Neural Comput & Applic
DOI 10.1007/s00521-011-0641-3

ORIGINAL ARTICLE

Trajectory planning and collisions detector for robotic arms

José de Jesús Rubio · Enrique García · Jaime Pacheco

Received: 7 February 2011 / Accepted: 4 May 2011

Abstract The major contributions of this paper are as follows: (1) the Gilbert–Johnson–Keerthi (GJK) algorithm is a collisions detector algorithm, a modified Gilbert–Johnson–Keerthi algorithm is presented, the proposed GJK algorithm uses a different distance, (2) some examples of GJK algorithm are presented, in the last example, the GJK distance algorithm is used to detect the collisions of a camera with its environment inside of a warehouse, the camera cannot cross any part of the structure of the warehouse, the camera needs to go around the structure, when the camera touches the structure, the camera goes to the right or to the left, (3) the time used in a cycle of work of the transelevator robotic arm is presented, it can be extended to other kind of robotic arms, (4) some examples of the time used in a cycle of work are presented, in the least example, the algorithm is used to control the time needed for the transelevator to go from one place to other one, (5) this paper presents a new trajectory planning algorithm which divides the trajectory in n periods, when n is equal to 2, the proposed algorithm is the same as other algorithms, but for n higher than 2, the proposed algorithm gives other optional trajectories, so the proposed algorithm lets the designer to take a better trajectory than with the previous algorithms, (6) some examples of the proposed trajectories planning algorithm are presented, in the least example, the proposed trajectory planning algorithm is used to control the movements of a transelevator inside of a warehouse.

J. d. J. Rubio (✉) · E. García · J. Pacheco
Sección de Estudios de Posgrado e Investigación,
ESIME Azcapotzalco, Instituto Politécnico Nacional,
Av. de las Granjas no.682, Col. Santa Catarina,
02250 Mexico, D.F., Mexico
e-mail: jrubioa@ipn.mx

Keywords Collisions detector algorithm · Distance algorithm · Cycle of work algorithm · Trajectory planning algorithm

1 Introduction

There is some research about robotics as is [1, 6, 8, 9, 18, 19, 21, 23, 24, 31], and [32]. In, [6, 18, 19, 21, 23, 24, 32 32], they propose control algorithms and in [1, 9, 31] they propose obstacle avoidance algorithms.

In robotics fields as control and obstacle avoidance, it is important to know if two objects, characterized by mathematical models in three dimension space, intersect or are in near proximity [11].

The Gilbert–Johnson–Keerthi algorithm is especially fit for use in collisions detection of objects modeled using various types of geometric primitives, such as boxes, cones, and spheres, and their images under affine transformation [2].

There is some research about the proximity of two objects as is [2, 10, 11], and [33].

The paper of [2] presents an implementation of the Gilbert–Johnson–Keerthi algorithm for computing the distance between convex objects that has improved performance, robustness, and versatility over earlier implementations. In the paper of [10], they focus on objects represented by convex polyhedra, which are defined as the convex hull of points in three-dimensional space. Although the proposed approach can be applied for convex polytopes (bounded polyhedra) in three-dimensional space, a wider class of objects is permitted, since it is also possible to treat conveniently nonconvex shapes as a union of convex polytopes. In [11], an efficient and reliable algorithm for computing the euclidean distance between a pair of convex

Published online: 18 May 2011

sets is described. In the article of [33], the collisions of a robot with its environment is studied. In normal applications of a robot arm, a collision takes place because of the velocity of the end effector relative to the object at the time of contact. The collision has effects on the velocities and internal forces of the robotic system.

The aforementioned papers are interesting, but none explains how to obtain the distance of the GJK algorithm, this paper uses a modified Gilbert–Johnson–Keerthi algorithm because it uses the distance algorithm given by [4]. In addition, none has studied the collisions of a camera with its environment as in this paper.

In this paper, the time used in a cycle of work of the transelevator robotic arm given by [5, 29] is used. It can be extended to other kind of robotic arms.

In robotics fields as control and obstacle avoidance, it is important to propose a trajectory planner where it computes a function that completely satisfies the desired motion of the robot as it traverses the path [28].

There is some research about trajectory planning as is [7, 12, 17, 22], and [30].

Typically the trajectories are described by a nonlinear model problem introduced in the article of [7]. Stability of a switched system that consists of a set of linear time invariant subsystems and a periodic switching rule is investigated in [12]. In [17], they present a novel approach for solving the trajectory planning problem (TPP) in time-varying environments. The paper of [22] presents a study of special trajectories attainment for mobile robots based on the dynamic features of chaotic systems. In [30], an alternative formulation of the discrete trajectory planner is also provided for the efficient generation of optimal trajectories when equal time intervals are required.

The aforementioned papers are interesting, but none is interested to control the time needed to reach the velocity stationary period in the trajectory. The proposed trajectory planner algorithm is like a generalization of the algorithm proposed by [28].

In this paper, a modified Gilbert–Johnson–Keerthi algorithm is presented, the proposed GJK algorithm uses a different distance, and some examples of GJK algorithm are presented; in the last example, the GJK distance algorithm is used to detect the collisions of a camera with its environment inside of a warehouse; the camera cannot cross any part of the structure of the warehouse; the camera needs to go around the structure; when the camera touches the structure, the camera goes to the right or to the left. The time used in a cycle of work of the transelevator robotic arm is presented, it can be extended to other kind of robotic arms, some examples of the time used in a cycle of work are presented; in the least example, the algorithm is used to control the time needed for the transelevator to go from one place to other one. This paper presents a new trajectory planning algorithm which divides the trajectory in n periods, when n is equal to 2, the proposed algorithm is the same as other algorithms, but for n higher than 2, the proposed algorithm gives other optional trajectories, so the proposed algorithm lets the designer to take a better trajectory than with the previous algorithms, some examples of the proposed trajectories planning algorithm are presented, in the least example, the proposed trajectory planning algorithm is used to control the movements of a transelevator inside of a warehouse.

2 Collisions detector algorithm

Sometimes, it is needed to know when an object called object 1 touches another one called object 2.

The algorithm to get the distance between convex polyhedral models of [10] is an iterative method which can be used as a collision detector using many objects as are cubes, cones, and spheres.

