International Journal of Control and Automation
Vol. 8, No. 4 (2015), pp. 181-200
http://dx.doi.org/10.14257/ijca.2015.8.4.19

Acceleration of the Collision Detection for the Grasping of Objects by a Robotic Hand

[1]Redha Fourar and [2]Djamel Melaab

Department of Electrical Engineering, Batna University, Algeria
[1]fourarredha@yahoo.fr, [2]Melaab_djamel@yahoo.fr

Abstract

The objective of this work is to propose two new algorithms for collision detection for real-time application. They are applicable to rigid objects enclosed in boxes in order to improve the time of collision detection. The proposed algorithms, called Neuro-SAT and perceptron learning with displacement of the base frame, will be compared with the algorithm Separating Axis Test (SAT) based on the hierarchy of the OBB tree. A grasping operation of an object, with a robotic hand, was executed to test the developed algorithms. The results, of simulation experiments, reveal a great improvement in the time of collision detection.

Keywords: *algorithm Neuro-SAT, algorithm SAT, collision detection, OBB tree, perceptron, robotics.*

1. Introduction

The hierarchy of bounding volumes is recognized as a good means of collision detection and is widely used in many areas of the virtual environment. The basic idea is to enclose objects into bounding volumes and perform collision tests on these volumes rather than perform collision tests on the objects themselves. This process can result in significant performance improvements. The asymptotic complexity of the time remains the same and the use of bounding volumes improves the situation by a constant factor. The arrangement of bounding volumes in a hierarchy tree is called BVH (Bounding Volume Hierarchy).

A bounding volume allows define an area of interest around the virtual object. For example, a sphere [1,2], a box AABB [3] (Axis-Aligned Bounding Box) or a box OBB [4] (Oriented Bounding Box) can approximate a part of the object or the entire object. When the complexity of an object increases, a hierarchy of bounding volumes is used. This is generally a tree of spheres or boxes. Each volume delimits one or more geometric primitives. A set of these volumes, called parent, delimits the space of all the geometric primitives of its leaves. Generally, each bounding volume is optimized in terms of volume, air surface, diameter, etc., in order to have the better compactness around the bounded primitive. According to the design choices, the leaves of a tree can contain a single primitive or a collection of geometric primitives.

Several studies have been done to improve the collision detection time. In [5], an algorithm based on the AABB which saves a large amount of memory space and minimizes the collision detection time is developed. In [6], a hybrid collision detection algorithm, based on AABB and OBB, applicable to solid objects is proposed. Another algorithm, efficient for the collision detection according to a double hierarchy delimitation, which is composed of AABB and OBB is proposed in [7]. The AABB test is applied first to remove the distant objects; the remaining objects are tested using the separation axis of the OBB.

All the previously presented algorithms are based on research of a simple tree hierarchy in order to perform fewer collision tests to save time. In this work, we will propose two new efficient algorithms that accelerate the collision test between boxes OBBs, by looking for a separating axis rather than looking for a hierarchy that minimizes the number of test between boxes. This algorithm can be used to improve the algorithms proposed in [5], [6] and [7].

2. Construction of the Hierarchy of the Bounding Volumes

The construction of bounding volumes generally uses two approaches: from top to bottom [8] (top-down) and from bottom to top [9] (down-top) (figure 1). The principle of the top-down approach is to start from the root to the leaves [10], while the principle of the down-top approach is to start from the leaves to the root [11]. Relative to the top-down approach, the down-top approach is more complicated to implement and has a slower construction time. This is why we chose the top-down approach to construct the hierarchy of volumes.

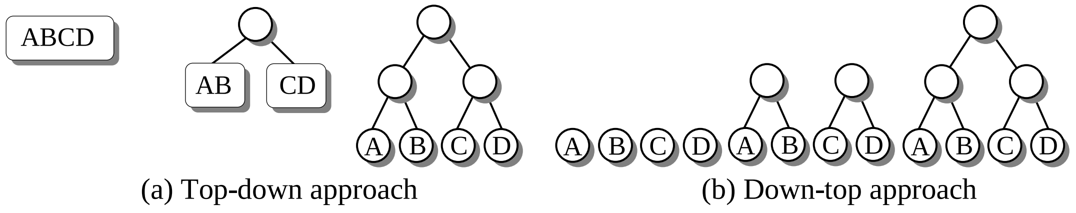


Figure 1. The Two Approaches for the Construction of Small Tree of Four Objects

3. The OBB Method

The goal is to construct a box that remains close to the dimensions of the object and which would have its orientation [12,13]. For a list of "n" triangles, be P^i , q^i and r^i the summits of the i^{th} triangle. The expressions of the mean μ and covariance matrix c are then:

$$\mu = \frac{1}{3n} \sum_{i=1}^{n} (p^i + q^i + r^i) \tag{1}$$

$$\begin{cases} c_{ij} = \frac{1}{3} \sum_{i=1}^{n} (\bar{p}^i_j p^i_k + \bar{q}^i_j q^i_k + \bar{r}^i_j r^i_k) \\ 1 \le i, j \le 3 \end{cases} \tag{2}$$

Where:

$$\begin{cases} \bar{p}^i = p^i - \mu \\ \bar{q}^i = q^i - \mu \\ \bar{r}^i = r^i - \mu \end{cases} \tag{3}$$

C is a 3x3 symmetric matrix. This matrix is diagonalized with the assurance that the eigenvectors are mutually orthogonal: these latter are chosen as axes of the box [14]. The maximum distances, between the summits along each axis, determine the dimensions of the bounding parallelepiped.

The test of intersections between OBBs are based on the test of the separating axis SAT (Separating Axis Test).

The hierarchical representation, with a tree of bounding boxes, uses this test to descend into the tree. The fastest algorithm is the SAT [4]. We will show in the following that the proposed method, based on the search for a separator axis, is faster than the SAT method.

4. Notion of the Separating Axis

Consider two separated boxes OBBs, A and B. B is placed with a rotation R and translation T relative to A. Then there exists a separator axis between the two boxes which satisfies the following condition [4]:

$$|T \cdot L| > \sum_i \left|(a_i A^i) \cdot L\right| + \sum_i \left|(b_i B^i) \cdot L\right| \tag{4}$$

Where A^i and B^i are the unit vectors of the six axes, of the boxes OBBs. a_i and b_i are the dimensions of the boxes according to the six axes. $L = AXB$, where A and B are separate vectors, selected from the six axes of the boxes. In Figure 2, L is a separating axis between the two OBBs, because it exists disjoint intervals when we perform a projection on this axis.

