim*[1], Tang Xiao Qi[2], Song Bao[3]
National Engineering Research Center of Numerical Control System,
Huazhong University of Science and Technology, 430074, Wuhan, Hubei, P.R. China
*Corresponding author, e-mail: ali.judy@yahoo.com[1], xqtang@hust.edu.cn[2], songbao@hust.edu.cn[3]

Abstract

Collision problems of convex polyhedra, in different applications, attract researcher's attentions recently. This paper proposes an efficient collision detection algorithm, for palletizing robot's end-effector grasped a convex polyhedra object (Box), based Oriented Bounded Boxes (OBB) theory. The OBB theory is preferable for detecting collision, because of its ability to handle unspecified orientations of objects and the transformations, with high-accuracy. The key factor for detecting collision between two OBBs is the Separating Axes Theorem (SAT). The spatial rotation representation of boxes is based Z-Y-Z Euler's angles. The algorithm presents incremental distance computation, for planning translation path and spherical trajectory. Checking and simulations in C++ language and AutoCAD software, attest the accurate results.

Keywords*: Collision Detection, Oriented Bounded Boxes (OBB), SAT-theorem, Euler's Angles, Translation, Spherical Trajectory*

1. Introduction

The importance of collision detection in Industrial palletizing robotics has been increased by the researchers and companies recently. In this paper, a presentation of simple methods related with robot end-effector that can detect before or during a collision of the robot. This algorithm often sensing the collision and it could avoid or stop the collision. The problem is basically computationally and geometrically; it can be formulated as a linear programming [1] . Finding the separating states between objects, calculated by determine the dimensions and the positions of objects testing.

To solve the problem of separating or detecting collision state by simplifying the complexity of the shapes and reduce the costs. The main common theories ideas are bounding by volumes the object, theories like Oriented Bounded Boxes (OBB), Axis Aligned Bounded Boxes (AABB), and Sphere Bounding Volumes. The main idea of these theories is modification complex computationally object to Simplified and easy to handle mathematically and geometrically[2] .

AABB has many representations, but the most common one is searching for the maximum and the minimum dimensions of the new box (corners) which is bonded the object; the distances between these points represent the align axis of the box[3] . It is very fast detecting method, and its speeds up the collision detection process, because of simple implementation idea. Bounding Sphere Theory is Simple overlap test method and very fast continues tests with cheap memory costs, because only the radiuses is required to compute the distances between objects [4] .

However, both of these methods are not convenient with transformation like a rotation or trajectory, because both have one orientation and they need to update data after each one step and the results will not be accurate enough, hard implementing and complex. Mainly, it is more desirable with video game graphics and animations.

Oriented Bounded Boxes (OBB) represents the best solution. Bonding Volumes by OBB theory is a rectangular block bounding any complex objects and makes it feasible computationally and geometrically. In other words, its minimum box that bounding a complex convex polyhedron's object perfectly and tightly[5] . Representation of OBB is Center Point (c) and three axes start from this center point (x, y, z) and oriented by right hand rule. OBB makes

Received July 11, 2015; Revised October 24, 2015; Accepted November 15, 2015

a bounded box in arbitrary orientation and that make it perfect choice with transformation while the robot end effector manipulating the box from a frame to another. The number of parameters required to test is less than the other theories [6] , but the memory required is the almost high values, however, development of high memory of the computer chips filled this gap between bounding volumes theories now days [7] . Testing two of OBBs is related with Separating Axes Theorem (SAT). SAT assume that two convex polyhedra; they are not colliding if and only if there is at least one separating plane between them and the normal vector of this plane separate them. This technic is hard and complicated to implement it, even though; the proposed algorithm will try to combine it with the transformation, and planning path or trajectory[8] .

Next sequence of this paper will be as follows: section 2 present best rotation representation. In 3rd section, most important equations and notices require to achieve an accurate collision detection. In 4th section present path planning by translation and 5th section presents spherical trajectory planning. In section 6 a presentation of simulation results from previous sections. This paper is concluded in section 7 and offers contributions of algorithm.

