2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Continuous Distance Computation for Planar Non-holonomic Motions with Constant Accelerations

Enrique J. Bernabeu

Abstract—**A method for computing the distance between two mobile objects following linear or arc-like motions with constant accelerations is introduced in this paper. This distance is obtained without stepping or discretizing any object's motion. Objects are modeled by bi-dimensional convex hulls. The distance-computation algorithm obtains the instant in time when two mobile objects are at their minimum translational distance of separation or penetration. The distance and the instant in time are parallely computed. This method is so fast that can be run as frequent as new information from the world is received.**

I. Introduction

DETECTING a collision in motion planning is still an open research line in Robotics. Nowadays, powerful motion planners are developed, where collision tests are an unavoidable step and represent, in general, a decisive time-consuming part in the planning algorithms.

A recent example and with important social impact is shown by [1], [2]. An estimated motion for an obstacle and a desired one for the robotized car Boss are stepped at a determined time instants. Then, collision tests between the positions of both objects at each considered time instant are run. Objects are modeled by boxes or circles. This collision-detection technique has several limitations as shown by [3]. Nevertheless, this approach is frequent in the literature in order to detect collisions between mobile objects [4].

Other group of collision-detection methods is called Continuous Collision Detection (CCD). In general, these methods also provide, if objects collide, the instant in time of the first contact. Some representative examples are [3], [5−7].

In any case, with all these types of methodologies is really a hard problem to find the exact instant in time when two mobile objects are at their minimum distance.

This paper introduces a technique for obtaining the instant in time when two objects are at their minimum translational distance. If objects do not collide, then the Euclidean distance is computed, otherwise, their minimum translation distance of penetration, defined as [8], is returned.

Objects are modeled by convex hull geometries and follow planar non-holonomic motions with constant accelerations. Specifically, only linear and arc-like motions are considered in this paper.

Considering the previous author's work in [9] as a collision detector, the main contributions of this paper are twofold: arc-like motions are also considered, and obstacles follow motions with non-null acceleration.

The method in this paper is fast enough to be run as frequent as new information from the world is received. And, it is intended to be used as a collision-detection module in sampling-based algorithms for vehicle-like robots.

Manuscript received September 15, 2009.

Enrique J. Bernabeu is with Instituto Universitario de Automática e Informática Industrial, Universidad Politécnica de Valencia, Camino de Vera s/n, Valencia, E-46022, Spain (e-mail: ebernabe@isa.upv.es).

II. Minkowski Difference of Two Motions

In this Section, two mobile objects with null accelerations and linear motions are being considered. Objects are modeled by either a polytope [10] or a spherical-extended polytope (s-tope) [11]. Motions and objects are constrained to be bi-dimensional.

Formally, an s-tope is the convex hull of a finite set of spheres, circles if bi-dimensional, $S=\{s_0,s_1,\ldots,s_{n-1}\}$ with $s_i=(c_i,r_i)$, where c_i is the center and r_i is the radius. S-tope S_S contains an infinite set of swept spheres/circles expressed by

$$S_S=\left\{s=(c,r):\ c=c_0+\sum_{i=1}^{n-1}\lambda_i(c_i-c_0),\ r=r_0+\sum_{i=1}^{n-1}\lambda_i(r_i-r_0),\ s_i=(c_i,r_i)\in S,\lambda_i\in[0,1],\sum_{i=1}^{n-1}\lambda_i\le 1\right\} \quad (1)$$

Note that, if all radii r_i are zero, then (1) is the polytope definition [10]. Consequently, a polytope is a particular case of an s-tope. For this reason, from now, all the objects in this paper are generally modeled by s-topes. The order of the s-tope S_S is the number of spheres/circles in S.

Let $S^A(t_s)$ be the A's position at the instant in time t_s. A is modeled by an n-order s-tope with $S^A(t_s)=\{s_0{}^A(t_s),s_1{}^A(t_s),\ldots,s_{n-1}{}^A(t_s)\}$, where $c_i{}^A(t_s)\in\Re^2$ are the centers and $r_i{}^A\in\Re$ are the radii of circles $s_i{}^A(t_s)=(c_i{}^A(t_s),r_i{}^A)$, i=0,1,…,$n$−1. As A's size does not change, then radii $r_i{}^A$ do not depend on time. A's speed at t_s is stated by the vector $v_A(t_s)\in\Re^2$. $\|v_A(t_s)\|$ indicates the magnitude and $v_A(t_s)$ the direction.

Let $S^B(t_s)$ be the position of a mobile object B at t_s. B is modeled by an m-order s-tope, with $S^B(t_s)=\{s_0{}^B(t_s),s_1{}^B(t_s),\ldots,s_{m-1}{}^B(t_s)\}$. $c_j{}^B(t_s)\in\Re^2$ and $r_j{}^B\in\Re$ are the centers and radii of circles $s_j{}^B(t_s)$, with j=0,1,…,m−1. B's speed at t_s is $v_B(t_s)\in\Re^2$. $\|v_B(t_s)\|$ indicates the magnitude and $v_B(t_s)$ the direction.

Each of the infinite intermediate positions of mobile objects A and B from t_s to a given time horizon Δt, i.e. $S^A(t)$ and $S^B(t)$ for all $t\in[t_s,t_s+\Delta t]$, are parameterized by $\lambda\in[0,1]$ as

$$\begin{aligned} &S^A(t)=\{s_i^A(t)=(c_i^A(t),r_i^A):\ c_i^A(t)=c_i^A(t_s)+\lambda\cdot\Delta t\cdot v_A(t_s);\ i=0,..,n-1\}\\ &S^B(t)=\{s_j^B(t)=(c_j^B(t),r_j^B):\ c_j^B(t)=c_j^B(t_s)+\lambda\cdot\Delta t\cdot v_B(t_s);\ j=0,..,m-1\}\\ &\forall t:\ t=t_s+\lambda\cdot\Delta t;\ t\in[t_s,t_s+\Delta t]\ \text{ and }\ \lambda\in[0,1] \end{aligned} \quad (2)$$

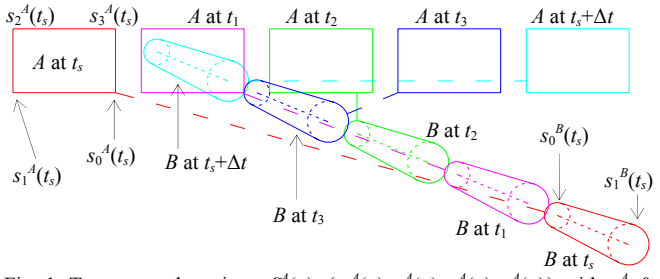

Fig. 1. Two stepped motions. $S^A(t_s)=\{s_0^A(t_s),s_1^A(t_s),s_2^A(t_s),s_3^A(t_s)\}$ with $r_i^A=0$, $\forall$i, and $S^B(t_s)=\{s_0^B(t_s),s_1^B(t_s)\}$ represent A and B positions at t_s. Dashed lines show the distance between A and B at the instants in time $t_s,t_1,t_2,t_3,t_s+\Delta t$.