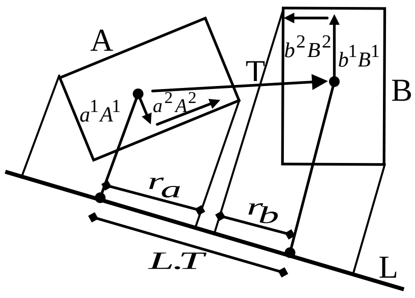


Figure 2. Separating Axis between two OBBs

The first algorithm that we will propose, uses learning of a perceptron to find the separating axis. It will search for a plane that separates the two boxes, using only 8 corner points of each box. The equation of the plane is given by:

$$ax + by + cz + d = 0 \tag{5}$$

Where $(a,b,c) \neq (0,0,0)$

The problem here is to find the coefficients a, b, c and d which determines the position and orientation of the plan, in order to separate the space into two parts: A part containing the first object and the other part containing the other object. The problem can be simplified by the following scheme:

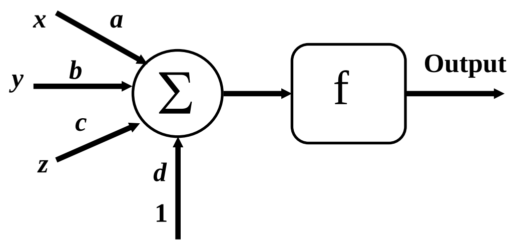


Figure 3. Descriptive Scheme of the Search for a Separator Plane

Where x, y and z are the coordinates of the points of the two objects. Some of them take the class "1" and others the class "0" depending on the classification function. This scheme is the same as that used in neural methods to solve the classification problems, where a, b, c and d are the weights, x, y and z are the inputs and f the transfer function (threshold function). This is the perceptron. The learning algorithm used is the back-propagation of the gradient.

5. Improved of the Detection Collision Time by Changing the Base Frame

The displacement of the reference in the middle of the two objects may result in an important improvement of the collision detection time. In fact, this displacement minimizes the space traversed by the plan and then the number of the learning iterations will be low (Figure 4).

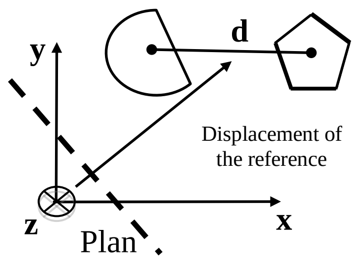


Figure 4. Displacement of the Base Frame During the Search of the Separation Plan

To search the separation plane, the following algorithm is applied:

- Displacement of the base frame in the middle of the two boxes;
- Computation of the sum: $S = ax + by + cz + d$;
- Computation of $f(s)$ where f is the threshold function;
- Compare the obtained result (O) with the desired result (D): If the result is the same, one does not change the weights, if it is different the weights a, b and c are changed as follows:
- w = w + (D - O) * input vector;
- d = d + (D - O);
- One passes to the next point;

At each iteration, the weights are modified to satisfy each of the points. When these weights are not modified by any of the points, the search is stopped. Then, there exists a plane that separates the two objects.

6. Triangle-Triangle Intersection Test

Detect if two objects are in collision can be done on two levels:

- Detection between including volumes (already treated);
- Exact detection of collision between triangle-triangle;

We present in this part the algorithm of exact detection of collision between triangle-triangle [15].

Intersection test method

Let $T1$ and $T2$ be two triangles; possess vertices V_0^1, V_1^1, V_2^1 and V_0^2, V_1^2, V_2^2 respectively. The plans on which the triangles are located are π_1 and π_2. The equation of the plan π_2: $N_2.x + d_2 = 0$.(Where x is any point on the plan) [15].

$$\begin{cases} N_2 = (V_1^2 - V_0^2) \times (V_2^2 - V_0^2) \\ d_2 = -N_2.V_0^2 \end{cases} \tag{6}$$

The distances between the vertices of $T1$ and π_2 are given by [15]:

$$d_{V_i^1} = N_2.V_i^1 + d_2 \,; i = 0,1,2 \tag{7}$$

If all the $d_{V_i^1} \neq 0$, $i = 0,1$ and 2, and all have the same sign, then $T1$ are beside p_2 and the overlapping is rejected [15]., The same calculation is done for $T2$ and p_1.

If all the $d_{V_i^1} = 0$, $i = 0,1$ and 2, then the triangles are coplanar, Otherwise, the intersection between p_1 and p_2 is a line, $L = O + tD$, Where $D = N_1 \times N_2$ is the direction of the line and O is some point on it [15].

Now, assume that we want to compute a scalar interval on L that represents the intersection between $T1$ and L, and that, for example, V_0^1 and V_2^1 lie on the same side of p_2 and that V_1^1 lies on the other side (if not, you have already rejected it). To find scalar values that represent the intersection between the edges $\overline{V_0^1V_1^1}$ and $\overline{V_1^1V_2^1}$ and L the vertices are first projected on to L : $P_{V_i^1} = D.(V_i^1 - O)$.

The equation can be simplified by [15]:

$P_{V_i^1} = D.V_i^1$, We do not need to calculate O .

$$P_{V_i^1} = \begin{cases} V_{ix}^1, si\left|D_x\right| = \max(\left|D_x\right|,\left|D_Y\right|,\left|D_Z\right|) \\ V_{iy}^1, si\left|D_y\right| = \max(\left|D_x\right|,\left|D_Y\right|,\left|D_Z\right|) \\ V_{iz}^1, si\left|D_z\right| = \max(\left|D_x\right|,\left|D_Y\right|,\left|D_Z\right|) \\ i = 0,1,2 \end{cases} \tag{8}$$

Then we want to compute a line parameter value, t_1 :

$$t_1 = P_{V_0^1} + (P_{V_1^1} + P_{V_0^1})\frac{d_{V_0^1}}{d_{V_0^1} - d_{V_1^1}} \tag{9}$$

Similar calculations are done to compute t_2, and an interval for $T2$ is computed as well. If these intervals overlap, the triangles intersect.

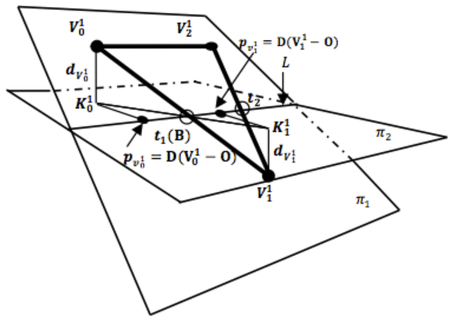


Figure 5. Collision Interval,[t1, t2] between the Line L and the Triangle T1

7. Control of the Manipulator Arms and Fingers

Figure 6 shows the diagram of the manipulator arm and the hand. The hand is considered in relation to the manipulator arm like a mass value M, it allows us to decompose the dynamic model in two parts, a dynamic model for the manipulator arm and a dynamic model for the hand.