2. Rotation Representation

Best representation of spatial rotation of boxes assumed to be done by Z-Y-Z Euler's angles, which is representing one of the common rotation's representations. Euler's angles assume the rotation about one fixed (z-axis) *Roll* by an angle (α), and then rotate about (y-axis) *Pitch* by an angle (β) and finally rotate about (z-axis again) *Yaw* by an angle (γ) [9] . The final matrix that represents the total box spatial rotation by Z-Y-Z Euler's angles, the box will transform from frame $\{A\}$ to frame $\{B\}$, is as follows:

$$ {}^{A}_{B}R_{Z_1YZ_2}(\gamma,\beta,\alpha) = \begin{pmatrix} c\alpha c\beta \,\mathrm{c}\,\gamma - s\alpha s\gamma & -c\alpha c\beta s\gamma - s\alpha c\gamma & c\alpha s\beta \\ s\alpha c\beta c\gamma + c\alpha s\gamma & -s\alpha c\beta s\gamma + c\alpha c\gamma & s\alpha s\beta \\ -s\beta c\gamma & s\beta s\gamma & c\beta \end{pmatrix} \quad (1) $$

Where, $c\alpha = \cos\alpha$ and $\mathrm{s}\,\beta = \sin\beta$ and so on. As a result, the representation of the box have coordinates position (x, y, z) after spatial rotation in its frame, and by simplify eq.1, we will get the matrix of this box as follows:

$$ {}^{\mathrm{A}}_{\mathrm{B}}R_{Z_1YZ_2}(\gamma,\beta,\alpha) = \begin{pmatrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2) $$

3. Detecting Collision

Collision detection between two OBB represents a homogeneous test; hence, minimizing testing numbers of boxes tested, this step will eschew the tree construction cost between numbers of boxes. By considering the Cartesian coordinates positions and analyzing the dimensions of OBBs into vectors takes the right hand orientation method. Assume two Boxes (A & B) having an arbitrary orientation, and the following parameters, Figure 1:

Box – A	Box – B
$\mathrm{P_A}$ = coordinate position of the center of A.	$\mathrm{P_B}$ = coordinate position of the center of B.
$\mathrm{A_X}$ = unit vector representing the x-axis of A.	$\mathrm{B_X}$ = unit vector representing the x-axis of B.
$\mathrm{A_Y}$ = unit vector representing the y-axis of A.	$\mathrm{B_Y}$ = unit vector representing the y-axis of B.
$\mathrm{A_Z}$ = unit vector representing the z-axis of A.	$\mathrm{B_Z}$ = unit vector representing the z-axis of B.

W_A = half width of A. W_B = half width of B.
H_A = half height of A H_B = half height of B
D_A = half depth of A. D_B = half depth of B.

After rotation, each box will have its own matrix. For box A, the matrix will be as follows;

$$R^{A}{}_{Z_1YZ_2} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & X_A \\ a_{21} & a_{22} & a_{23} & Y_A \\ a_{31} & a_{32} & a_{33} & Z_A \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3)$$

and for box B matrix will be like:

$$R^{B}{}_{Z_1YZ_2} = \begin{pmatrix} b_{11} & b_{12} & b_{13} & X_B \\ b_{21} & b_{22} & b_{23} & Y_B \\ b_{31} & b_{32} & b_{33} & Z_B \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (4)$$

In each box, three edges to concern about it, as a result will get (3*3) possible separating planes produced by cross product of these edges, and these edges parallel to the axes of the box. Hence, the expecting separating planes will be $(A_X \times B_X)$, $(A_X \times B_Y)$, $(A_X \times B_Z)$, $(A_Y \times B_X)$, $(A_Y \times B_Y)$, $(A_Y \times B_Z)$, $(A_Z \times B_X)$, $(A_Z \times B_Y)$, $(A_Z \times B_Z)$