$A, B \in \Re^3$ are objects modeled by convex bi-dimensional polygons x_A and x_B with vertex $Q_A = \{1, \ldots, m_A\}$, $Q_B = \{1, \ldots, m_B\}$, the Euclidean distance $d(A, B)$ between x_A and x_B is obtained. The algorithm developed by Gilbert, Johnson, and Keerthi (GJK distance algorithm) is as follows:

Let X be the Minkowski difference denoted as $X = x_A - x_B$, the distance between the objects A and B is given as follows:

$$d(A, B) = \min\{\|x_A - x_B\| : x_A \in A, x_B \in B\} \quad (1)$$

The objective is to find the distance and the nearer point to the origin of the coordinate system. The Fig. 1a shows a person which is near to touch a load. The Fig. 1b shows that the person is inside of a hexagon and the load is inside of a cube. The Fig. 1c shows the result of $x_A - x_B$ as a new polygon of ten vertex, 8 vertex are seen, and 2 are over written because the total number of vertex is $m = m_A + m_B$. The distance to the nearer point of $x_A - x_B$ to the origin is 0.5 which correspond to the nearer distance from the object A to the object B.

The Fig. 2 explains the GJK distance algorithm. The GJK distance algorithm is given as follows:

1. From the polygon $x_A - x_B$, consider a set of points $Q = \{Q_0, Q_1, Q_2\}$.
2. The minimum distance between the triangle $Q = \{Q_0, Q_1, Q_2\}$ and the origin, it is denoted as P.
3. If P is in the origin, the algorithm ends and returns $d(A,B) = 0$.
4. If P is between Q_1, Q_2, consider $Q = \{Q_1, Q_2\}$, else if P is between Q_0, Q_2, consider $Q = \{Q_0, Q_2\}$, else if P is between Q_0, Q_1, consider $Q = \{Q_0, Q_1\}$.

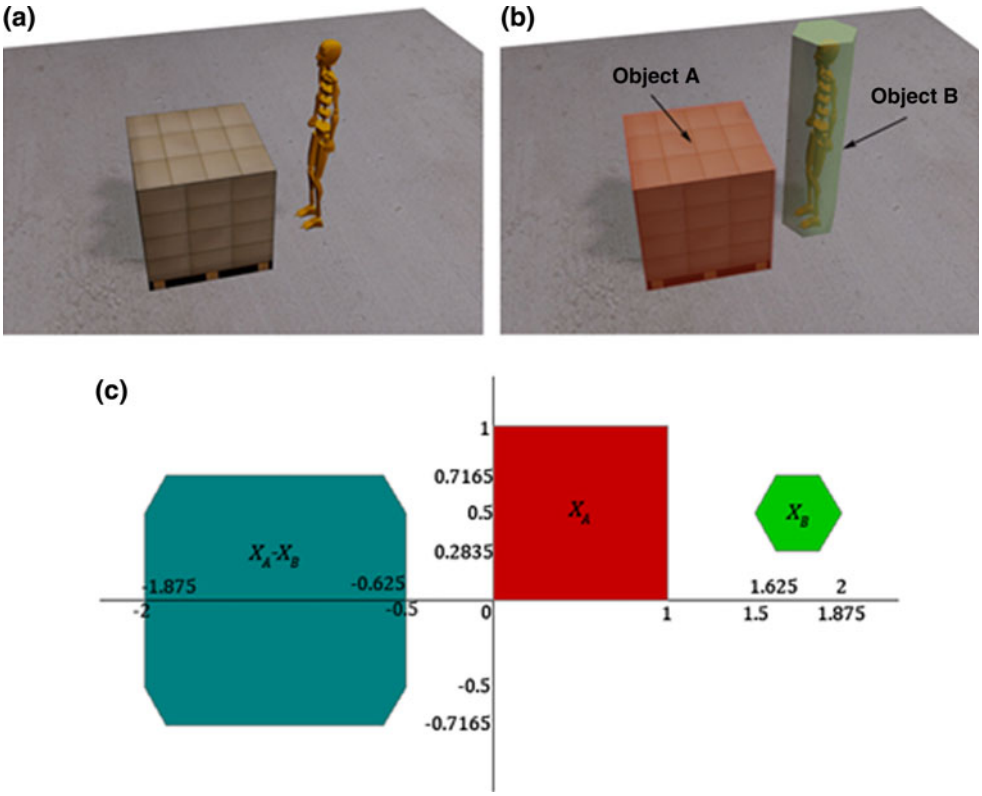

Fig. 1 One example of the Minkowski difference

5. The vertex V is the nearest point to the origin selected between the points different to Q_0, Q_1, and Q_2. Obtain $f(V)$ as a support function parallel to Q in a selected vertex V.
6. If $f(V)$ finds P, the algorithm ends and returns $d(A, B) = \|P\|$.
7. Include V to the triangle $Q = \{Q_1, Q_2, V\}$, or $Q = \{Q_0, Q_2, V\}$, or $Q = \{Q_0, Q_1, V\}$ depending of the point 4, return to the point 2.

The Fig. 3 shows the distance between the triangle $Q = \{Q_0, Q_1, Q_2\}$ and the origin (0,0) used in the second point of the GJK distance algorithm.

Let us to consider a_1, a_2, and a_3 the vectors from the origin to the points $P_1 = Q_0$, $P_2 = Q_1$, and $P_3 = Q_2$, respectively. It is needed to find the nearest side of the triangle to the origin, it is found considering the two smallest vectors, for this case a_1 and a_3 are the two smallest vectors. Let us define $P_{L1}(x_{L1},y_{L1})$ and $P_{L2}(x_{L2},y_{L2})$ as the two points of the nearest side of the triangle to the origin, for this case $P_{L1} = P_1$ and $P_{L2} = P_3$.

The Fig. 4 let us find the point of the smallest distance between the triangle $Q = \{P_1, P_2, P_3\}$ and the origin (0,0).

The equation of a line which is defined trough two points P_{L1} (x_{L1},y_{L1}) and $P_{L2}(x_{L2},y_{L2})$ is:

$$P = P_{L1} + u(P_{L2} - P_{L1}) \tag{2}$$

The point $P(x, y)$ is closest point of the line tangent to the line P_{L1}, P_{L2} which passes trough the origin (0,0), the dot product of the two tangent lines is zero, it is given as follows:

$$(0 - P)(P_{L2} - P_{L1}) = 0 \tag{3}$$

Substituting $P(x, y)$ of (2) into (3) gives:

$$(-P_{L1} - u[P_{L2} - P_{L1}])(P_{L2} - P_{L1}) = 0 \tag{4}$$

Solving this gives the value of u as follows:

$$u = \frac{x_{L1}^2 + y_{L1}^2 - x_{L1}x_{L2} - y_{L1}y_{L2}}{\|P_{L2} - P_{L1}\|^2} \tag{5}$$

where $\|P_{L2} - P_{L1}\|^2 = x_{L1}^2 + y_{L1}^2 - 2(x_{L1}x_{L2} + y_{L1}y_{L2}) + x_{L2}^2 + y_{L2}^2$.