An example for two objects following a linear motion with constant speed is shown in fig. 1. Distances at each different instant in time are also shown. These distances have been obtained by applying the algorithm in [12], which is an update from the GJK one in [10]. Then, they have been obtained by computing the separation from the origin point O to the Minkowski difference between s-topes A and B at each considered instant in time. Formally, the Minkowski difference between A and B at a given t, is an s-tope $S^{A-B}(t)$, defined by the set of $n\times m$ circles $\{s_{ij}^{A-B}(t)\}$

$$\{s_{ij}^{A-B}(t)\}=\{(c_{ij}^{A-B}(t),r_{ij}^{A-B}): c_{ij}^{A-B}(t)=c_i^A(t)-c_j^B(t); \quad r_{ij}^{A-B}=r_i^A+r_i^B;\ \forall i,j\} \tag{3}$$

Fig. 2 shows all the Minkowski difference s-topes between A and B positions at all the time instants from fig. 1.

The Minkowski difference between A and B positions for all $t\in[t_s,t_s+\Delta t]$ is called $S^M(t)$ and is defined by the set of $n\times m$ circles $\{s_{ij}^M(t)\}$. $s_{ij}^M(t)$ are parameterized by $\lambda\in[0,1]$ as

$$\{s_{ij}^M(t)\}=\{(c_{ij}^M(t),r_{ij}^M): r_{ij}^M=r_i^A+r_i^B; \quad c_{ij}^M(t)=[c_i^A(t_s)+\lambda\cdot\Delta t\cdot v_A(t_s)]-[c_j^B(t_s)+\lambda\cdot\Delta t\cdot v_B(t_s)];\ \forall i,j\} \tag{4}$$
$$\forall t: t=t_s+\lambda\Delta t;\ t\in[t_s,t_s+\Delta t] \text{ and } \lambda\in[0,1]$$

Note that, for instance, if $\lambda=0$, then $S^M(t)$ represents the Minkowski difference between A and B at t_s, i.e., $S^{A-B}(t_s)$. And if $\lambda=1$, then $S^M(t)$ states the Minkowski difference of A and B at $t_s+\Delta t$, i.e, $S^{A-B}(t_s+\Delta t)$.

Each $s_{ij}^M(t)$ for all $t\in[t_s,t_s+\Delta t]$ sweeps an area consisting of a rectangle whose ends are capped off with circles. This geometrical figure is referred to as stadium by [13]. Then, $S^M(t)$ is formed by $n\times m$ stadiums, and each one is defined by three parameters: a start point $c_{ij}^M(t_s)=c_i^A(t_s)-c_j^B(t_s)$, a radius $r_{ij}^M=r_i^A+r_j^B$ and a linear axis $p^M(\lambda)\in\Re^2$. $p^M(\lambda)$ is parametrically defined by $\lambda\in[0,1]$ as

$$p^M(\lambda)=\lambda\cdot\Delta t\cdot(v_A(t_s)-v_B(t_s)) \tag{5}$$

The stadium's axis is the locus swept by $c_{ij}^M(t)$ from t_s to $t_s+\Delta t$. All the axes of the stadiums are equal and their length is $\|p^M(1)\|$. Fig. 3 shows $S^M(t)$ with its $n\times m$ stadiums. Note that $S^M(t)$ contains all the Minkowski differences in fig. 2.

Proposition 1: The distance from O to $S^M(t)$, d_O^M, is the distance at the instant in time when A and B are at their minimum translational distance (MTD) of separation or penetration. Fig. 3 also shows d_O^M computation. Formally, d_O^M is

$$d_O^M=\min_{t\in[t_s,t_s+\Delta t]}\left\{\inf_{\tau\in\Re^2}\{\|\tau\|: dist(S^A(t)+\tau,S^B(t))=0\}\right\} \tag{6}$$

If A and B do not collide during their respective motions, then d_O^M is the Euclidean distance. Nevertheless, if they collide, then the sign of d_O^M is negative and d_O^M acquires the meaning of the MTD of penetration given by [8].

Proof: It is trivial and is a direct consequence of the $S^M(t)$ definition given by (4) and from conclusions by [10]

Let $c_{ab}^M(t_s)=c_a^A(t_s)-c_b^B(t_s)$, r_{ab}^M with axis $p^M(\lambda)$ be the stadium in $S^M(t)$ that is the closest to O, then the distance between O and $S^M(t)$ is from (4)

$$d_O^M=\left\|[c_a^A(t_s)+\lambda\cdot\Delta t\cdot v_A(t_s)]-[c_b^B(t_s)+\lambda\cdot\Delta t\cdot v_B(t_s)]\right\|-r_{ab}^M \tag{7}$$

d_O^M is obtained by finding λ_m, with $\lambda_m\in[0,1]$ that minimizes (7). Given that the axes of the stadiums in $S^M(t)$ are linear, λ_m is obtained by computing O_c, with $O_c=c_{ij}^M(t_s)+p^M(\lambda_m)$, i.e. by projecting O onto such an axis. Therefore,

$$\lambda_m=-\left((c_a^A(t_s)-c_b^A(t_s))\cdot p^M(1)\right)/\|p^M(1)\|^2 \tag{8}$$

Then $d_O^M=\|O_c\|-r_{ab}^M$. As parameter λ is related with time, the time instant t_O^M, when MTD between A and B is d_O^M, is

$$t_O^M=t_s+\lambda_m\cdot\Delta t;\ \text{where } t_O^M\in[t_s,t_s+\Delta t] \text{ and } \lambda_m\in[0,1] \tag{9}$$

Substituting λ_m in (2), $S^A(t_O^M)$ and $S^B(t_O^M)$ are obtained. They respectively represent the positions of mobile objects A and B at time t_O^M, i.e., when their MTD is d_O^M.

Nevertheless, distance computation in (7) fails when O is inside the area delimited by the axes of the stadiums. As $S^M(t)$ is a Minkowski difference, then A and B will collide during their motions. When this situation is presented, d_O^M has to be reformulated as finding $\lambda_m\in[0,1]$ that maximizes

$$d_O^M=-\left(\left\|[c_a^A(t_s)+\lambda\cdot\Delta t\cdot v_A(t_s)]-[c_b^B(t_s)+\lambda\cdot\Delta t\cdot v_B(t_s)]\right\|+r_{ab}^M\right) \tag{10}$$

where $c_a^A(t_s)-c_b^B(t_s)$, r_{ab}^M states the axis of the external stadium in $S^M(t)$ that is the closest to O. In this situation, note that radius r_{ab}^M is added. Sign of d_O^M is negative, because it holds a translational distance of penetration. λ_m and t_O^M are respectively computed as indicated by (8), and (9).