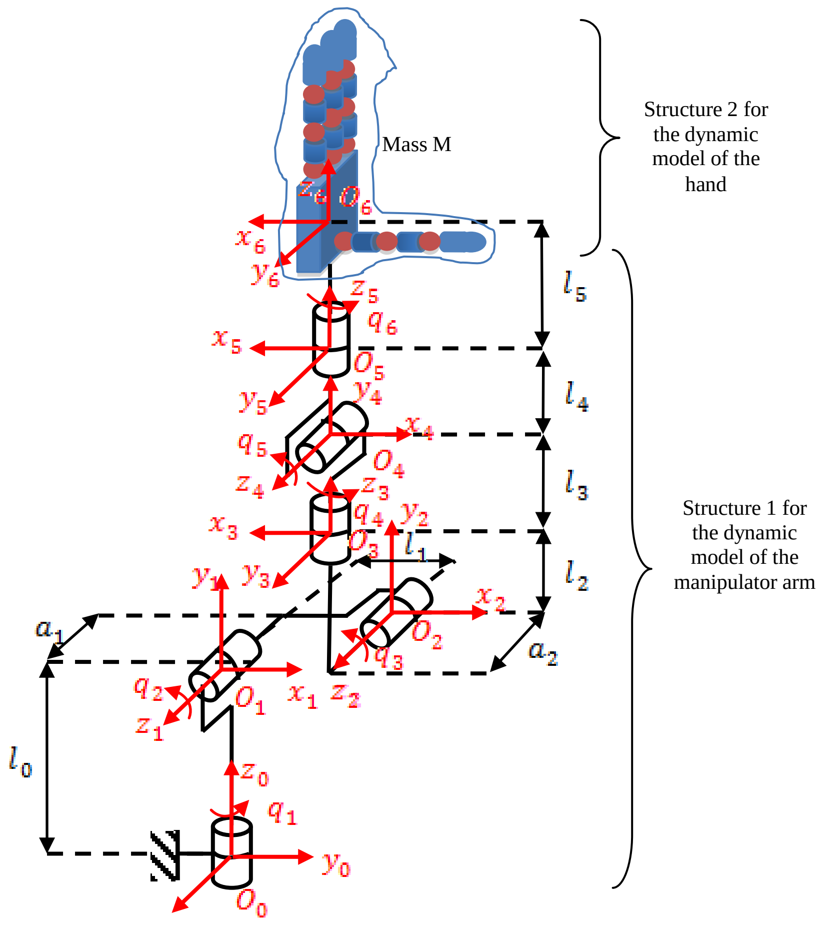

Mass M
Structure 2 for the dynamic model of the hand
Structure 1 for the dynamic model of the manipulator arm
z_6 O_6 x_6 y_6
z_5 q_6 x_5 O_5 y_5
y_4 x_4 q_5 O_4 z_4
z_3 q_4 y_2 x_3 O_3 l_1 y_3
y_1 x_2 O_2 q_3
a_1 a_2 q_2 O_1 x_1 z_2 z_1
l_0 l_2 l_3 l_4 l_5
z_0 q_1 y_0 O_0

Figure 6. Geometry of the Manipulator Arm and the Hand

The dynamic equation of the manipulator arm is given as follows:

$$[D(q)\ddot{q}+C(q,\dot{q})\dot{q}+g(q)=\tau]_{\text{Manipulator arm}} \tag{10}$$

The dynamic equation of the hand is given by the dynamic matrix of the fingers following:

$$\begin{bmatrix} [D(q)\ddot{q}+C(q,\dot{q})\dot{q}+g(q)=\tau]_{Doigt1} \\ [D(q)\ddot{q}+C(q,\dot{q})\dot{q}+g(q)=\tau]_{Doigt2} \\ [D(q)\ddot{q}+C(q,\dot{q})\dot{q}+g(q)=\tau]_{Doigt3} \\ [D(q)\ddot{q}+C(q,\dot{q})\dot{q}+g(q)=\tau]_{Doigt4} \end{bmatrix}_{Hand} \tag{11}$$

8. Linearization and Decoupling of the Dynamic Model

Figure 7 shows the block diagram of the control of the manipulator arm and fingers, the trajectory of the manipulator arm depends on the coordinates of the object and the trajectories of different fingers are calculated by the perceptron (calculated points collisions between each finger and object).

Control by inverse dynamics allows the cancellation of the nonlinear terms and the decoupling of the dynamics of each joint:

$$D(q)v + C(q,\dot{q})\dot{q} + g(q) = \tau \tag{12}$$

selection of the command v :

$$v = \ddot{q}_d + K_v(\dot{q}_d - \dot{q}) + K_p(q_d - q) \tag{13}$$

We replace equation (13) in the equation (12):

$$D(q)\left[\ddot{q}_d + K_v(\dot{q}_d - \dot{q}) + K_p(q_d - q)\right] + C(q,\dot{q})\dot{q} + g(q) = \tau \tag{14}$$

We replace the equatuion (14) in the eqauation (10):

$$\ddot{e}_q + K_v\dot{e}_q + K_p e_q = 0 \tag{15}$$

With :

$$e_q = q_d - q \tag{16}$$

The error dynamic equation is stable for a suitable choice of K_v and K_p matrices.

Another choice of the command:
$v = \text{fuzzy control}$

The block of the fuzzy controller has two inputs, *e* (error) and *de* (the variation of the error) and an output *du* where:

$$e = q_d - q \tag{17}$$

$$de = \frac{de(t)}{dt} = \frac{\Delta e}{\Delta t} = \frac{e(t) - e(t-T)}{t-T} \tag{18}$$

Where T is a time constant.

- The input e has two values (N, P);
- The input de has two values (N, P);
- The output du has three values (N, P, ZE);

Where:

N: set of negative values.

P: set of positive values.

ZE: set of zero values.

The membership functions for the inputs e, de and for the output du are shown in Figure 8.