Finally, the distance from the origin to the point $P(x, y)$ is found using (2) as follows:

$$\begin{aligned} x &= x_{L1} + u(x_{L2} - x_{L1}) \\ y &= y_{L1} + u(y_{L2} - y_{L1}) \\ d &= \sqrt{x^2 + y^2} \end{aligned} \tag{6}$$

The (5) and (6) are used to find the distance from the origin (0,0) to the point $P(x, y)$ which is used in the point 2 of the GJK distance algorithm.

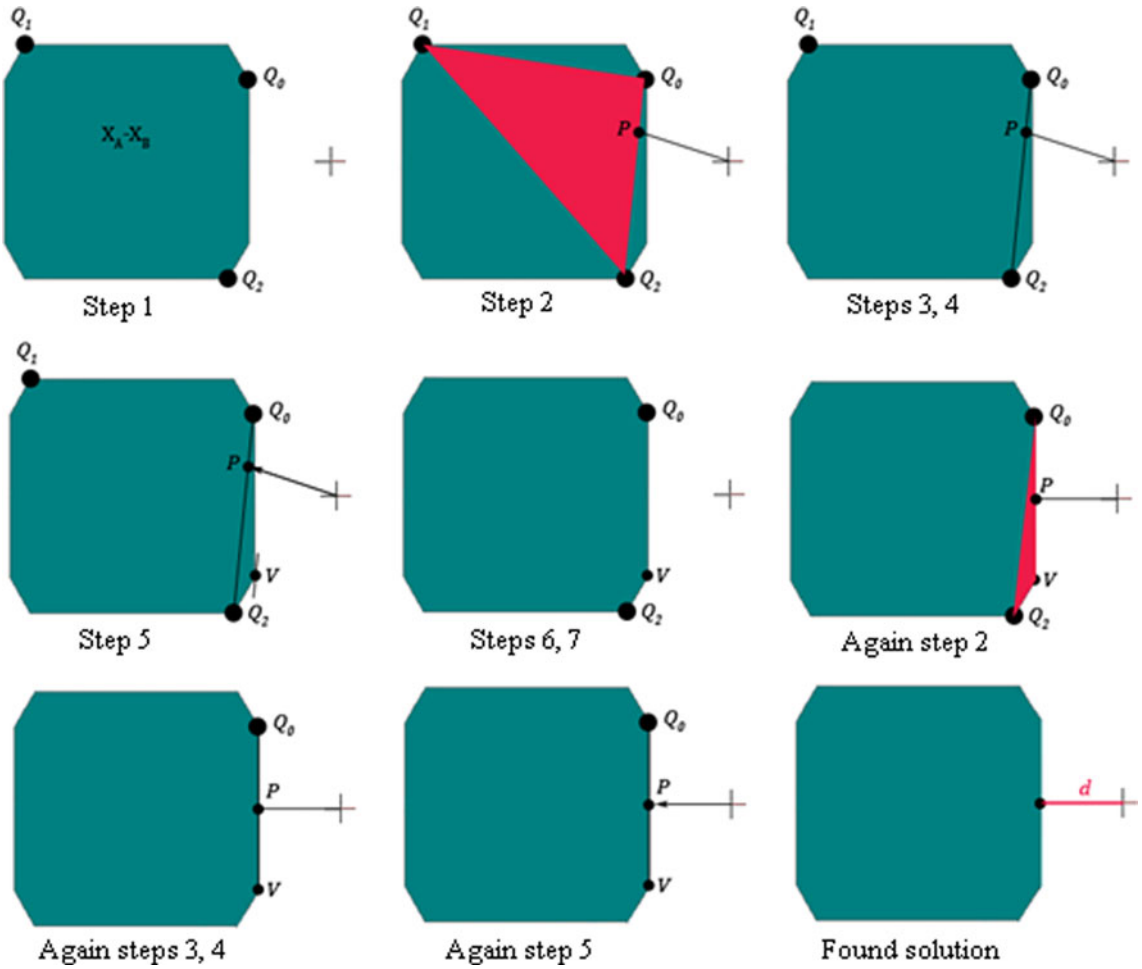

Fig. 2 One example of the GJK distance algorithm

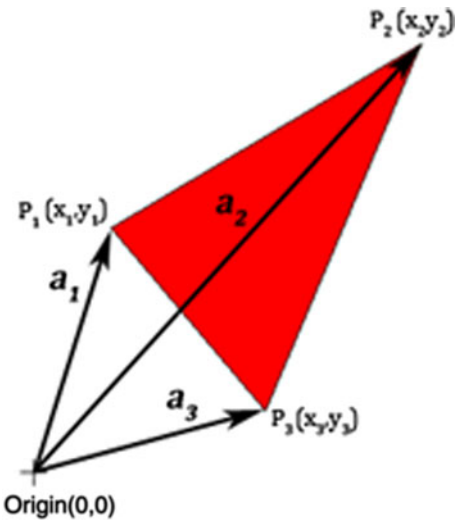

Fig. 3 The second point of the GJK distance algorithm

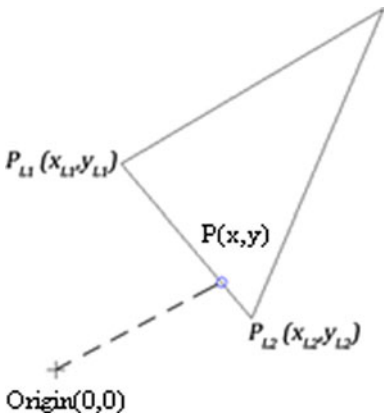

Fig. 4 The smallest distance between the triangle and the origin

Remark 1 The papers [2, 10, 11, 33] propose algorithms to find the proximity of two objects, but none explains how to obtain the distance of the GJK algorithm, this paper uses a modified Gilbert–Johnson–Keerthi algorithm because it uses the distance algorithm given by [4]. In addition, none has studied the collisions of a camera with its environment as in this paper.

3 The cycle of work algorithm used in a Cartesian or a transelevator robotic arm

The cycle of work of a Cartesian or a transelevator robotic arm is the time needed to take or to place a load from the home place; the time used for the cycle of work of a robotic arm denoted as T_{SC} is given as follows [5, 29]:

$$T_{SC} = 2E + 2T_{L/U} \tag{7}$$

Where $T_{L/U}$ is the time used to take or to place the load when the arm is nearest to the load, E is the time used to go from the home place to the load or from the load to the home place.