An open problem is finding the closest stadium in $S^M(t)$ to O. This problem is solved by using a GJK-based algorithm, specifically by defining the appropriate support and solution functions. This point will be explained in the next sections.

As a conclusion of this Section, the instant in time, t_O^M, when two mobile objects are at their MTD, while they are following linear paths with constant speeds, is fast obtained without stepping any of the objects motions.

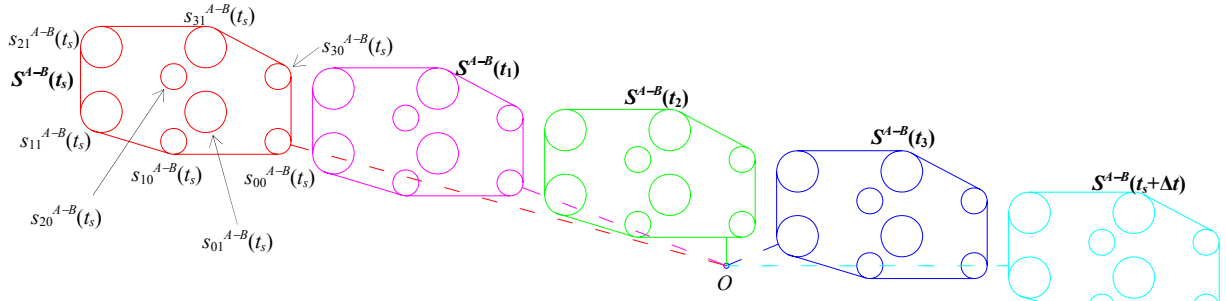

Fig. 2. Minkowski differences s-topes between A and B positions at the instants t_s, t_1, t_2, t_3, $t_s+\Delta t$ (from fig. 1) and their distances to origin point O

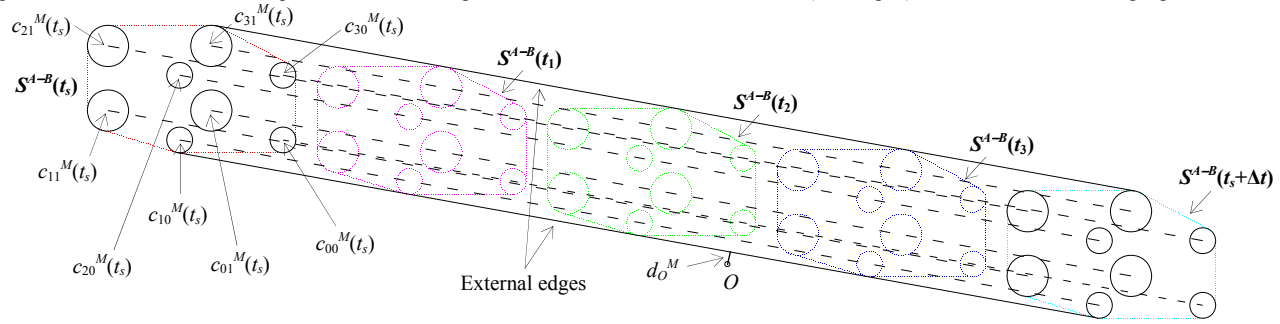

Fig. 3. Stadiums in $S^M(t)$ (black) from motions in fig. 1. For clarity, only the axes (dashed lines) and the extreme circles of the stadiums are depicted. $c_{10}^{M}(t_s)$, $c_{31}^{M}(t_s)$ are the start point of the external stadiums. Their external edges are also shown. Dotted lines show the Minkowski difference s-topes from fig. 2.

III. Distance Between Two Objects Following Linear Motions with Constant Acceleration

A technique for determining, without stepping, the instant in time when two mobile objects are at their MTD, while they are following linear motions with constant accelerations, is introduced in this Section.

Let A be a mobile object, modeled by an n-order s-tope, whose position at t_s is $S^A(t_s)=\{s_0^A(t_s),s_1^A(t_s),\ldots,s_{n-1}^A(t_s)\}$, where $c_i^A(t_s)\in\Re^2$ and $r_i^A\in\Re$ are respectively the centers and radii of circles $s_i^A(t_s)=(c_i^A(t_s),r_i^A)$, $\forall$i. A's speed at t_s is $v_A(t_s)\in\Re^2$. Its constant acceleration is $a_A\in\Re$.

Let B be a mobile object, modeled by an m-order s-tope, whose position at t_s is $S^B(t_s)=\{s_0^B(t_s),s_1^B(t_s),\ldots,s_{m-1}^B(t_s)\}$. $c_j^B(t_s)\in\Re^2$ and $r_j^B\in\Re$ are respectively the centers and radii of circles $s_j^B(t_s)$, $\forall$j. B's speed at t_s is $v_B(t_s)\in\Re^2$ and its constant acceleration is $a_B\in\Re$.

Assuming a time horizon Δt, positions $S^A(t)=\{(c_i^A(t),r_i^A), \forall i\}$, $S^B(t)=\{c_j^B(t),r_j^B), \forall j\}$ for all $t\in[t_s,t_s+\Delta t]$ are parameterized by $\lambda\in[0,1]$, as follows

$$\begin{aligned} c_i^A(t)&=c_i^A(t_s)+\lambda\cdot\Delta t\cdot v_A(t_s)+0.5\cdot\lambda^2\cdot\Delta t^2\cdot a_A\cdot\hat{v}_A(t_s);\ i=0,..,n-1\\ c_j^B(t)&=c_j^B(t_s)+\lambda\cdot\Delta t\cdot v_B(t_s)+0.5\cdot\lambda^2\cdot\Delta t^2\cdot a_B\cdot\hat{v}_B(t_s);\ j=0,..,m-1 \end{aligned} \quad (11)$$

where $\hat{v}_A(t_s)=v_A(t_s)/\|v_A(t_s)\|$ and $\hat{v}_B(t_s)=v_B(t_s)/\|v_B(t_s)\|$. A constraint is introduced in the motions defined by (11). When acceleration is negative, and the time horizon Δt is long enough, the sign of a motion might change, e.g. from moving forward to backwards. Then, if this situation happens, A and B motions in (11) will be conveniently divided. Only motions without changes in their signs are considered.

As mentioned in the previous section, the Minkowski difference between A and B positions for all $t\in[t_s,t_s+\Delta t]$ is $S^M(t)$ and is defined by $n\times m$ stadiums. These stadiums are peculiar because their axes are parabolic. Despite this fact, these geometrical figures are also termed stadiums. Each stadium is defined by a start point $c_i^A(t_s)-c_j^B(t_s)$, a radius $r_i^A+r_j^B$, and a parabolic axis $p^M(\lambda)\in\Re^2$. Axis $p^M(\lambda)$ is common for all the stadiums and is parametrically defined by $\lambda\in[0,1]$ as

$$p^M(\lambda)=\lambda\cdot\Delta t\left(v_A(t_s)-v_B(t_s)\right)+0.5\cdot\lambda^2\cdot\Delta t^2\left(a_A\cdot\hat{v}_A(t_s)-a_B\cdot\hat{v}_B(t_s)\right) \quad (12)$$

The instant in time t_O^M, when mobile object A and B are at their MTD, is $t_O^M=t_s+\lambda_m\cdot\Delta t$, where λ_m is obtained by finding the parameter that minimizes the distance between O and the external stadium in $S^M(t)$ that is the closest to O.