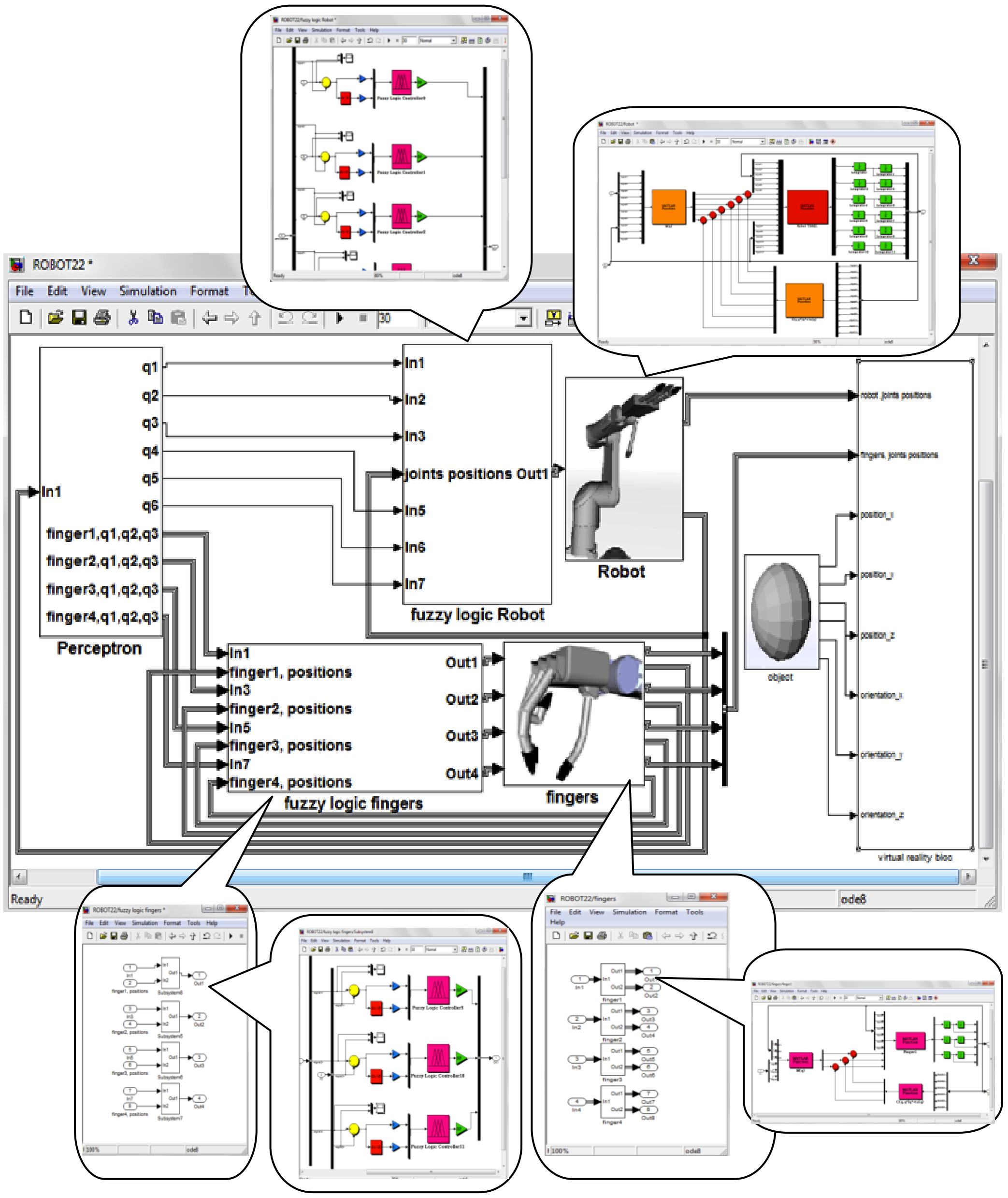


Figure 7. Schematic of the Control Loop

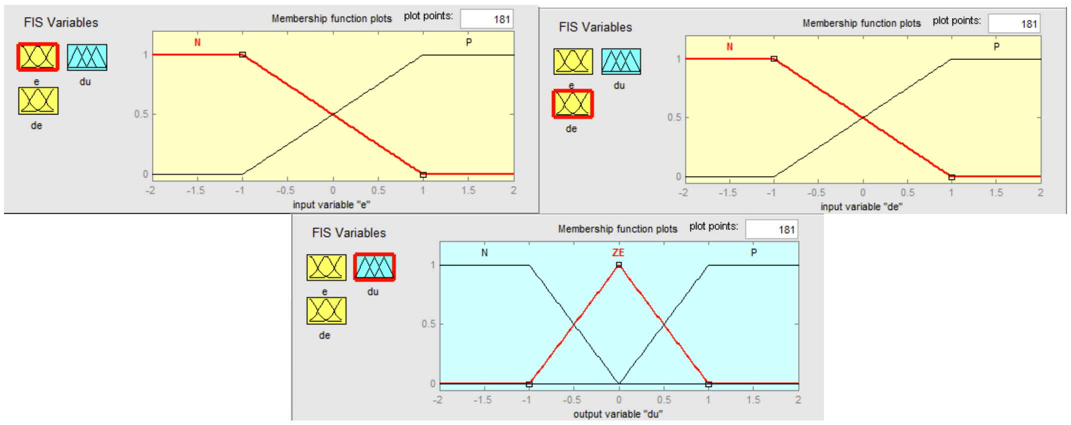


Figure 8. Membership Functions of the Inputs: e, de and Output du

Figures 9,10, shows the results of the application of the fuzzy control on the manipulator arm and the hand (results of a single finger).

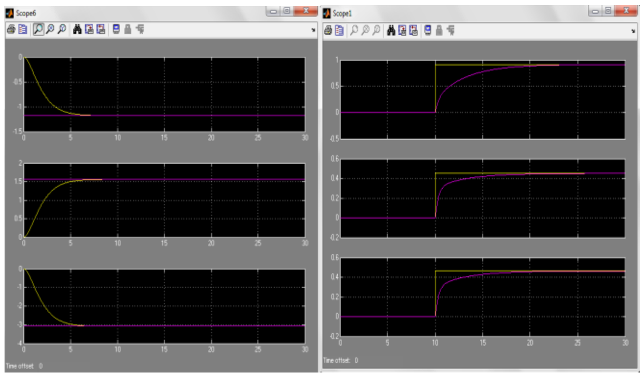


Figure 9. The Rotation Angles of the Joints 1-3 (in radians) for the Manipulator Arm and One Finger of the Hand

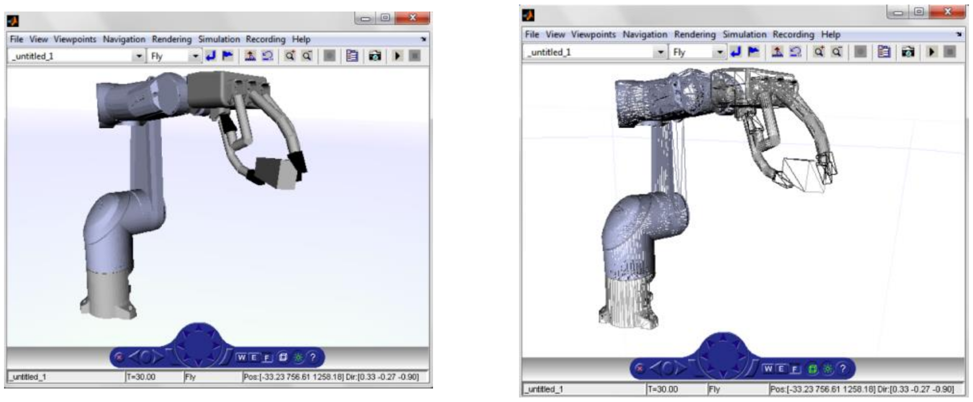

File View Viewpoints Navigation Rendering Simulation Recording Help
_untitled_1
Fly
T=30.00
Pos:[-33.23 756.61 1258.18] Dir:[0.33 -0.27 -0.90]

Figure 10. Result of the Grasping of the Object

9. Simulation Results of the Collision Detection

The previously presented algorithm was tested on two types of hands: The first is formed only by parallelepipeds and the second by triangles. Figure 11 shows the hand and the object, that we have created, which are formed by parallelepipeds.