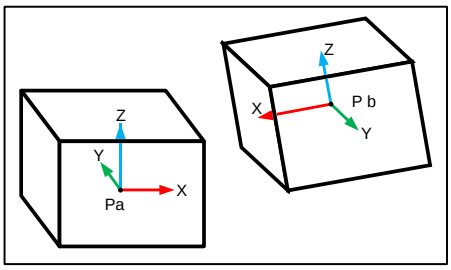

Figure 1. Right Hand Rule of Two Boxes

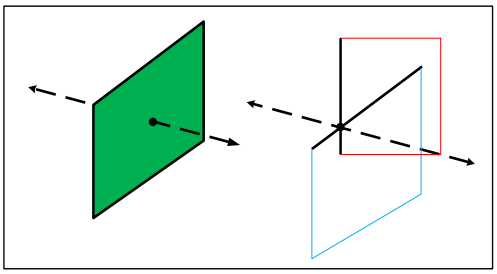

Figure 2. Shows Cross Product of Two Different Edges & The Normal Vector

These nine planes produced by the cross product of boxes edges. Figure 2 illustrates this idea. In each plane, there is normal vector that separates boxes. The expecting separating states will add 6 more expecting axes, which are the edges of boxes as a separating axes (3*2) (A_X, A_Y, A_Z, B_X, B_Y, B_Z). Totally will get 15 separating vectors must to find. The general assumption of SAT equation is:

$$|d \bullet \mathrm{L}| > |W_A \bullet \mathrm{L}| + |H_A \bullet \mathrm{L}| + |D_A \bullet \mathrm{L}| + |W_B \bullet \mathrm{L}| + |H_B \bullet \mathrm{L}| + |D_B \bullet \mathrm{L}| \quad (5)$$

That's mean if the projection of the distance between boxes (***d***) bigger than the projections of boxes on the same separating axes, then there is no collision, else, there will be a collision. Where (***d***) represent the distance between two boxes and its value can be find by:

$$d = P_B - P_A = \sqrt{(X_B - X_A)^2 + (Y_B - Y_A)^2 + (Z_B - Z_A)^2} \quad (6)$$

and (***L***) Represent the expecting separating axes, which is one of the 15 vectors. Finding the projection of (***d***) on (***L***) depending on the other dimensions of boxes coordinates positions after rotation. To find the projection of (***d***) on (A_X), and the same idea can apply it with other axes:

$$| d \bullet A_X | = \frac{| a_{11} * d_X | + | a_{21} * d_Y | + | a_{31} * d_Z |}{| W_A |} \quad (7)$$

To continue finding the right side of SAT assumption in Eq. (5), by finding the projection of boxes on (A_X):

$$| A \bullet A_X | = | W_A \bullet A_X | + | H_A \bullet A_X | + | D_A \bullet A_X | \quad (8)$$

and also finding the projection of box (B) on (A_X):

$$| B \bullet A_X | = | W_B \bullet A_X | + | H_B \bullet A_X | + | D_B \bullet A_X | \quad (9)$$

The projection of dimension parallel to (***L***) will be equal to its value. Like the projection of ($| W_B \bullet A_X |$) on (A_X) will be the same value of (W_B), in figure 3. All projections of boxes dimensions can be found by implementing the same method implemented before to find the projection of (***d***) in Eq. (7). Also, this idea can be implementing it with ($A_X, A_Y, A_Z, B_X, B_Y, B_Z$).

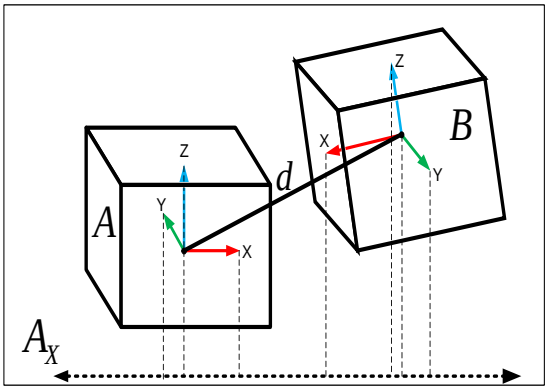

Figure 3. Projections on (A_X) axis

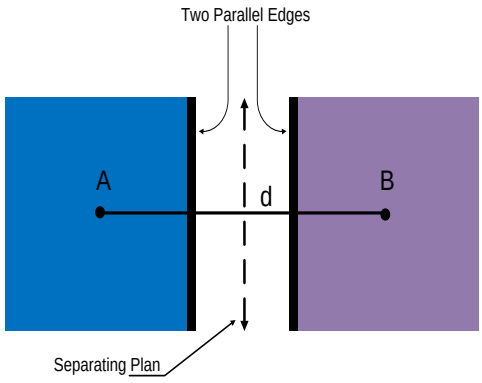

Figure 4. NO Cross Product Between Two Parallel Edges

To find the projection of (***d***) and the boxes on one of the expecting separating planes that contain the normal vector of $(A_X \times B_X)$, $(A_X \times B_Y)$, $(A_X \times B_Z)$, $(A_Y \times B_X)$, $(A_Y \times B_Y)$, $(A_Y \times B_Z)$, $(A_Z \times B_X)$, $(A_Z \times B_Y)$, $(A_Z \times B_Z)$ will be as follows, by taking the $(A_X \times B_X)$ plan as an example, and can implementing the same method in others planes.