A double problem is now presented: a) computing the distance between O and a stadium with a parabolic axis, b) finding the closest external stadium to O. Both problems are solved by applying a GJK-based algorithm, termed *LL-GJK*.

Set V_k in the *LL-GJK* algorithm always contains one or two stadiums from $S^M(t)$. V_k only stores the start point and radius. The *subdistance_algorithm* computes the distance between O and the stadiums in V_k. Let $c_a^A(t_s)-c_b^B(t_s)$, $r_a^A+r_b^B$ with $(c_a^A(t_s),r_a^A)\in S^A(t_s)$ and $(c_b^B(t_s),r_b^B)\in S^B(t_s)$ be a stadium in V_k. The distance between O and such a stadium is determined by finding the solution λ_c that verifies

$$d\left\|c_a^A(t_s)-c_b^B(t_s)+p^M(\lambda)\right\|/d\lambda=0 \quad (13)$$

$\|c_a^A(t_s)-c_b^B(t_s)+p^M(\lambda)\|$ only contains one minimum for all $\lambda\in[0,1]$. λ_c is found by applying the root-finding technique, termed Secant method [14], to (13). Experimentally, $\lambda_0=0.45$ and $\lambda_1=0.55$ have been confirmed as good choices. Accuracy for the Secant method has been set to 10^{-6}.

LL-GJK algorithm

Input: $S^A(t_s)$, $S^B(t_s)$, t_s, Δt, $p^M(\lambda)$
Output: (λ_m, $t_O{}^M$, $d_O{}^M$) or (*failure*, V_k)

1: k=0, V_k=$\{c_0{}^A(t_s)-c_0{}^B(t_s), r_0{}^A+r_0{}^B\}$ with $(c_0{}^A(t_s), r_0{}^A)\in S^A(t_s)$ and $(c_0{}^B(t_s), r_0{}^B)\in S^B(t_s)$
2: **do**
3: $(\lambda_c, d_O, O_c, V'_k, O\_in) \leftarrow$ *subdistance_algorithm*(V_k)
4: **if** *O_in* **then return**(*failure*,V_k)
5: *compute* $h_M(-O_c,\lambda_c)$, $s_M(-O_c)$, $h'_M(-O_c,\lambda_c)$, $s'_M(-O_c,\lambda_c)$
6: **if** $g_M(-O_c,\lambda_c)$=0 **then** *exit_loop* **endif**
7: **if** $s'_M(-O_c,\lambda_c)$=$s_M(-O_c)$ **or** $h_M(-O_c)$>$h'_M(-O_c,\lambda_c)$ **then** V_{k+1}=$V'_k\cup\{s_M(-O_c)\}$ **else** V_{k+1}=$V'_k\cup\{s'_M(-O_c,\lambda_c)\}$ **endif**
8: k=k+1
9: **while** *true*
10: λ_m=λ_c; $t_O{}^M$=t_s+$\lambda_m\cdot\Delta t$; $d_O{}^M$=d_O−$(r_p{}^A+r_q{}^B)$ where V'_k=$\{c_p{}^A(t_s)-c_q{}^B(t_s), r_p{}^A+r_q{}^B\}$; with $(c_p{}^A(t_s), r_p{}^A)\in S^A(t_s)$, $(c_q{}^B(t_s), r_q{}^B)\in S^B(t_s)$
11: **return**(λ_m, $t_O{}^M$, $d_O{}^M$)

After finding λ_c, O_c and d_O are obtained as

$$O_c=c_a^A(t_s)-c_b^B(t_s)+p^M(\lambda_c);\quad O_c\in\Re^2;\quad d_O=\|O_c\|; \qquad (14)$$

If V_k contains one stadium, then λ_c, d_O, $O_c\in\Re^2$, V'_k=V_k and *O_in=false* are returned by the *subdistance_algorithm*.

If V_k contains two stadiums, first step consists of checking if O is inside the area delimited by the axes of the stadiums in V_k. If so, *O_in=true* is returned by the *subdistance_algorithm*, and then *LL-GJK* algorithm finishes returning failure (see step 4). On the contrary, if O is not inside, the distance from O to each stadium in V_k is computed. Parameters λ_c, d_O, O_c from the closest stadium to O are returned. V'_k only contains the closest stadium. The furthest stadium is rejected and is not considered anymore in the *LL-GJK* algorithm.

In order to find the external stadiums in $S^M(t)$, two pairs of support and solution functions are introduced. The first pair of support $h_M(\eta)$ and solution $s_M(\eta)$ functions with $\eta\in\Re^2$ is

$$h_M(\eta)=\max_{\forall i,j}\left\{(c_i^A(t_s)-c_j^B(t_s))\cdot\eta+(r_i^A+r_j^B)\cdot\|\eta\|\right\} \quad \text{with } (c_i^A(t_s), r_i^A)\in S^A(t_s);\ (c_j^B(t_s), r_j^B)\in S^B(t_s) \qquad (15)$$

$s_M(\eta)$ is the circle $s_M(\eta)$=$(c_a{}^A(t_s)-c_b{}^B(t_s), r_a{}^A+r_b{}^B)$ that gives value to $h_M(\eta)$, i.e. it represents the stadium in $S^M(t)$ whose start point is the furthest from O in the direction η [12], [10].

Given that axes are parabolic, a second pair of support $h'_M(\eta,\lambda_c)$ and mapping $s'_M(\eta,\lambda_c)$ functions is defined. These functions find the furthest stadium from O in the direction η at the points where the axes of the stadiums are close to O.

$$h'_M(\eta,\lambda_c)=\max_{\forall i,j}\left\{\left(c_i^A(t_s)-c_j^B(t_s)+p^M(\lambda_c)\right)\cdot\eta+(r_i^A+r_j^B)\cdot\|\eta\|\right\} \quad \text{with } (c_i^A(t_s), r_i^A)\in S^A(t_s);\ (c_j^B(t_s), r_j^B)\in S^B(t_s) \qquad (16)$$

LL_{in}-GJK algorithm

Input: $S^A(t_s)$, $S^B(t_s)$, t_s, Δt, $p^M(\lambda)$, a one-element set V_{in}
Output: (λ_m, $d_O{}^{M'}$)