Let us define X as movement of the robotic arm in the horizontal direction, Y as movement of the robotic arm in the vertical direction, T_X as the time used for the movement in the horizontal direction, T_Y as the time used for the movement in the vertical direction. v_X as the velocity used for the movement in the horizontal direction and v_Y as the time used for the movement in the vertical direction in the go and the back. Then the time required in the go and the back is given as:

$$\begin{aligned} T_X &= \frac{X}{v_X} \\ T_Y &= \frac{Y}{v_Y} \end{aligned} \tag{8}$$

Let us define x as the movement of the robotic arm in the horizontal direction, y as movement of the robotic arm in the vertical direction, t_x as the time used for the movement in the horizontal direction, t_y as the time used for the movement in the vertical direction. v_x as the velocity used for the movement in the horizontal direction and v_y as the time used for the movement in the vertical direction in go or in the back. Then the time required in the go or in the back is given as:

$$\begin{aligned} t_x &= \frac{x}{v_x} \\ t_y &= \frac{y}{v_y} \end{aligned} \tag{9}$$

The time required in the go or in the back is obtained as [29]:

$$\begin{aligned} &\frac{1}{T_X T_Y}\int_0^{T_X}\int_0^{T_Y} \max\{t_x, t_y\}\mathrm{d}t_x\mathrm{d}t_y \\ &= E = \begin{cases} \frac{1}{T_X T_Y}\left[\int_0^{T_X}\int_0^{t_y} t_y \mathrm{d}t_x\mathrm{d}t_y + \int_0^{T_Y}\int_{t_y}^{T_X} t_x \mathrm{d}t_x\mathrm{d}t_y\right] & \text{if } T_X \geq T_Y \\ \frac{1}{T_X T_Y}\left[\int_0^{T_Y}\int_0^{t_x} t_x \mathrm{d}t_y\mathrm{d}t_x + \int_0^{T_X}\int_{t_x}^{T_Y} t_y \mathrm{d}t_y\mathrm{d}t_x\right] & \text{if } T_X < T_Y \end{cases} \end{aligned} \tag{10}$$

Solving (10) for each case gives:

$$E = \begin{cases} \frac{T_Y^2}{6T_X} + \frac{T_X}{2} & \text{if } T_X \geq T_Y \\ \frac{T_X^2}{6T_Y} + \frac{T_Y}{2} & \text{if } T_X < T_Y \end{cases} \tag{11}$$

Let us define T and Q as follows:

$$\begin{aligned} T &= \max\{T_X, T_Y\} \\ Q &= \min\left\{\frac{T_X}{T_Y}, \frac{T_Y}{T_X}\right\} \end{aligned} \tag{12}$$

Substituting (12) into (11) one obtains:

$$\begin{aligned} E &= \begin{cases} \frac{Q^2 T}{6} + \frac{T}{2} & \text{if } T_X \geq T_Y \\ \frac{Q^2 T}{6} + \frac{T}{2} & \text{if } T_X < T_Y \end{cases} \\ E &= \frac{Q^2 T}{6} + \frac{T}{2} \end{aligned} \tag{13}$$

Substituting (13) into (7) gives:

$$T_{SC} = 2\left(\frac{Q^2 T}{6} + \frac{T}{2}\right) + 2T_{L/U} \tag{14}$$

The (14) gives the time used for the cycle of work of a robotic arm T_{SC}.

Remark 2 In this paper, the time used in a cycle of work of Cartesian or transelevator robotic arms given by [5] and [29] is used. It can be extended to other kind of robotic arms.

4 The trajectory planning algorithm for a robotic arm

The Fig. 5 shows the position, the velocity, and the acceleration with a stationary period in the central part of the trajectory. The parameter n is the number of periods. The periods are as follows:

1. The transfer periods are at the start and the end of the trajectory with time of $\frac{1}{n}$ of the total time.
2. The stationary period is in the central part of the trajectory with time $\frac{n^2-2}{n}$ of the total time.

The Fig. 6 shows the first transfer period, the velocity $\dot{q}$ is given as follows:

$$\dot{q} = m_1 t + b_1 \tag{15}$$

When $t = 0, \dot{q}(0) = 0$, then $b_1 = 0$. When $t = \frac{t_f}{n}, \dot{q}\left(\frac{t_f}{n}\right) = \dot{q}\max$, then $m_1 = \frac{n}{t_f}\dot{q}\max$.

$\dot{q}\max$ is the maximum velocity, it is obtained considering the distance of the trajectory q_f and the time needed to reach this distance $\frac{(n-1)t_f}{n}$, it gives:

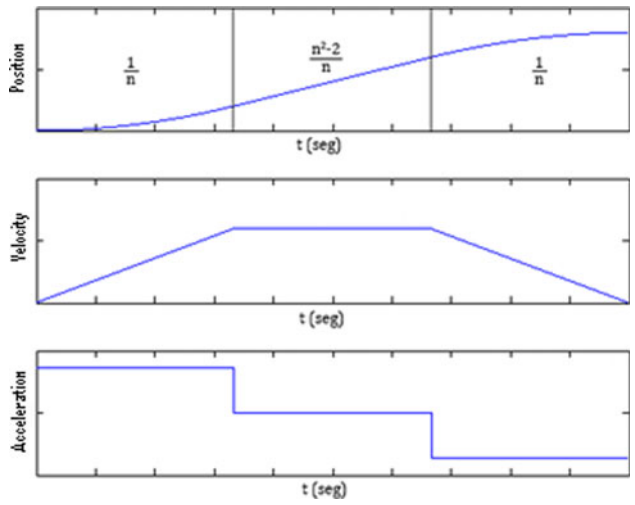

Fig. 5 Position, velocity, and acceleration of the trajectory

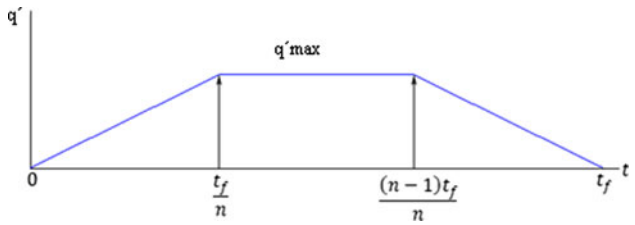

Fig. 6 Behavior of the velocity

$$\dot{q}\max = \left(\frac{n}{n-1}\right)\frac{q_f}{t_f} \tag{16}$$

Substituting b_1, m_1, and $\dot{q}\max$ in (15) gives velocity for the first transfer period as follows:

$$\dot{q} = \left(\frac{n^2}{n-1}\right)\frac{q_f}{t_f^2}t \quad \text{for } t \in \left[0, \frac{t_f}{n}\right] \tag{17}$$