$$\begin{aligned} |d \bullet (A_X \times B_X)| > {} & |W_A \bullet (A_X \times B_X)| + |H_A \bullet (A_X \times B_X)| + |D_A \bullet (A_X \times B_X)| \\ & + |W_B \bullet (A_X \times B_X)| + |H_B \bullet (A_X \times B_X)| + |D_B \bullet (A_X \times B_X)| \end{aligned} \quad (10)$$

Finding the projection of (***d***) on ($A_X \times B_X$), will be as follows:

$$d \bullet (A_X \times B_X) = (d \bullet A_Z) * (A_Y \bullet B_X) - (d \bullet A_Y) * (A_Z \bullet B_X) \quad (11)$$

Find the normal vector results from the cross product of $(A_X \times B_X)$ and then project the dimensions on it. The same idea can be implementing it with other planes.

In some cases, there are two edges (axes) are parallels, and there is no cross product between them. To check the edges are parallel and they are not intersected, simply, the projection of (***d***) on both is equal to (***zero***). Here, the user directly must assume that the cross product equal to (***zero***) and the value of the separating axis will be also (***zero***). The projection of (***d***) is equal to (***zero***) too. Figure 4 depicting this idea[10] [11] .

4. Translation Path Planning

Assume the start point coordinates of the box (1) in figure 5, are (*x, y, z*) and the end point or the target coordinates are (*x′, y′, z′*) hence, the total translation (***t***) equation between points will be as follows:

$$t(x, y, z) = (\Delta x, \Delta y, \Delta z) \quad (12)$$

then:

$$(x', y', z') = (x + \Delta x, y + \Delta y, z + \Delta z) \quad (13)$$

where *(∆x, ∆y, ∆z)* represent the translation along (x, y, z) axes respectively. Simply, adding the translation value to the original value. While, Inverse translation can also be considered if the translation done in the inverse direction, it will be as follows:

$$(x', y', z') = (x - \Delta x, y - \Delta y, z - \Delta z) \quad (14)$$

The most important notice in translation is that one of the parameters must be fixed, and the other is in motion. The whole system must be in one space. In this assumption, assume the plane that contains all objects is fixed by an origin point (**o**) which represents the reference point and it's often starting point.

The rotations combine the translation, then, the transformation of any point (*P(x, y, z)*) will transform to (*P′ (x′, y′, z′)*), then the total transformation (T) will be as follows:

$$T = \begin{pmatrix} r_{11} & r_{12} & r_{13} & \Delta x \\ r_{21} & r_{22} & r_{23} & \Delta y \\ r_{31} & r_{32} & r_{33} & \Delta z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (15)$$

A (3D) rigid body that is capable of translation and rotation, therefore, had six degrees of freedom (*(∆x, ∆y, ∆z, α, β, γ*).