1: k=0, V_k=V_{in}
2: **do**
3: $(\lambda_c, d_O, O_c, V'_k, V_k) \leftarrow$ *subdistance_in_algorithm*(V_k)
4: *compute* $h_M(O_c)$, $s_M(O_c)$, $h'_M(O_c,\lambda_c)$, $s'_M(O_c,\lambda_c)$
5: **if** $\hat{g}_M(O_c,\lambda_c)$=0 **then** *exit_loop* **endif**
6: **if** $s'_M(O_c,\lambda_c)$=$s_M(O_c)$ **or** $h_M(O_c)$>$h'_M(O_c,\lambda_c)$ **then** V_{k+1}=$V'_k\cup\{s_M(O_c)\}$ **else** V_{k+1}=$V'_k\cup\{s'_M(O_c,\lambda_c)\}$ **endif**
7: k=k+1
8: **while** *true*
9: λ_m=λ_c; $d_O{}^{M'}$=−$(d_O+(r_p{}^A+r_q{}^B))$ where V'_k=$\{c_p{}^A(t_s)-c_q{}^B(t_s), r_p{}^A+r_q{}^B\}$; with $(c_p{}^A(t_s), r_p{}^A)\in S^A(t_s)$, $(c_q{}^B(t_s), r_q{}^B)\in S^B(t_s)$
10: **return**(λ_m, $d_O{}^{M'}$)

where λ_c comes from the last execution of the *subdistance_algoritm*. The circle $s'_M(\eta,\lambda_c)$ represents the stadium in $S^M(t)$ that gives value to $h'_M(\eta,\lambda_c)$. If $s_M(\eta)$ and $s'_M(\eta,\lambda_c)$ represent different stadiums, the one with the greater support function is selected. See step 7 in the *LL-GJK* algorithm.

LL-GJK algorithm finishes when $g_M(-O_c,\lambda_c)$=0 is verified with $g_M(-O_c,\lambda_c)$=$\|O_c\|^2-radius(s'_M(-O_c,\lambda_c))\|O_c\|+h'_M(-O_c,\lambda_c)$ [12]. In other words, no other stadium is closer to O that the one in V_k, and finally, distance between O and $S^M(t)$ is obtained. The distance $d_O{}^M$ and time $t_O{}^M$ are then returned. Function *radius* returns the radius of circle $s'_M(-O_c,\lambda_c)$.

If *LL-GJK* algorithm finishes with a failure, then O is inside the area delimited by the axes of the stadiums and the returned set V_k contains two stadiums. In this case, distances from the inner O to the two external stadiums in $S^M(t)$ have to be computed. The *LL_{in}-GJK* algorithm computes the distance from O to one external stadium. For this reason, *LL_{in}-GJK* is called twice. Each call receives as input, set V_{in}, one stadium from the returned V_k.

The procedure *subdistance_in_algorithm* is only different from the *subdistance_algorithm* in the *LL-GJK* algorithm when V_k contains two stadiums. In this case, the furthest stadium from O is selected and assigned to V'_k, while the other one is rejected and is not used anymore in the current execution of the *LL_{in}-GJK* algorithm.

The external stadium is found in the *LL_{in}-GJK* algorithm by searching the furthest stadium in the direction O_c. For this reason, $h'_M(O_c,\lambda_c)$, $s'_M(O_c,\lambda_c)$, $h_M(O_c)$, $s_M(O_c)$ are now computed. This makes that the final condition also changes. For this reason, the *LL_{in}-GJK* algorithm finishes when $\hat{g}_M(O_c,\lambda_c)$=−$\|O_c\|^2-radius(s'_M(O_c,\lambda_c))\|O_c\|+h'_M(O_c,\lambda_c)$ is 0.

Each execution of the *LL_{in}-GJK* algorithm returns a parameter λ_m and a negative distance $d_O{}^{M'}$. The maximum of the two returned distances holds $d_O{}^M$, i.e. the MTD of penetration between both mobile objects. And, from its associated parameter λ_m, $t_O{}^M$ is then obtained by applying (9).

Two objects, A and B, following linear motions with constant accelerations and the corresponding Minkowski difference $S^M(t)$ are shown in fig. 4. The positions, where A and B are at their MTD, are depicted in red. A is modeled by a 4-order s-tope (potytope) and B is a 2-order s-tope. $d_O{}^M$ and $t_O{}^M$ have been obtained in 6.4 μs in an Intel® Core™ 2 Duo E8200 processor at 2.66 GHz.

IV. Distance between Objects following Linear and Arc-like Motions with Constant Accelerations

The computation of the distance between two mobile objects following respectively linear and arc-like motions with constant accelerations is shown in this Section.

Let A be an object with an arc-like motion and modeled by an n-order s-tope. A's position at t_s is $S^A(t_s)=\{s_0{}^A(t_s), s_1{}^A(t_s),\ldots,s_{n-1}{}^A(t_s)\}$, where $c_i{}^A(t_s)\in\Re^2$ and $r_i{}^A\in\Re$ are the centers and the radii of circles $s_i{}^A(t_s)=(c_i{}^A(t_s),r_i{}^A)$, $\forall$i. A's arc-like motion is centered at c_A. With respect to c_A, each center $c_i{}^A(t_s)$ is expressed by a radius $\rho_i{}^A$ and angle $\theta_i{}^A(t_s)$ such that $c_i{}^A(t_s)=c_A+\rho_i{}^A(\cos(\theta_i{}^A(t_s),\sin(\theta_i{}^A(t_s))$. A's angular speed at t_s is $\omega_A(t_s)\in\Re$. A's constant angular acceleration is $\alpha_A\in\Re$.

Assuming a time horizon Δt, positions $S^A(t)=\{(c_i{}^A(t),r_i{}^A), \forall i\}$ for all $t\in[t_s,t_s+\Delta t]$ are parameterized by $\lambda\in[0,1]$ as

$$c_i{}^A(t)=c_A+\rho_i{}^A\left(\cos(\theta_i{}^A(t)),\sin(\theta_i{}^A(t))\right);\ i=0,..,n-1$$
$$\text{where}\quad \theta_i{}^A(t)=\theta_i{}^A(t_s)+\lambda\cdot\Delta t\cdot w_A(t_s)+0.5\cdot\lambda^2\cdot\Delta t^2\cdot\alpha_A \qquad (17)$$
$$\forall t: t=t_s+\lambda\cdot\Delta t;\ t\in[t_s,t_s+\Delta t]\ \text{ and }\ \lambda\in[0,1]$$

Let B be an object with a linear motion and modeled by an m-order s-tope. B's position at t_s is $S^B(t_s)=\{s_0{}^B(t_s),s_1{}^B(t_s), \ldots,s_{m-1}{}^B(t_s)\}$. $c_j{}^B(t_s)\in\Re^2$ and $r_j{}^B\in\Re$ are respectively the centers and radii of circles $s_j{}^B(t_s)$, $\forall$j. B's speed at t_s is $v_B(t_s)\in\Re^2$ and its constant acceleration is $a_B\in\Re$. B's positions $S^B(t)$ for all $t\in[t_s,t_s+\Delta t]$ are indicated in (11).