The Fig. 6 shows the stationary period, for this case the velocity is:

$$\dot{q} = \dot{q}\max \tag{18}$$

Substituting $\dot{q}\max$ of (16) into (18) gives the velocity for the stationary period as follows:

$$\dot{q} = \left(\frac{n}{n-1}\right)\frac{q_f}{t_f} \quad \text{for } t \in \left[\frac{t_f}{n}, \frac{(n-1)t_f}{n}\right] \tag{19}$$

The Fig. 6 shows the second transfer period, the velocity $\dot{q}$ is given as follows:

$$\dot{q} = m_2 t + b_2 \tag{20}$$

When $t = \frac{(n-1)t_f}{n}, \dot{q}\left(\frac{(n-1)t_f}{n}\right) = \dot{q}\max$, then $\dot{q}\max = m_2\left(\frac{(n-1)t_f}{n}\right) + b_2$. When $t = t_f, \dot{q}(t_f) = 0$, then $b_2 = -m_2 t_f$.

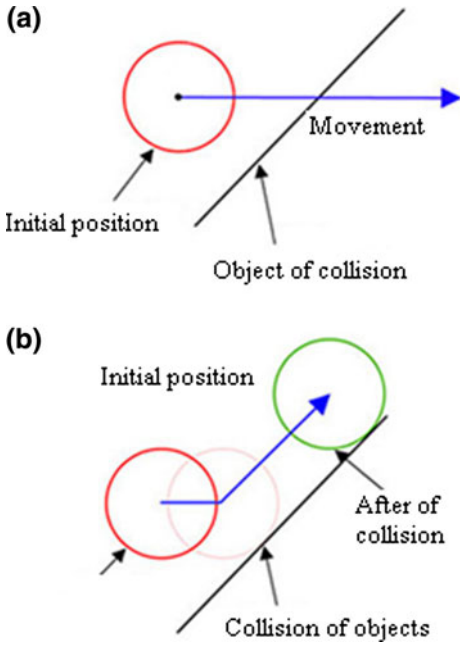

Fig. 7 The collision of a circle with a wall

Substituting b_2 in (20) and making equal to (16) gives m_2 as follows:

$$m_2 = -\left(\frac{n^2}{n-1}\right)\frac{q_f}{t_f^2} \tag{21}$$

Substituting b_2 and m_2 into $\dot{q}$ of (20) gives the velocity for the second transfer period as follows:

$$\dot{q} = \left(\frac{n^2}{n-1}\right)\frac{q_f}{t_f}\left(1 - \frac{t}{t_f}\right) \quad \text{for } t \in \left[\frac{(n-1)t_f}{n}, t_f\right] \tag{22}$$

The (17), (19), and (22) are the equations for the velocity of the trajectory.

For the first transfer period, the position of the trajectory is obtained making the integral of (17) as follows:

$$q = \left(\frac{n^2}{n-1}\right)\frac{q_f}{t_f^2}\frac{t^2}{2} + c_1 \tag{23}$$

When $t = 0$, $q = 0$, then $c_1 = 0$. The position for the first transfer period is:

$$q = \left(\frac{n^2}{n-1}\right)\frac{q_f}{t_f^2}\frac{t^2}{2} \text{ for } t \in \left[0, \frac{t_f}{n}\right] \tag{24}$$

For the stationary period, the position of the trajectory is obtained making the integral of (19) as follows:

$$q = \left(\frac{n}{n-1}\right)\frac{q_f}{t_f}t + c_2 \tag{25}$$

Considering $t = \frac{t_f}{n}$ in (25) gives:

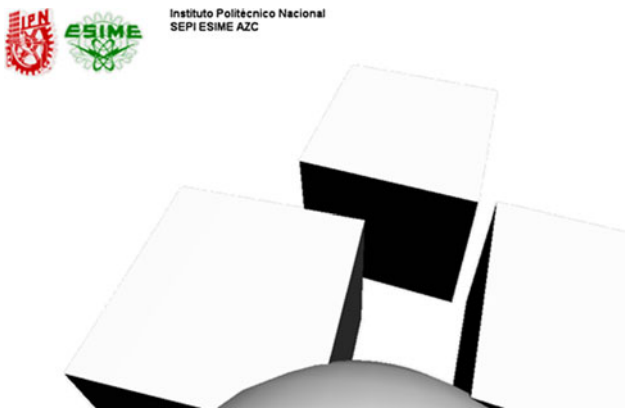

Fig. 8 The collision of a sphere with some cubes

$$q\left(\frac{t_f}{n}\right) = \frac{q_f}{n-1} + c_2 \tag{26}$$

Substituting $t = \frac{t_f}{n}$ in (24) gives:

$$q\left(\frac{t_f}{n}\right) = \frac{1}{2}\frac{q_f}{n-1} \tag{27}$$

Making equal the (25) and (26) gives $c_2 = -\frac{1}{2}\frac{q_f}{n-1}$. The position for the stationary period is:

$$q = \left(\frac{n}{n-1}\right)\frac{q_f}{t_f}t - \frac{1}{2}\frac{q_f}{n-1} \quad \text{for } t \in \left[\frac{t_f}{n}, \frac{(n-1)t_f}{n}\right] \tag{28}$$

For the second transfer period, the position of the trajectory is obtained making the integral of (22) as follows:

$$q = \left(\frac{n^2}{n-1}\right)\frac{q_f}{t_f}\left(t - \frac{t^2}{2t_f}\right) + c_3 \tag{29}$$

Considering $t = \frac{(n-1)t_f}{n}$ in (29) gives:

$$q\left(\frac{(n-1)t_f}{n}\right) = \left(\frac{n^2}{n-1}\right)\frac{q_f}{t_f} \times \left(\frac{(n-1)t_f}{n} - \frac{1(n-1)^2 t_f}{2 \quad n^2}\right) + c_3 \tag{30}$$

Substituting $t = \frac{(n-1)t_f}{n}$ in (28) gives:

$$q\left(\frac{(n-1)t_f}{n}\right) = q_f - \frac{1}{2}\frac{q_f}{n-1} = \frac{q_f}{n-1}\left(n - \frac{3}{2}\right) \tag{31}$$

Making equal the (30) and (31) gives $c_3 = \frac{q_f}{n-1}(n - 1 - \frac{n^2}{2})$. The position for the the second transitory period is:

$$q = \left(\frac{n^2}{n-1}\right)\frac{q_f}{t_f}\left(t - \frac{t^2}{2t_f}\right) + \frac{q_f}{n-1}\left(n - 1 - \frac{n^2}{2}\right)$$

$$q = \frac{q_f}{n-1}\left(\frac{n^2}{t_f}\left(t - \frac{t^2}{2t_f}\right) + n - 1 - \frac{n^2}{2}\right)$$

$$\text{for } t \in \left[\frac{(n-1)t_f}{n}, t_f\right] \tag{32}$$

The (24), (28), and (32) are the equations for the position of the trajectory.