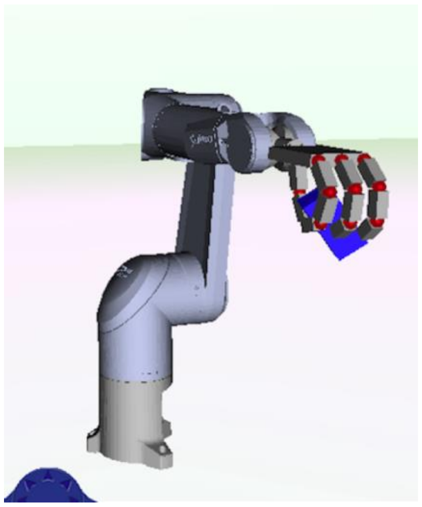
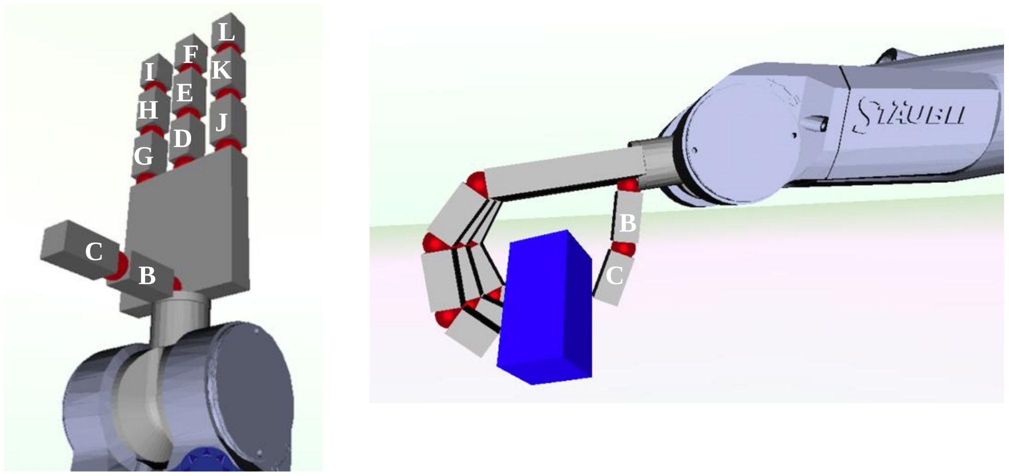


Figure 11. Representation of the Hand and the Object Formed by Parallelepipeds

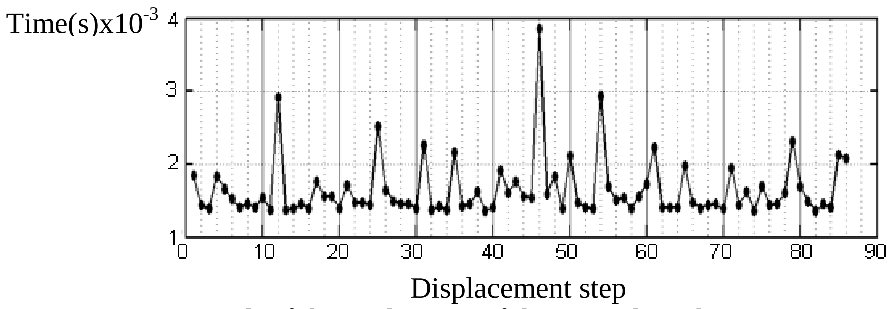
The algorithm SAT and our perceptron learning algorithm with displacement of base frame, will be confronted in the detection of collision time during the grasp of an object by a robotic hand. The simulation of the grasp of the object is performed with a step of 0.003rad. The maximum number of iterations, to find the separating axis, is fixed at 10000. Figure 12 shows the different fingers used for the grasp of the object.



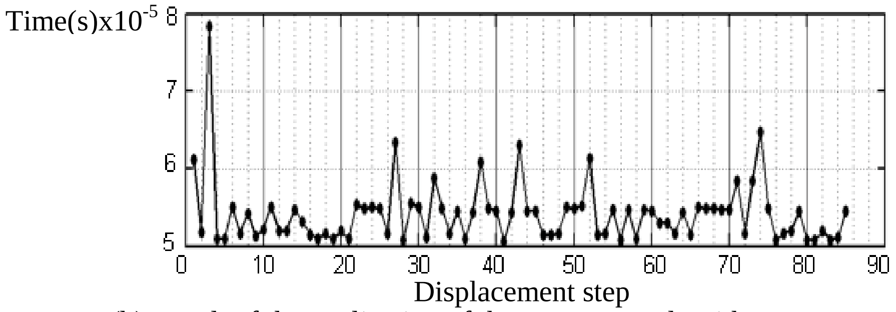
(a) The different fingers of the hand (b) Grasp of an object

Figure 12. The Hand and Grasping Operation

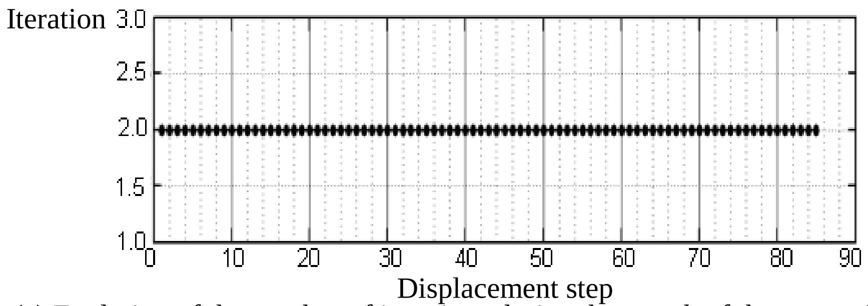
For the grasp of the same object, Figure 13 and 14 show the results of the application of the algorithms, SAT and perceptron learning with displacement of the base frame, on the segments B and C which form a finger.



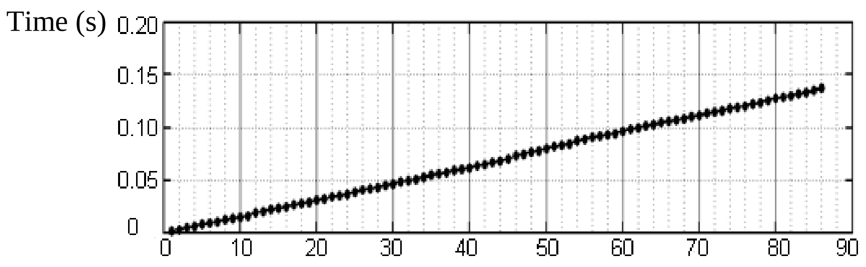
(a) Result of the application of the SAT algorithm on segment B



(b) Result of the application of the perceptron algorithm on segment B

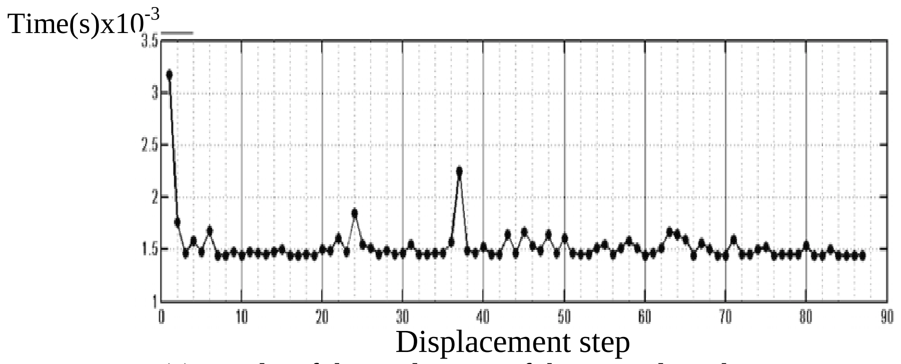


(c) Evolution of the number of iterations during the search of the separation plan when learning the perceptron