Depending on the acceleration and the time horizon, the sign of a motion might change, for instance, from forward to backwards. If this situation happens, A and B motions will be conveniently divided. Only motions without changes in their signs are considered.

Each of the $n\times m$ stadiums in $S^M(t)$ is defined by a start point $c_i{}^A(t_s)-c_j{}^B(t_s)$, a radius $r_i{}^A+r_j{}^B$ and an axis. From the definition of the A and B motions in (17) and (11) is concluded that there are n different axes and they are cycloid-like. Each of the n axes $p_i{}^M(\lambda)\in\Re^2$ $\forall$i is described by $\lambda\in[0,1]$ as

$$p_i{}^M(\lambda)=\rho_i{}^A\left(\cos(\theta_i{}^A(t)),\sin(\theta_i{}^A(t))\right)-\lambda\cdot\Delta t\cdot v_B(t_s)- \qquad (18)$$
$$-0.5\cdot\lambda^2\cdot\Delta t^2\cdot a_B\cdot\hat{v}_B(t_s);\ i=0,..,n-1$$

The angle $\theta_i{}^A(t)$ has been defined in (17).

The instant in time when A and B are at their MTD is obtained by applying the algorithms *AL-GJK* and *AL$_{in}$-GJK*.

The *AL-GJK* and *AL$_{in}$-GJK* algorithms are respectively analogous to the *LL-GJK* and *LL$_{in}$-GJK* ones. Only the sub

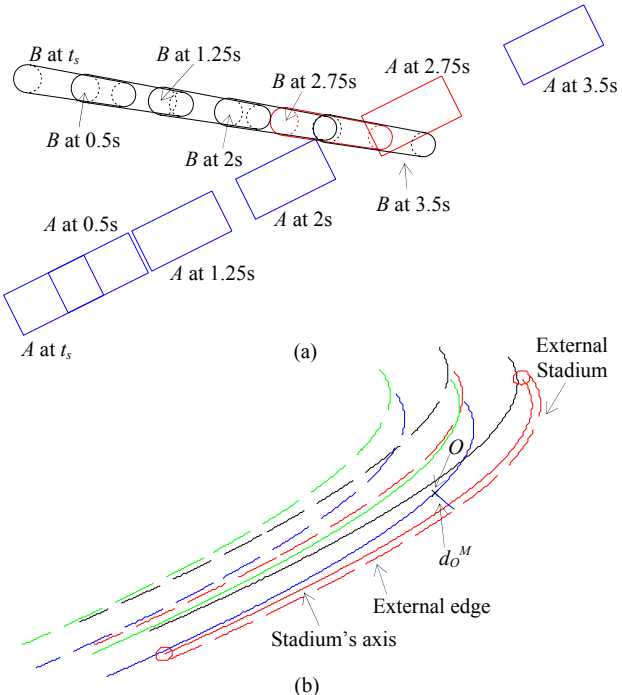

Fig. 4. Distance between two mobile objects following linear motions, with $\|v_A(t_s)\|$=2.2 m/s, a_A=1 m/s^2, $\|v_B(t_s)\|$=3 m/s, a_B=−0.5 m/s^2, t_s=0s, and Δt=5s. (a) A and B positions are only depicted at t_s, 0.5s, 1.25s, 2s, 2.75s and 3.5s. The positions where A and B are at their MTD are in red. The MTD of penetration $d_O{}^M$ is given at $t_O{}^M$=2.75s. (b) Axes of the eight stadiums in $S^M(t)$ and the distance $d_O{}^M$. For clarity, only extreme circles, axis, and edge of the closest external stadium to O are depicted.

tle differences between them are pointed out in this Section.

The *subdistance_algorithm* and *subdistance_in_algorithm* now compute the distance between O and a cycloid-like axis. Let $c_a{}^A(t_s)-c_b{}^B(t_s)$, $r_a{}^A+r_b{}^B$ with $(c_a{}^A(t_s),r_a{}^A)\in S^A(t_s)$ and $(c_b{}^B(t_s),r_b{}^B)\in S^B(t_s)$ be a stadium with axis $p_a{}^M(\lambda)$. The distance between O and such a stadium is determined by finding the λ_c that minimizes $\|c_A-c_b{}^B(t_s)+p_a{}^M(\lambda)\|$, i.e., by solving

$$d\,\|c_A-c_b^B(t_s)+p_a^M(\lambda)\|\,/\,d\lambda=0 \qquad (19)$$

λ_c is then found by applying the Secant method to (19), but this method works properly if there is one minimum in $\|c_A-c_b{}^B(t_s)+p_a{}^M(\lambda)\|$. Given that the axes of the stadiums are cycloid-like, if A's angular displacement is lower than π, then $\|c_A-c_b{}^B(t_s)+p_a{}^M(\lambda)\|$ with $\lambda\in[0,1]$, contains, in the worst case, one maximum and one minimum (apart from the extremes of the search interval). Consequently, if such a condition is false, then A and B motions are properly divided before running the distance-computation algorithms.

Note that while Secant method iterates, it is trivial to detect if the root being searched is a maximum or a minimum.

As a consequence of dealing with cycloid-like axes, support function $h'_M(\eta)$ has to be updated in the *AL-GJK* and *AL$_{in}$-GJK* algorithms as follows

$$h'_M(\eta,\lambda_c)=\max_{\forall i,j}\left\{\left(c_A-c_j^B(t_s)+p_i^M(\lambda_c)\right)\cdot\eta+(r_i^A+r_j^B)\cdot\|\eta\|\right\} \qquad (20)$$

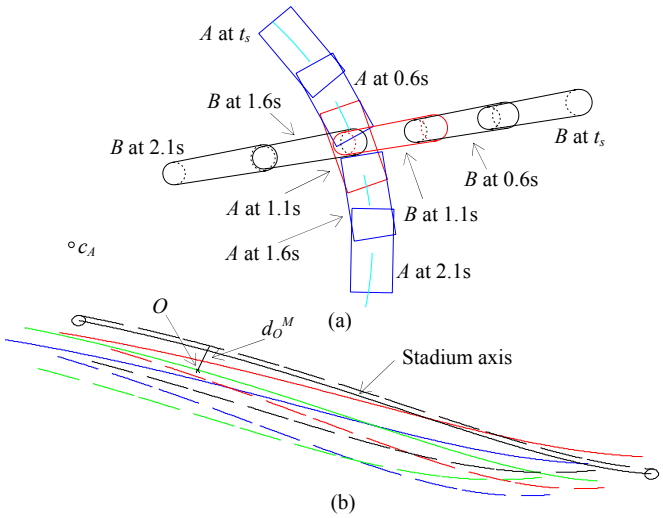

Fig. 5. Distance between two mobile objects. A's motion is arc-like. B's motion is linear. The motions are defined by $\omega_A(t_s)$=−18º/s, α_A=−0.5º/s^2, $\|v_B(t_s)\|$=3 m/s, a_B=1 m/s^2, t_s=0 s and Δt=5 s. (a) A and B positions are only depicted at t_s, 0.6s, 1.1s, 2s, 1.6s and 2.1s. The positions where A and B are at their MTD are in red. The MTD of penetration d_O^M is given at t_O^M=1.1s. (b) Cycloid-like axes of the corresponding eight stadiums in $S^M(t)$ and the distance d_O^M. For clarity, only extremes circles, axis, and edge of the closest external stadium to O are depicted.