For the first transfer period, the acceleration is the derivative of the velocity (17) as follows:

$$\ddot{q} = \left(\frac{n^2}{n-1}\right)\frac{q_f}{t_f^2} \quad \text{for } t \in \left[0, \frac{t_f}{n}\right] \tag{33}$$

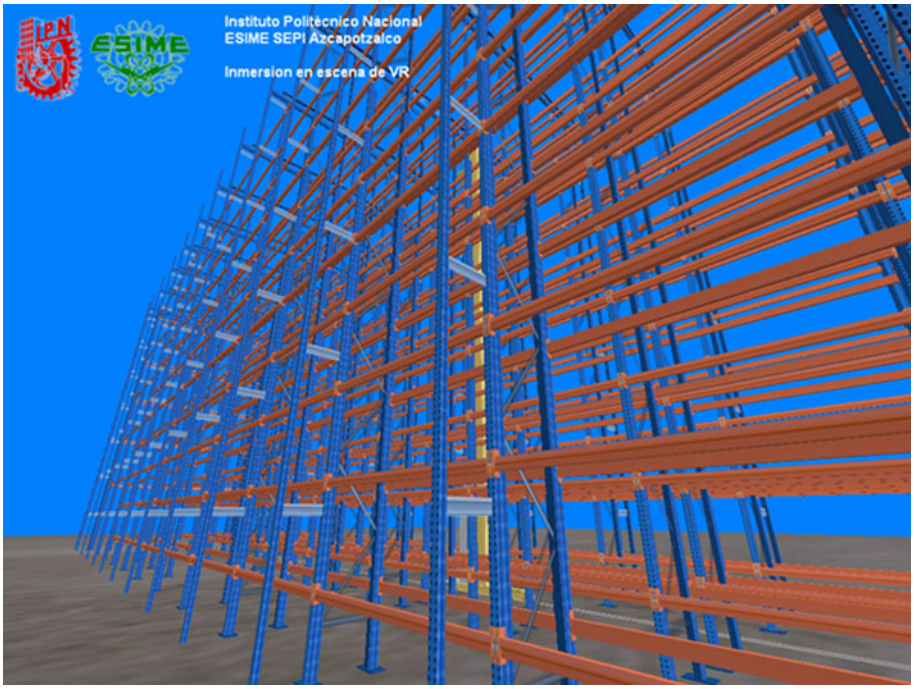

Fig. 9 The GJK distance algorithm applied in a camera inside of a warehouse

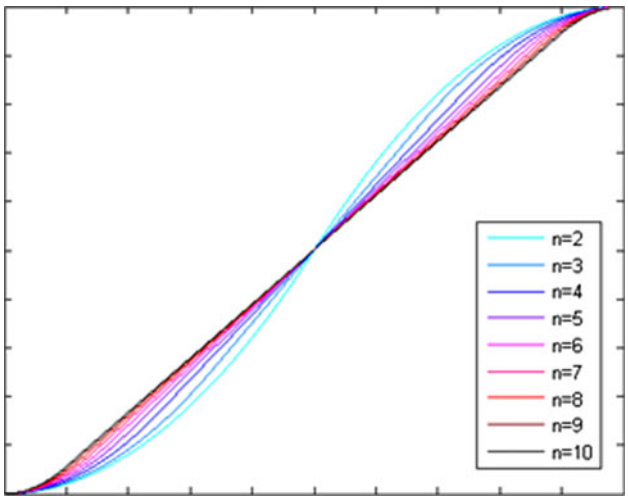

Fig. 10 The position trajectories when n changes from 2 to 10

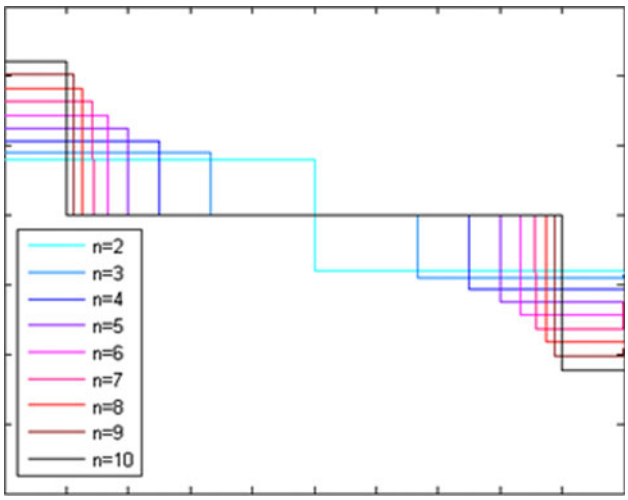

Fig. 12 The acceleration trajectories when n changes from 2 to 10

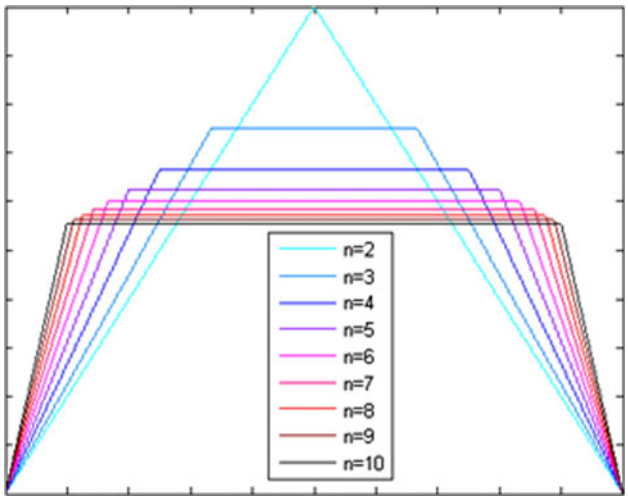

Fig. 11 The velocity trajectories when n changes from 2 to 10

Fig. 13 The trajectory of two cubes

For the stationary period, the acceleration is the derivative of the velocity (19) as follows:

$$\ddot{q} = 0 \quad \text{for } t \in \left[\frac{t_f}{n}, \frac{(n-1)t_f}{n}\right] \tag{34}$$

For the second transfer period, the acceleration is the derivative of the velocity (22) as follows:

$$\ddot{q} = -\left(\frac{n^2}{n-1}\right)\frac{q_f}{t_f^2} \quad \text{for } t \in \left[\frac{(n-1)t_f}{n}, t_f\right] \tag{35}$$

The (33)–(35) are the equations of the acceleration of the trajectory.