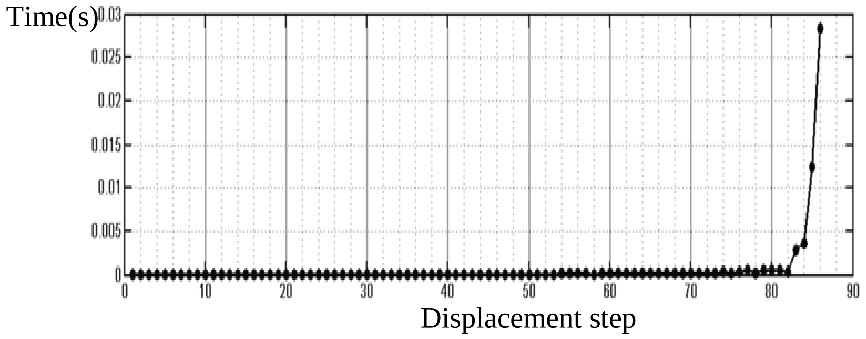


(d) Evolution of the saved time by the perceptron algorithm

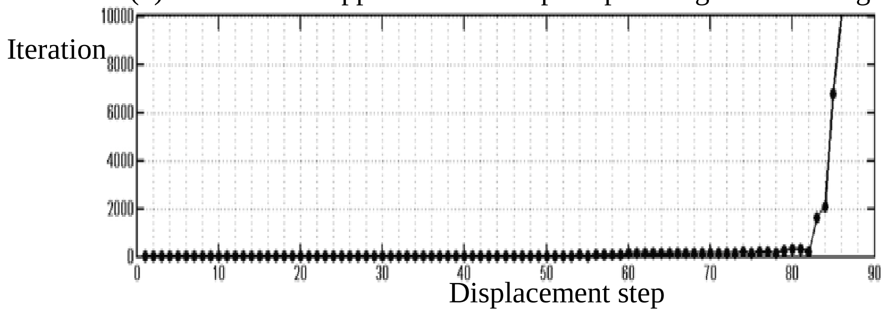
Figure 13. Results of the Application of the SAT and Perceptron Algorithms on Segment B



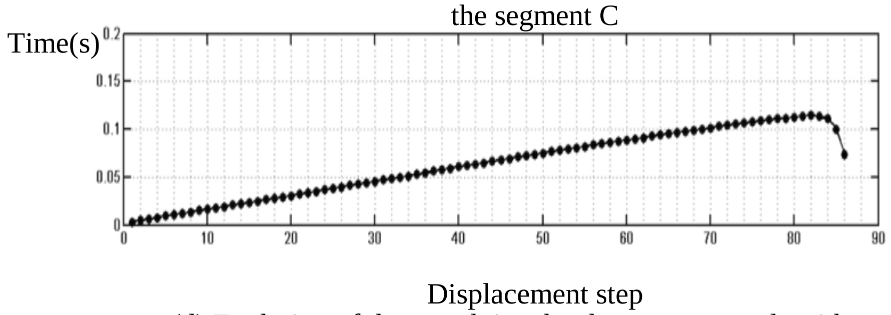
(a) Results of the application of the SAT algorithm on segment C



(b) Result of the application of the perceptron algorithm on segment C



(c) Evolution of the number of iterations as a function of displacement step for the segment C



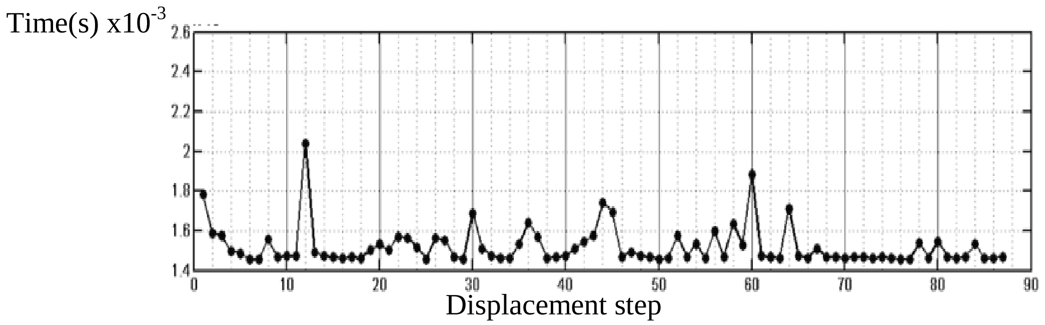
(d) Evolution of the saved time by the perceptron algorithm

Figure 14. Results of the Application of the SAT and Perceptron Algorithms on Segment C

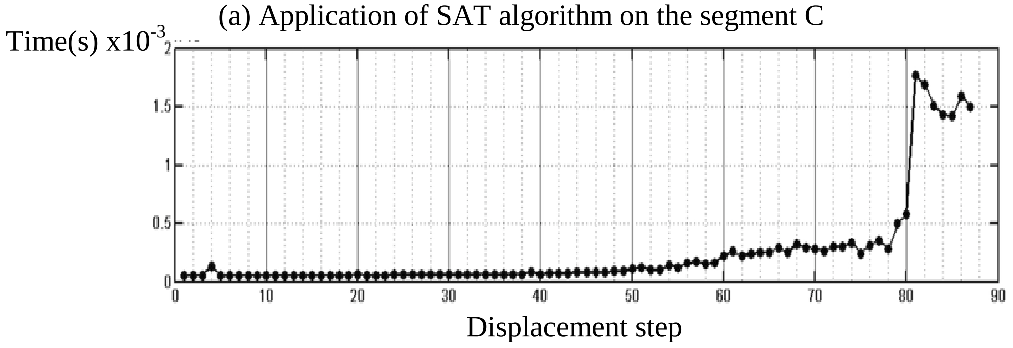
According to the curves of Figure 13 and 14, one finds that the perceptron algorithm with displacement of base frame improves the collision detection time. In the case of segment B, the saved time increases, according to the number of steps (Figure 13.d), thus the number of iterations is low (Figure 13.c) as the segment B is still far from the object (Figure 12). In the case of segment C, the saved time increases up to step 82 where the segment C is very close to the object. It remains only 0.8594°, which corresponds to 5 steps, before the collision occurs. After the step 82, the time is decreasing due to the large number of iterations needed to find the separation plane (Figure 14.c). To solve this problem, another algorithm based on the cooperation between the two algorithms SAT and perceptron is developed.