Two objects, A and B, following respectively arc-like and linear motions with constant accelerations, s-tope $S^M(t)$, and the positions where A and B are at their MTD are shown in fig. 5. A is modeled by a 4-order s-tope (potytope), while B is a 2-order s-tope. d_O^M and t_O^M have been obtained in 17.6 μs in an Intel® Core™ 2 Duo E8200 processor at 2.66 GHz.

V. Distance between Two Objects Following Arc-like Motions with Constant Angular Acceleration

The computation of the distance between two objects following arc-like motions with constant angular accelerations is tackled in this Section.

Let A be an object modeled by an n-order s-tope whose arc-like motions is the same that the given in the previous Section.

And, let B be an object following an arc-like motion and modeled by an m-order s-tope. B's position at t_s is given by $S^B(t_s)=\{s_0^B(t_s),s_1^B(t_s),\ldots,s_{m-1}^B(t_s)\}$. $c_j^B(t_s)\in\Re^2$ and $r_j^B\in\Re$ are respectively the centers and radii of circles $s_j^B(t_s)$, $\forall$j. B's arc-like motion is centered at c_B. With respect to c_B, each center $c_i^B(t_s)$ is expressed by a radius ρ_j^B and angle $\theta_j^B(t_s)$. B's angular speed at t_s is $\omega_B(t_s)\in\Re$. B's constant angular acceleration is $\alpha_B\in\Re$.

Assuming a time horizon Δt, positions $S^B(t)=\{(c_j^B(t),r_j^B), \forall j\}$ for all $t\in[t_s,t_s+\Delta t]$ are parameterized by $\lambda\in[0,1]$

$$c_j^B(t)=c_B+\rho_j^B\left(\cos(\theta_j^B(t)),\sin(\theta_j^B(t))\right);\ j=0,..,n-1$$
$$\text{where}\quad \theta_j^B(t)=\theta_j^B(t_s)+\lambda\cdot\Delta t\cdot w_B(t_s)+0.5\cdot\lambda^2\cdot\Delta t^2\cdot\alpha_B \qquad (21)$$
$$\forall t: t=t_s+\lambda\cdot\Delta t;\ t\in[t_s,t_s+\Delta t]\ \text{ and }\ \lambda\in[0,1]$$

If signs of the angular speed and acceleration are different, and depending on Δt, an arc-like motion might change, for instance, from going clockwise to counterclockwise. If this situation happens, then A and B motions have to be divided. Only motions without these changes are considered.

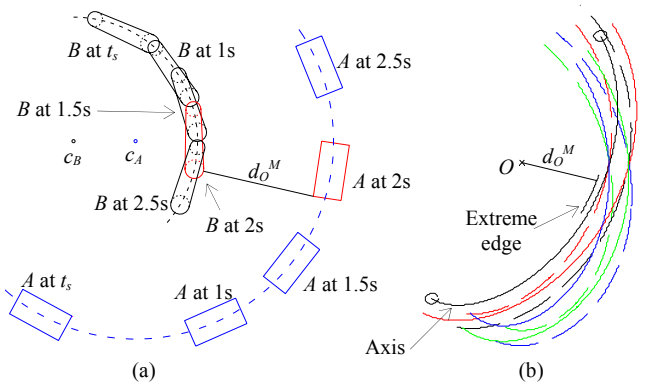

Fig. 6. Distance between two objects with arc-like motions. The motions are described by $\omega_A(t_s)$=50º/s,α_A=5º/s^2, $\omega_B(t_s)$=−30º/s, a_B=−2.5º/s^2, t_s=0s, and Δt=3s. (a) A and B positions are only depicted at t_s, 1s, 1.5s, 2s, and 2.5s. The positions, where A and B are at their MTD, are in red. The MTD, d_O^M, is given at t_O^M=2s. (b) Rose-like axes of the eight stadiums in $S^M(t)$ and the distance d_O^M (at a different scale). For clarity, only extremes circles, axis, and partially the edge of the closest external stadium to O are depicted.

The axes of the stadiums in $S^M(t)$ are now rose-like (rhodonea curve) [15]. $S^M(t)$ has $n\times m$ different axes $p_{ij}^M(\lambda)\in\Re^2$ $\forall$i,j. These axes are parameterized by $\lambda\in[0,1]$ as

$$p_{ij}^M(\lambda)=\rho_i^A(\cos(\theta_i^A(t)),\sin(\theta_i^A(t)))-\rho_j^B(\cos(\theta_j^B(t)),\sin(\theta_j^B(t))) \qquad (22)$$

where $t=t_s+\lambda\cdot\Delta t$. The angles $\theta_i^A(t)$ and $\theta_j^B(t)$ are respectively given by (17) and (21). Each stadium is defined by a start point $c_i^A(t_s)-c_j^B(t_s)$, a radius $r_i^A+r_j^B$, and an axis $p_{ij}^M(\lambda)$.

The instant in time when A and B are at their MTD is obtained by applying the *AA-GJK* and *AA$_{in}$-GJK* algorithms. These algorithms are analogous to *LL-GJK* and *LL$_{in}$-GJK*.

The *subdistance_algorithm* and *subdistance_in_algorithm* now compute the distance between O and a stadium whose axis is rose-like. Let $c_a^A(t_s)-c_b^B(t_s)$ with $(c_a^A(t_s),r_a^A)\in S^A(t_s)$ and $(c_b^B(t_s),r_b^B)\in S^B(t_s)$, radius $r_a^A+r_b^B$, and axis $p_{ab}^M(\lambda)$ be a stadium. This distance is obtained by finding λ_c that minimizes $\|c_A-c_B+p_{ab}^M(\lambda)\|$, i.e. by applying the Secant method to

$$d\,\|c_A-c_B+p_{ab}^M(\lambda)\|\,/\,d\lambda=0 \qquad (23)$$

As axes of the stadiums are rose-like, if A and B angular displacements are lower than π, then $\|c_A-c_b^B(t_s)+p_a^M(\lambda)\|$ with $\lambda\in[0,1]$ contains, in the worst case, one maximum and minimum (apart from the extremes of the interval of search). If this condition is not true, then A and B motions have to be divided before running the distance-computation algorithms.