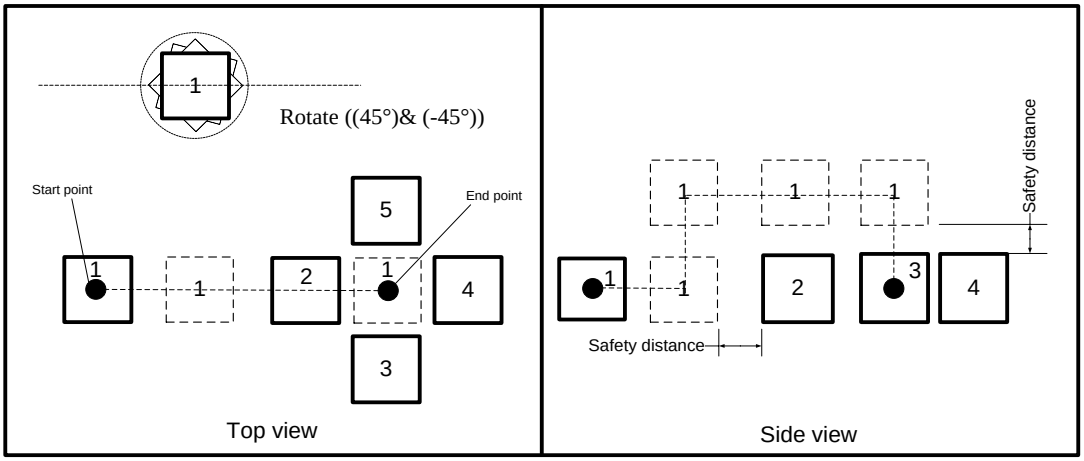

Figure 5. Translation Path Assumption

Nothing better than a straight line connect the start point and an end point to represent the translation. For the grasped box, it is important to consider the end effector dimensions to moving and placing it accurately, but ignoring the grasping calculations and focusing on box motion and its path. Hence, finding a straight line could be more complicated. So, considering that the robot motion will be autonomous, and it will react after detecting any collision (before collide), by finding different path to reach the target safely. Another important technic before translation is rotating the grasped box with an end effector (45° & -45°) and enlarging the total dimensions, and that to avoid collision by adding safety distance[12] . As figure 5 shows, the box try to move in straight line towards its target, but it's changed a path immediately after detecting collision with another boxes. The safety distance it has to calculate before starting motion[13] . For that, divide the rotation and the total translation into steps, and detecting collision in each step, if there is no collision; continue adding distance and checking again the new step... and so on, until reach the end point. The reaction of the robot must be reacting autonomously, and will always search for new path in different directions, with high computing accuracy. For C++ language's programmers make a while loop and dividing all the distance into small distances will be perfect to achieve that. As example, dividing (10) into ten steps and that's mean detecting for collision every (1 step). The while loop stop, after reach the end point target.

5. Spherical Trajectory Planning

To plan a spherical trajectory, three points are requires to find before achieving that target. It is important to calculate safety space before starting motion, by rotating the end effector that grasped the box (45° & -45°), just like in translation path, and then by assume a fix coordinate point as reference point (o) to connect all other points with it. Then, will assume that the start arc point $P_0 = (\mathrm{x}_0, \mathrm{y}_0, \mathrm{z}_0)$, and an end arc point $P_2 = (\mathrm{x}_2, \mathrm{y}_2, \mathrm{z}_2)$, which represent the target. Third point $P_1 = (\mathrm{x}_1, \mathrm{y}_1, \mathrm{z}_1)$ is arbitrary assume lay in the middle distance of the arc between the two other points, and also can make equal chords between (P_2) and (P_0). To find (P_1), assuming that the line from the center middle point of ($\overrightarrow{\mathrm{P}_0\mathrm{P}_2}$) vector is equal to $((P_2 - P_0)/4)$. Figure 6 shows the points. As a result, the center point of the arc ($c = (x_c, \mathrm{y}_c, \mathrm{z}_c)$) has the same distance equally from all three points. Figure (7) shows the assumption of the trajectory and the points. Another assumption has to consider is the speed (**F**) and the interpolating cycle (**T**) which is important to give a value previously [14] .

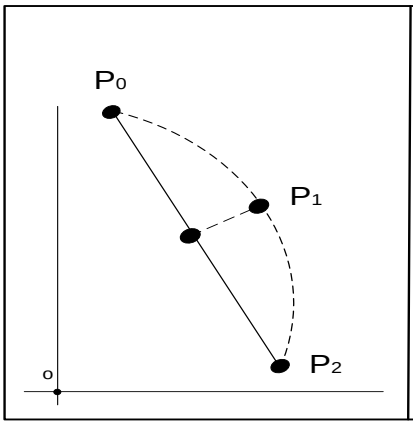

Figure 6. (P_1) Assumption

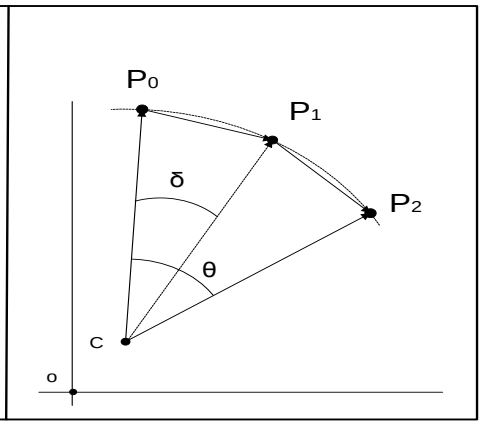

Figure 7. Three Point of Spherical Trajectory

The center of the arc can be calculated as follows:

$$\mathrm{A}[x_c, \mathrm{y}_c, \mathrm{z}_c]^{\mathrm{T}} = \mathrm{B} \tag{16}$$

where:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 2(X_1 - X_0) & 2(Y_1 - Y_0) & 2(Z_1 - Z_0) \\ 2(X_2 - X_1) & 2(Y_2 - Y_1) & 2(Z_2 - Z_1) \\ \left[a_{13}(a_{13}a_{22} - a_{21}a_{23})\right]/8 & [a_{13}(a_{11}a_{23} - a_{13}a_{12})]/8 & -\left[a_{11}(a_{13}a_{22} - a_{21}a_{23})\right]/8 \end{pmatrix} \tag{17}$$

$$\mathrm{B} = \begin{pmatrix} [(X_1^2 + Y_1^2 + Z_1^2)^2 - (X_0^2 + Y_0^2 + Z_0^2)^2] \\ [(X_2^2 + Y_2^2 + Z_2^2)^2 - (X_1^2 + Y_1^2 + Z_1^2)^2] \\ [a_{31}X_0 + a_{32}Y_0 + a_{33}Z_0] \end{pmatrix} \tag{18}$$

To find the normal vector of chords, and the points (**n**) which are perpendicular on all of them, the projection on them always equal to zero. As follows:

$$n = u_i + v_j + w_k = \overrightarrow{P_1P_0} \times \overrightarrow{P_2P_1} \tag{19}$$

The propose of computing (**n**) is to find the value of arc angle (θ), which is depending on (H), where (H) value can be found as follows:

$$H = uu_1 + vv_1 + ww_1 \tag{20}$$

where:

$$u_1 = (y_0 - y_C)(z_2 - z_0) - (z_0 - z_C)(y_2 - y_0) \tag{21}$$

$$v_1 = (z_0 - z_C)(x_2 - x_0) - (x_0 - x_C)(z_2 - z_0) \tag{22}$$

$$w_1 = (x_0 - x_C)(y_2 - y_0) - (y_0 - y_C)(x_2 - x_0) \tag{23}$$

The value of (u,v,w) results from the cross product of $(\overrightarrow{P_1P_0}\times\overrightarrow{P_2P_1})$, which represent the value of the normal vector (**n**). If the value of $(H\geq 0)$, then $(\theta\leq\pi)$, and can compute it as follows:

$$\theta = 2\arcsin\{\left[\left(X_2-X_0\right)^2+\left(Y_2-Y_0\right)^2+\left(Z_2-Z_0\right)^2\right]^{\frac{1}{2}}/(2R)\} \tag{24}$$

if $(H<0)$ then the value of $(\theta>\pi)$, and can be computing it as follows:

$$\theta = 2\pi - 2\arcsin\{\left[\left(X_2-X_0\right)^2+\left(Y_2-Y_0\right)^2+\left(Z_2-Z_0\right)^2\right]^{\frac{1}{2}}/(2R)\} \tag{25}$$

Now setting the starting point $P_0=(0,0,0)$, and then start motion until reaching the point $P_{i+1}=(\mathrm{x}_{i+1},\mathrm{y}_{i+1},\mathrm{z}_{i+1})$, and making detection for collision in every (*i+1*), till reach the final end point. The interpolating point (P_{i+1}) can be calculating it as follows:

$$P_{i+1}=\begin{pmatrix}\mathrm{Y_{i+1}=Y_C+G(Y_i+En_i-Y_C)}\\ \mathrm{X_{i+1}=X_C+G(X_i+Em_i-X_C)}\\ \mathrm{Z_{i+1}=Z_C+G\left(Z_i+El_i-Z_C\right)}\end{pmatrix} \tag{26}$$

where (G) can be computed as:

$$G=1/\left[1+\left(FT/R\right)^2\right]^{1/2} \tag{27}$$

and (E):

$$E=FT/\left(m_i^2+n_i^2+l_i^2\right)^{1/2} \tag{28}$$

where:

$$\begin{cases} m_i=v\left(Z_i-Z_C\right)-w\left(Y_i-Y_C\right)\\ n_i=w\left(X_i-X_C\right)-u\left(Z_i-Z_C\right)\\ l_i=u\left(Y_i-Y_C\right)-v\left(X_i-X_C\right)\end{cases} \tag{29}$$