Remark 3 The papers [7, 12, 17, 22], and [30] propose algorithms for the trajectory planning, but none is interested into control the time needed to reach the velocity stationary period.

5 Simulations

In this section, the collisions detector, the time used in the cycle of work, and the trajectories planning algorithms are applied in some academic examples and in a warehouse. In many production processes, it is needed a warehouse to save the products [16]. Some robotics arms or cranes are used to move the loads from one place to another one [27].

Example 1 The proposed collisions detector algorithm

The Fig. 7 shows the collision detector algorithm of a circle when touches a wall, it can move free. The (a) shows when the circle has not touched the wall. The (b) shows when the circle touches the wall, the circle does not cross the wall, it goes up.

The Fig. 8 shows the collision of a sphere with some cubes. When the sphere touches any cube, it pushes them. From the Figure, it can be seen that the sphere is pushing two cubes.

The Fig. 9 shows when the GJK distance algorithm is used to detect the collisions of a camera with its

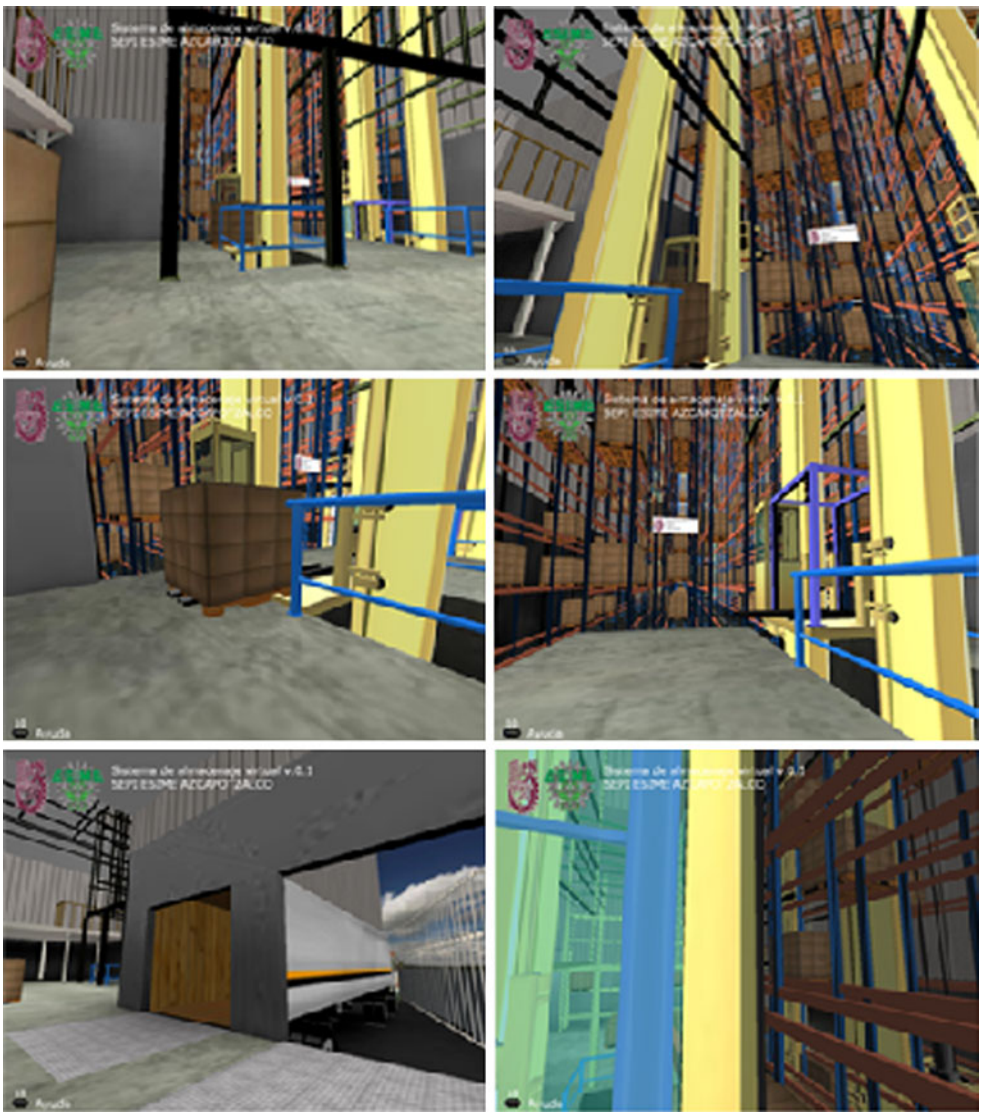

Fig. 14 Some trajectories and cycles of work included in a transelevator

environment inside of a warehouse, the camera is mobile while the structure of the warehouse is fixed. The camera can move in the x and y axis, the camera cannot move in the z axis. The camera cannot cross any part of the structure of the warehouse, the camera needs to go around the structure, when the camera touches the structure, the camera goes to the right or to the left. The proposed collisions detector algorithm is used for the above-mentioned process, the parts of the structure are considered inside of cubes or other simple figures and the camera is considered inside of a cube.

Example 2 The proposed trajectories and cycle of work algorithms

The Fig. 10 shows the position trajectories when n changes from 2 to 10. When $n = 2$, a trajectory without stationary period is obtained, because in $n = 2$, the algorithm is the same as the algorithm of [28], but from $n = 3$ to $n = 10$, the algorithm shows other optional trajectories which are not presented for the algorithm of [28].

The Fig. 11 shows the velocity trajectories when n changes from 2 to 10. When $n = 2$, a trajectory without stationary period is obtained, because in $n = 2$, the algorithm is the same as the algorithm of [28], but from $n = 3$ to $n = 10$, the algorithm shows other optional trajectories which are not presented for the algorithm of [28].

The Fig. 12 shows the acceleration trajectories when n changes from 2 to 10. When $n = 2$, a trajectory without stationary period is obtained, because in $n = 2$, the algorithm is the same as the algorithm of [28], but from $n = 3$ to $n = 10$, the algorithm shows other optional trajectories which are not presented for the algorithm of [28].