Given that rose-like axes are being dealt, support function $h'_M(\eta,\lambda_c)$ in the *AA-GJK* and *AA$_{in}$-GJK* algorithms is updated

$$h'_M(\eta,\lambda_c)=\max_{\forall i,j}\left\{\left(c_A-c_B+p_{ij}^M(\lambda_c)\right)\cdot\eta+(r_i^A+r_j^B)\cdot\|\eta\|\right\} \qquad (24)$$

Two objects, A and B, following arc-like motions with

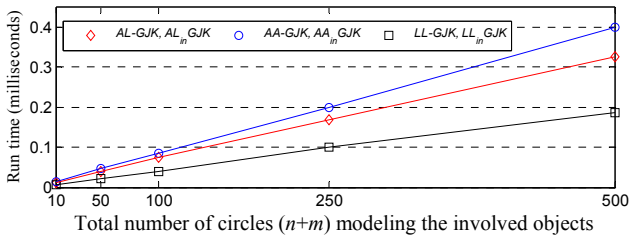

Fig. 7. Computational cost of the algorithms.

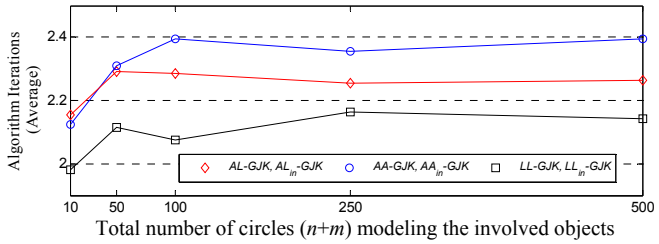

Fig. 8. Average number of iterations in the algorithms per distance

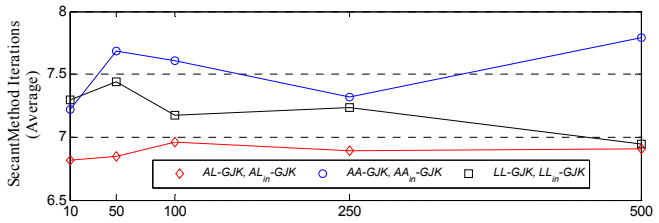

Fig. 9. Average number of iterations in the Secant method per distance.

constant accelerations, s-tope $S^M(t)$, and the positions where A and B are at their MTD are shown in fig. 6. A is modeled by a 4-order s-tope (polytope), while B is a 2-order s-tope. d_O^M and t_O^M have been obtained in 16.5 μs in an Intel® Core™ 2 Duo E8200 processor at 2.66 GHz.

VI. Algorithm Analysis

All the support functions in this paper verify

$$h_M(\eta)=h_{S^A(t_s)}(\eta)+h_{S^B(t_s)}(-\eta)$$
$$h'_M(\eta,\lambda_c)=h'_{S^A(t)}(\eta,\lambda_c)+h'_{S^B(t)}(-\eta,\lambda_c) \quad (25)$$

where $S^A(t)$, $S^B(t)$ represent A and B positions at $t=t_s+\lambda_c\cdot\Delta t$. Proof of (25) is trivial. As a consequence of conditions in (25), s-tope $S^M(t)$ does not need to be compute before running any of the *LL-GJK, LL$_{in}$-GJK*, *AL-GJK, AL$_{in}$-GJK*, *AA-GJK*, and *AA$_{in}$-GJK* algorithms. And then, complexity of all these algorithm is O(n+m) instead of O(n×m).

These algorithms are implemented in C and run in an Intel Core 2 Duo E8200 processor at 2.66 GHz. The objects and motions have been generated randomly. More than 1,500 different experiments have been run.

The runtime of the algorithms per each computed distance is shown in fig. 7. The complexity of the algorithms is linear with respect to the total number of circles modeling both objects.

The total number of iterations in all the algorithms is stable. Fig. 8 shows the average number of iterations per each computed distance. The iterations in the Secant method are also stable. See fig. 9.

VII. Conclusion

This paper has shown a method for detecting a collision between two mobile objects without stepping their motions. Specifically, this method obtains the instant in time two objects in motion are at their minimum translational distance of separation or penetration. The mentioned distance and time instant are parallely computed. The distance's sign encodes if objects collide or not.

Objects are modeled by bi-dimensional convex hulls. Their motions are non-holonomic (linear or arc-like) with constant accelerations. The positions of the objects are assumed to be measurable and their motions are estimable.

Some experiments have been run to conclude that the method is stable in the number of iterations. And, it is fast enough to be run as frequent as new information from the sensors system is receive.

This method is also able to compute the instant in time of the first contact by forcing the distance to be zero. A future extension of this work consists of updating it to deal with non-convex objects.

References

[1] D. Ferguson, T. M. Howard and M. Likhachev, "Motion planning in urban environment: Part I," in Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2008, pp. 1063-1069.

[2] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, et al., "Autonomous driving in urban environments: Boss and the urban challenge". *Journal of Field Rob.*, vol. 25, nº 8, pp. 425-466, 2008.

[3] F. Schwarzer, M. Saha, J-C. Latombe, "Adaptive dynamic collision checking for single and multiple articulated robots in complex environments," *IEEE Trans. on Robotics*, vol. 21, nº 3, pp. 338-353, 2005.

[4] P. Jimenez, F. Thomas, C. Torras, "3D collision detection: a survey," *Comput. Graph.*, vol. 25, pp 269-285, 2001.

[5] S. Redon, A. Kheddar, and S. Coquillart, "Fast continuous collision detection between rigid bodies," *Computer Graphic Forum*, vol. 21, no. 3, pp. 279-288, 2002.

[6] J. Canny, "Collision detection for moving polyhedra," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 8, nº 2, pp. 200–209, 1986.

[7] Y-K. Choi, W. Wang, Y. Liu and M-S. Kim, "Continuous collision detection for two moving elliptic disks," *IEEE Trans. Robotics*, vol. 22, nº 2, pp. 213-224, 2006.

[8] S. Cameron and R. K. Culley, "Determining the minimum translational distance between two convex polyhedral," in Proc. IEEE Int. Conf. on Robotics and Automation, 1986, pp. 591-596.

[9] E. J. Bernabeu, "Fast generation of multiple collision-free and linear trajectories in dynamic environments," *IEEE Trans. Robotics*, vol. 25, nº 4, pp. 967-975, 2009.

[10] E. G. Gilbert, D. W. Johnson, S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal Robot. & Autom.*, vol. 4, nº 2, pp 193-203, 1988.

[11] G. J. Hamlin, R. B. Kelley, and J. Tornero, "Efficient distance calculation using spherically-extended polytope (s-tope) model," in Proc. IEEE Int. Conf. on Robotics and Automation, 1992, pp. 2502-2507.

[12] E. J. Bernabeu and J. Tornero, "Hough transform for distance computation and collision avoidance," *IEEE Trans. Robotics & Automation*, vol. 18, nº. 3, pp. 393-398, 2002.

[13] http://www.mathworld.wolfram.com/Stadium.html

[14] J. H. Mathews, *Numerical Methods for Computer Science, Engineering and Mathematics* Prentice Hall, 1987, pp. 59-74.

[15] http://www.mathworld.wolfram.com/Rose.html.