10. Neuro-SAT Algorithm

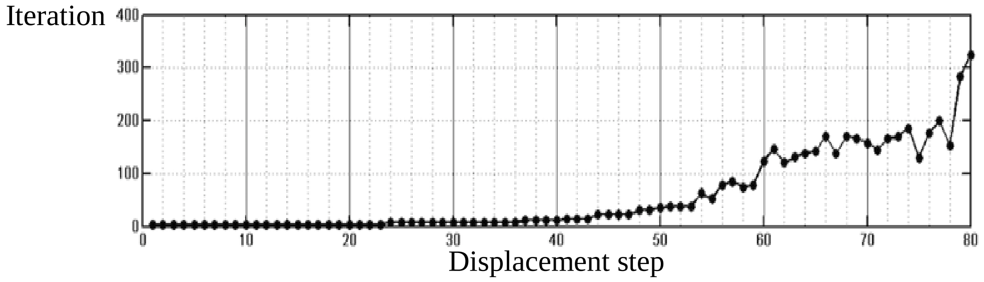
The application of the perceptron algorithm with displacement of base frame followed by the SAT algorithm can also result in a significant improvement in the time of collision detection. The application of this sequence of algorithms will do the object of the following discussion and the corresponding algorithm will be called Neuro-SAT. This is the second algorithm that we have developed. We will first apply the perceptron algorithm and when the elapsed time by this algorithm tends to be large, we switch to the SAT algorithm. Switching between the two algorithms is provided by the number of iterations: more large is this number, more large is the collision detection time (Figure 14 b and c). The switching threshold, from the perceptron algorithm to the SAT algorithm, is set at 300 iterations (Figure 15.c). This number is empirically obtained. When the switching occurs, one finds that the saved time remains almost constant (Figure 15.d).



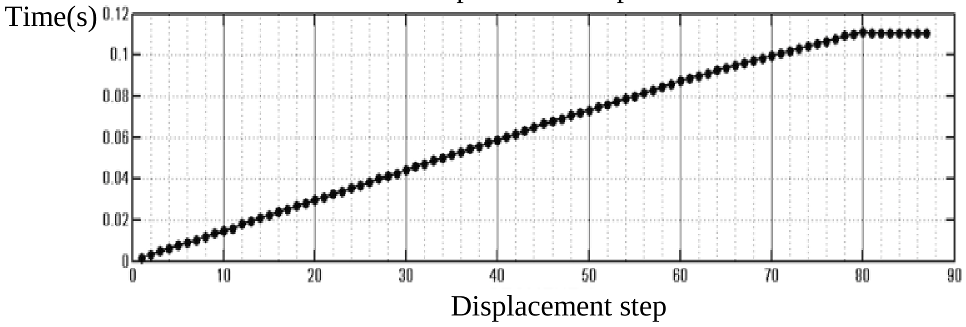
(a) Application of SAT algorithm on the segment C



(b) Result of application of Neuro-SAT algorithm on the segment C



(c) Evolution of the number of iterations according to the displacement step



(d) Evolution of the saved time by the perceptron algorithm

Figure 15. Results of the Application of SAT and Neuro-SAT Algorithms on the Segment C

11. Implementation of the Algorithms to a Hand and an Object Formed by Triangles

We will use in this part a hand and an object formed by triangles, to compare the three collision detection algorithms namely: the SAT algorithm, Neuro-SAT algorithm and perceptron learning algorithm with displacement of the base frame. The Figure 16 illustrates the hand and the object used.

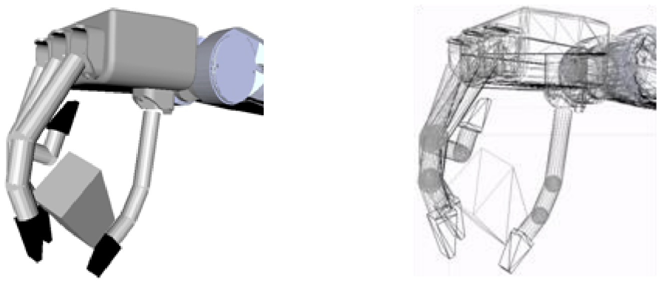


Figure 16. View of the Hand and the Object Used to Compare the Three Algorithms

To test the collision between the hand and the object, we must create the tree of the boxes OBBs, in order to apply the different algorithms on these boxes which include the fingers of the hand and the object. Figure 17 shows the boxes OBBs which include a part of the finger of the hand.

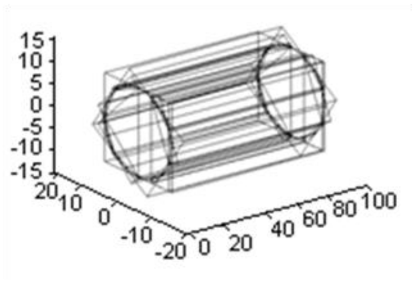
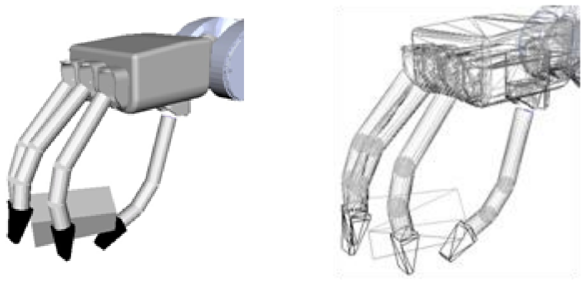


Figure 17. The Boxes OBBs which Include a Part of the Finger of the Hand

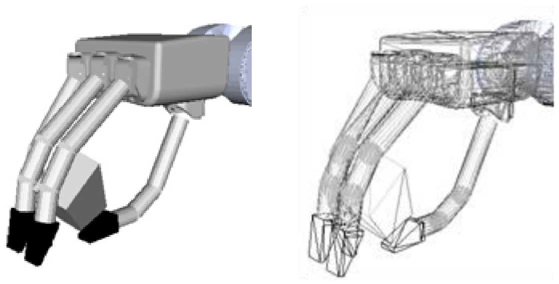
Table 1 shows the results, as to the execution time, between the three previously cited algorithms. All developments are done with MATLAB® software. The deference in the collision detection time, between these algorithms, is remarkable. The perceptron learning algorithm with displacement of the base frame converges faster than the other algorithms, because it makes a prediction of collision during the descent in the tree of the boxes OBBs by using only the eight corner points of each OBB. Before the collision occurs, one descends in the tree towards the base where another triangle-triangle test will be triggered to determine the exact point of collision. The prediction depends on the number of iterations required to train the perceptron. This number is set to a maximum value of 1500 iterations in our application. Another factor that has allowed the perception algorithm to give the best results, is the displacement of the base frame in the middle of the distance between the boxes of the fingers and boxes of the object, during the perceptron learning. For Neuro-SAT algorithm, while the fingers are distant from the object, it is the perceptron learning algorithm with displacement of base frame which is applied. This will save time, since the research of the separation plan does not require a large number of iterations, set at a maximum of 300 iterations in our application (see the results of Figure 18.c in Table 1). In this case there exists a finger which does not collide with the object, hence the gain of the time. In the contrary case, where the finger and the object are close to each other, the number of iterations will increase to determine if the boxes of the fingers on one side and the boxes of the object on the other side are colliding, then we switch to the SAT algorithm.