The equations above representing the calculations of the next point by adding and expecting small errors could cumulate after each interpolate. The total error value is very small amount and can be calculating it as $(FT^2/8R)$, which can neglect it. The interpolating numbers account (***N***) is equals to:

$$N=\frac{\theta}{\delta}+1 \tag{30}$$

where (δ) represent the Step angle which is dividing (θ) into steps along its arc, and compute as:

$$\delta = \arcsin\left(\frac{FT}{R}\right) \approx FT / R \tag{31}$$

This algorithm represents an accurate calculation for determining a spherical trajectory with minimizing errors expecting occurs while each step interpolating the grasping Box to new point until reach the final destination. This method is appropriate with transformation appears with homogeneous matrices of the grasped box, which meets the demands of users.

6. Results and Simulations

The presented before is an algorithm for planning an accurate collision detection combining the transformation, and this algorithm have been tested and checked by common tool (*C++ computer language*), which represents the simple tool to implement the proposed algorithm. Another tool for checking done by (*AutoCAD*) software, which showed the figures and the situations states of objects, and checking by naked eyes if the algorithm was correct or not by making the virtual environment graphics.

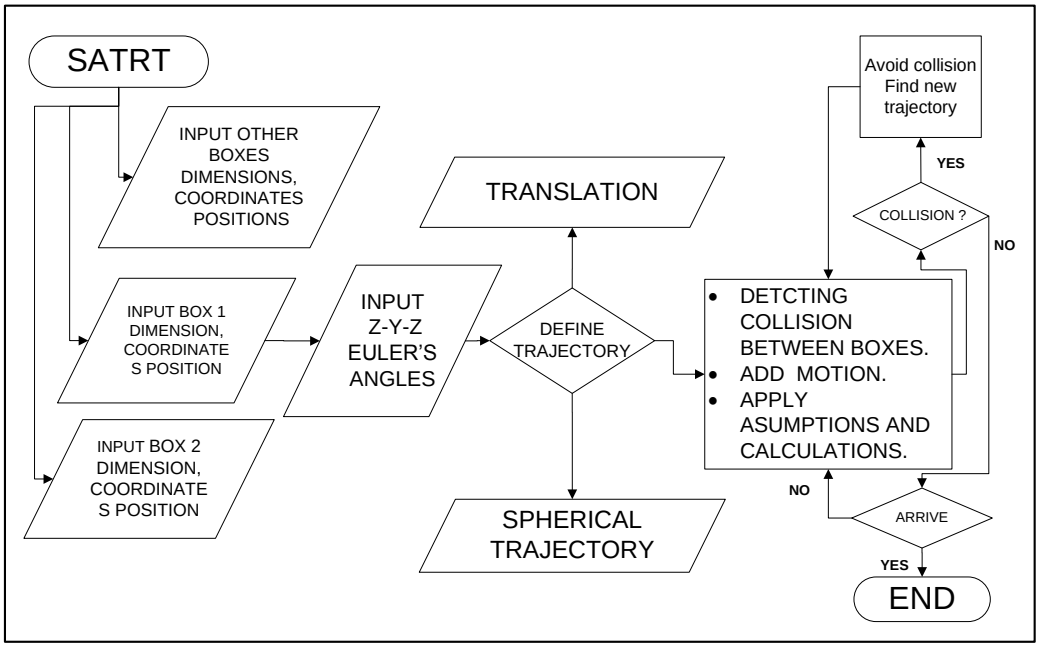

Figure 8. C++ Algorithm Logical Sequence

Drawing by AutoCAD shows the situation's states of boxes and the transformations, the spatial rotation of boxes combine translation or spherical trajectory from the frame to a new frame. As shown in Figure 8, the logical sequences of algorithm implemented in C++ and in Figure 9 the simulation of collision boxes and trajectory implemented by the AutoCAD. This method avoided expecting errors can be occurred, while dividing the curve interpolating into steps. For circuit contains hundreds, and even several hundred steps points, the intuitive three point's technic is impossible at all to be optimal. And the method based control is also infeasible due to its complexity and obstacles collision detection computation [15] .