The Fig. 13 shows the trajectory of two cubes, the green one is the proposed algorithm with stationary period ($n = 10$) and the red one is the proposed algorithm without stationary period ($n = 2$). From the figure, it can be seen

that the green cube ($n = 10$) goes faster than the red cube ($n = 2$), but both cubes will be at the end at the same time, the difference is that the cubes use different trajectories.

The Fig. 14 shows some trajectories and cycles of work included in a transelevator which moves inside of a warehouse. To show the transelevator in virtual reality, the direct kinematics are the mathematical model of it [13, 28], so the trajectories control the movements of the mathematical model related with each link of the transelevator, and the cycles of work control the time used from the transelevator to go a desired position. The collisions detector algorithm is used in the camera, so the camera cannot cross any structure of the warehouse.

6 Conclusion

In this paper, a modified Gilbert–Johnson–Keerthi algorithm was presented, the proposed GJK algorithm uses a different distance, the time used in a cycle of work of the transelevator robotic arm was presented This paper presented a new trajectory planning algorithm which divides the trajectory in n periods, when n is equal to 2, the proposed algorithm is the same as other algorithms, but for n higher than 2, the proposed algorithm gives other optional trajectories, so the proposed algorithm lets the designer to take a better trajectory than with the previous algorithms. In the future, the cycle of work will be extended to other robotic arms; the collisions detector, the cycle of work, and the trajectories planning algorithms will be applied to other robotic arms, and the proposed algorirthms will be mixed with the evolving systems theory [3, 14, 15, 20, 25, 26].

Acknowledgments The authors are grateful to the editors and the reviewers for their valuable comments and insightful suggestions, which helped to improve this research significantly. The authors thank the Secretaria de Investigacion y Posgrado del IPN, the Comision de Operacion y Fomento de Actividades Academicas del IPN, and the Consejo Nacional de Ciencia y Tecnologia for their help in this research.

References

1. Bachrach J, Beal J, McLurkin J (2010) Composable continuous-space programs for robotic swarms. Neural Comput Appl 19:825–847
2. van den Bergen G (1999) A fast and robust GJK implementation for collision detection of convex objects. J Graph GPU Game Tools 4(2):7–25
3. Bouchachia A (2010) An evolving classification cascade with self-learning. Evol Syst 1(3):143–160
4. Bourke P (1988) Minimum distance between a point and a line. University of Western Australia, Western Australia
5. Chiang SY, Kuo CT, Meerkov SM (2005) c-Bottlenecks in serial production lines: identification and application. Math Prob Eng 7(6)
6. Christensen DJ, Campbell J, Stoy K (2010) Anatomy-based organization of morphology and control in self-reconfigurable modular robots. Neural Comput Appl 19:787–805
7. Cooper GJ (1987) Stability of Runge-Kutta methods for trajectory problems. IMA J Numer Anal 7(1):1–13
8. Correll N, Grob R (2010) From swarm robotics to smart materials. Neural Comput Appl 19:785–786
9. Do Q, Jain L (2010) Application of neural processing paradigm in visual landmark recognition and autonomous robot navigation. Neural Comput Appl 19:237–254
10. Dyllong E, Luther W (2004) The GJK distance algorithm: an interval version for incremental motions. Numer Algorithms 37:127–136
11. Gilbert EG, Johnson DW, Keerty SS (1988) A fast procedure for computing the distance between complex objects in three-dimensional space. IEEE J Robot Autom 4(2)
12. Gokcek C (2004) Stability analisys of periodically switched linear systems using floquet theory. Math Prob Eng 1:1–10
13. Hernandez V, Bravo G, Rubio JJ, Pacheco J (2011) Kinematics for the SCARA and the cylindrical manipulators. ICIC Express Lett Part B Appl (ICIC-ELB) 2(2):421–425
14. Hisada M, Ozawa S, Zhang K, Kasabov N (2010) Incremental linear discriminant analysis for evolving feature spaces in multitask pattern recognition problems. Evol Syst 1(1):17–28
15. Iglesias JA, Angelov P, Ledezma A, Sanchis A (2010) An evolving classification of agent s behaviors: a general approach. Evol Syst 1(3):161–172
16. Jacobs D, Meerkob SM (1995) Mathematical theory of improbability for production systems. Math Prob Eng 1:95–137
17. Kant K (1986) Toward efficient trajectory planning: the path-velocity decomposition. Int J Robot Res 5(3):72–89
18. Kelly R, Santibáñez V, Loría A (2005) Control of robot manipulators in joint space, ISBN-13: 978-1-85233-994-4
19. Khalil HK (2002) Nonlinear systems. 3rd edn. Prentice-Hall, Upper Saddle River
20. Lemos A, Caminhas W, Gomide F (2011) Fuzzy evolving linear regression trees. Evol Syst 2(1):1–15
21. Lewis FL, Dawson DM, Abdallah CT (2004) Control of robot manipulators. Theory Practice, ISBN: 0-8247-4072-6
22. Martins-Filho LS, Macau EEN (2007) Patrol mobile robots and chaotic trajectories. Math Prob Eng
23. Murray RM, Li Z, Sastry SS (1994) A mathematical introduction to robotic manipulation. ISBN: 9780849379819
24. Oubbati M, Palm G (2010) A neural framework for adaptive robot control. Neural Comput Appl 19:103–114
25. Pedrycz W (2010) Evolvable fuzzy systems: some insights and challenges. Evol Syst 1(2):73–82
26. Rubio JJ, Vázquez DM, Pacheco J (2010) Backpropagation to train an evolving radial basis function neural network. Evol Syst 1(3):173–180
27. Silva ARd, Gadelha LC, Schafer B (2007) Joint dynamics modeling and parameter identification for space robot applications. Math Prob Eng
28. Spong MW, Hutchinson S, Vidyasagar M (2006) Robot modeling and control. Wiley, ISBN-13: 978-0-471-64990-8
29. Smith R (2006) Technology disruption in the simulation industry. J Def Model Simul 3(1)
30. Tan HH, Potts RB (1989) A discrete trajectory planner for robotic arms with six degrees of freedom. IEEE Trans Robot Autom 5(5):681–690
31. Wang H, Yu K, Mao B (2009) Self-localization and obstacle avoidance for a mobile robot. Neural Comput Appl 18:495–506
32. Wiklendt L, Chalup S (2009) A small spiking neural network with LQR control applied to the acrobot. Neural Comput Appl 18:369–375
33. Zheng Y-F, Hemami H (2007) Mathematical modeling of a robot collision with its environment. J Robot Syst 2(3):289–307