Table 1. Collision Detection Time for Each Algorithm (in seconds)

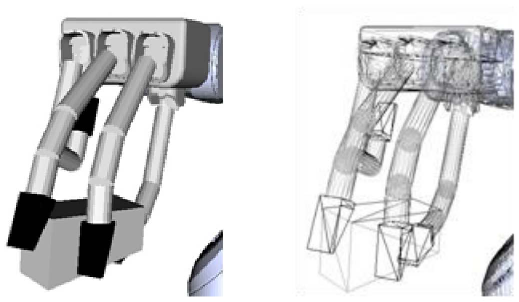
	SAT	Neuro-SAT	Perceptron
1st example	75.18	73.10	57.33
2nd example	114.75	112.08	63.87
third example	91.37	76.95	67.56



t(SAT)   = 75.18 Sec
t(Neuro-SAT) =73.10 Sec
t(Perceptron) = 57.33 Sec



t(SAT)   = 114.75 Sec
t(Neuro-SAT) = 112.08 Sec
t(Perceptron)   = 63.87Sec.



t(SAT)   = 91.37 Sec
t(Neuro-SAT) = 76.95 Sec
t(Perceptron)   = 67.56 Sec

Figure 18. Different Grasping of an Object

Figure 19 shows the result of the application of the perceptron learning algorithm, with displacement of base frame, to more complex objects.

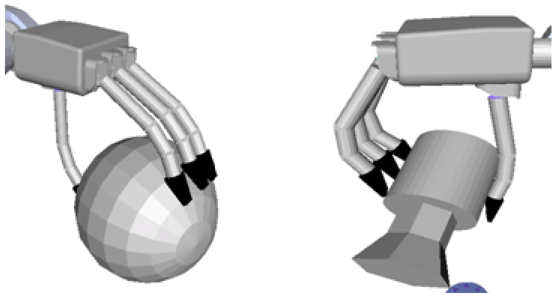


Figure19. Result of the Application of the Perceptron Learning Algorithm

12. Conclusion

In this work we have presented two new approaches to the problem of collision detection. In the first approach, we developed an algorithm that used a simple perceptron with displacement of base frame to find the separation plane, between the boxes OBBs which include the fingers and the object, by separating the 8 points of the corners of these boxes. The time collision detection is significantly improved compared to the conventional method SAT. This is due to the fact that this algorithm does not search to find the collision between the boxes OBBs, during the descent in the tree of the boxes, but it makes only a prediction. Some steps before that the collision takes place, one descends in the tree that which will save time.

In the second approach, we developed an algorithm called Neuro-SAT. It allowed also an improvement of the collision detection time, compared to the SAT algorithm. In this algorithm, we use first the perceptron algorithm with displacement of base frame, as the object is far from the fingers, which implies a reduced iterations number required to find the separating axis. A switching threshold, from the perceptron algorithm to the SAT algorithm is fixed and when this switchover occurs, we will start with a gain of time, compared to running the SAT algorithm alone. This gain remains constant during the search of the collision. The best results, among these three algorithms, correspond, however, to the algorithm of training the perceptron with displacement of the base frame.

References

[1] P.M. Hubbard, "Collision detection for interactive graphics applications," Visualization and Computer Graphics, vol.1, no. 3, **(1995)**, pp.218-230.
[2] P.M. Hubbard, "Approximating polyhedra with spheres for time- critical collision detection," ACM Transactions on Graphics, vol.15, no.3, **(1996)**, pp. 179-210.
[3] G. van den Bergen, "Efficient Collision Detection of Complex Deformable Models using AABB Trees," Journal of Graphics Tools, vol. 2, **(1997)**, pp. 1-13.
[4] S. Gottschalk, M. C. Lin, D. Manocha, "OBB-Tree: A Hierarchical Structure for Rapid Interference Detection," Proceedings of SIGGRAPH, vol. 96, **(1996)**, pp. 171-180.
[5] W. Xiao-rong, W. Meng, L. Chun-gui, "Research on Collision Detection Algorithm Based on AABB", Natural Computation, **(2009)**, pp.422 – 424.
[6] C. Tu, L. Yu, "Research on collision detection algorithm Based on AABB-OBB Bounding Volume," Education Technology and Computer Science, vol.1, (2009), pp. 331-333.

[7] F. Zhigang, J Jianxun, X Jie,W Xiaochi, " Efficient collision detection using bounding volume hierarchies of OBB-AABBs and its application," Computer Design and Applications (ICCDA), vol.5, (2010), pp. 242 - 246.

[8] N. Greene, "Detecting Intersection of a Rectangular Solid and a CoNvex Polyhedron. P.S. Heckbert", **(1994)**, pp. 74-82.

[9] G. Zachmann, "Exact and Fast Collision Detection. Diploma thesis, Fraunhofer Institute for Computer Graphics", Technische Hochschule Darmstadt, Fachbereich Informatik, **(1994)**.

[10] T. Kay, J. Kajiya, "Ray Tracing Complex Scenes", Computer Graphics (SIGGRAPH 1986 Proceedings), vol. 20, no. 4, **(1986)**, pp. 269–278.

[11] S. Omohundro, "Five Balltree Construction Algorithms", Technical Report TR-89-063, International Computer Science Institute, **(1989)**.

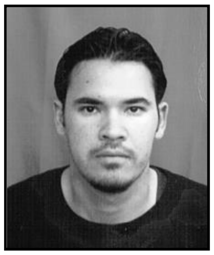
[12] G. Barequet, S. Har-Peled, "Efficiently Approximating the Minimum-volume Bounding Box of a Point Set in Three Dimensions", Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, **(1999)**, pp. 82–91.

[13] J. O'Rourke, "Finding Minimal Enclosing Boxes", International Journal of Computer and Information Sciences, vol. 14, no. 3, **(1985)**, pp. 183–199.

[14] D. Eberly, "Polyhedral Mass Properties (Revisited)", Technical Report, Magic Software, **(2003)**.

[15] T.Möller. "A Fast Triangle-Triangle Intersection Test" ,Journal of Graphics Tools, vol. 2, no. 2, **(1997)**, pp.25-30.

Authors



Redha Fourar was born in Batna, Algeria in 1983. He received his Engineering Master degree and magister in Robotics from University of Batna, Algeria. Currently, he is a PhD student. His interests include digital image processing, robotics and artificial intelligence, signal processing.

Djamel Melaab received his Ph.D. in Automatic from the University of Bordeaux, France in 1991. He is currently lecturer at the University of Batna, Algeria. He works in the field of artificial intelligence, and Robotics. His current research interests mainly include the application of artificial intelligence techniques in the field of biometric recognition, Biomedical Engineering and Robotics.