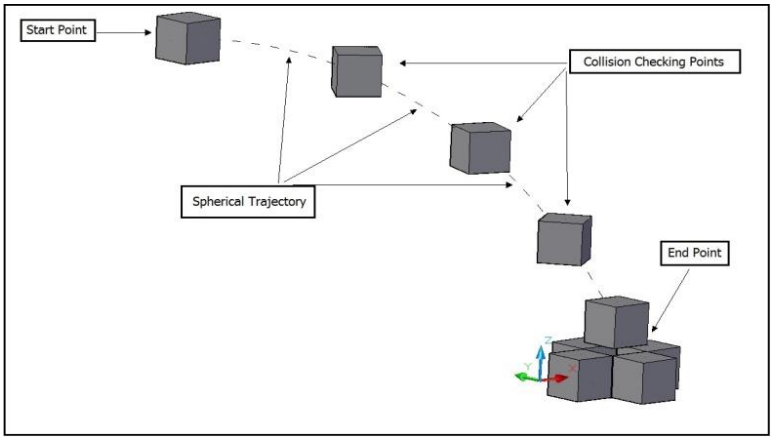

Figure 9. Auto CAD Spherical Trajectory Simulation

7. Conclusion

This paper presented an accurate algorithms and methods to achieve perfect collision detection, by presented correct steps to achieve this target, including implementing different types of transformation like rotation and planning translation and spherical trajectories. It is chosen to work in OBB bounding method instead of others theories and detecting collision by SAT theorem. SAT theorem represents the main tool to achieve collision detection accurately. The trajectory of grasped box by an end effector has translation and spherical trajectories, and divides the path into steps and detecting collisions in each step, and assuming the robot reacts autonomously. The efficiency of this algorithm have been tested and checked.

References

[1] Ericson C. Real-time collision detection. 1st Edition. United States, San Francisco: Elsevier. CRC Press. 2004; 5-156.
[2] Suaib N M. Bade A, Mohamad D. Hybrid Collision Culling by Bounding Volumes Manipulation in Massive Rigid Body Simulation. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2013; 11(6): 3115 - 3122.
[3] Bergen G. Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphics Tools*. 1997; 2(4): 1-13
[4] Philip M. Hubbard. Approximating Polyhedra with Spheres for Time-Critical Collision Detection. *ACM Transactions on Graphics*. 1996; 15(3): 179-210
[5] Fourar R, Melaab D. Acceleration of the Collision Detection for the Grasping of Objects by a Robotic Hand. *International Journal of Control & Automation*. 2015; 8(4): 181 - 200.
[6] Gottschalk S. *Collision queries using oriented bounding boxes*. University of North Carolina at Chapel Hill. 2000: 38-81
[7] Kay TL, Kajiya JT. Ray tracing complex scenes. *ACM SIGGRAPH computer graphics. ACM*. 1986; 20(4): 270-275
[8] Johnny Huynh. *Separating Axis Theorem for Oriented Bounding Boxes*. www.jkh.m. 2009: pp (3-45).
[9] Craig JJ. Introduction to robotics: mechanics and control. 3rd Edition. New Jersey, USA. Upper Saddle River: Pearson Prentice Hall. 2005; 19-61.
[10] Aarts JM. Plan and Geometry. 1st Edition. Netherland. Springer Science & business Media. 2009: 64-290
[11] Strang G, Aarikka K. Introduction to applied mathematics. 4th Edition. Wellesley, MA: Wellesley-Cambridge Press, USA. 2010; 376-460.
[12] Mason M.T. Mechanics of robotic manipulation.1st Edition. MIT press, USA. 2001; 41 - 71
[13] De Berg M, Van Kreveld M, Overmars M, et al. Computational geometry. 3rd Edition. Eindhoven, Netherland. Springer Berlin Heidelberg. 2000; 280-326.
[14] Bosheng Y. Implementation of arc interpolation in a robot by using three arbitrary points. *Journal-Hua Zhong University of Science and Technology Nature Science Edition*. 2007; 35(8): 5-8.
[15] Wang Y, Chi N. Path planning optimization for teaching and playback welding robot. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2013; 11(2). 960-968.