Accepted for publication in the IEEE SMC-B, Sept. 2003

Solving Minimum Distance Problems with Convex or Concave Bodies Using Combinatorial Global Optimization Algorithms

J. A. Carretero
Department of Mechanical Engineering
University of New Brunswick
Fredericton, NB, E5B 5A3, Canada
Email: Juan.Carretero@unb.ca

M. A. Nahon
Department of Mechanical Engineering
McGill University
Montréal, QC, H3A 2K6, Canada
Email: Meyer.Nahon@mcgill.ca

Abstract— **Determining the minimum distance between convex objects is a problem that has been solved using many different approaches. On the other hand, computing the minimum distance between combinations of convex and concave objects is known to be a more complicated problem. Most methods propose to partition the concave object into convex sub-objects and then solve the convex problem between all possible sub-object combinations. This can add a large computational expense to the solution of the minimum distance problem.**

In this paper an optimization-based approach is used to solve the concave problem without the need for partitioning the concave object into convex sub-objects. Since the optimization problem is no longer unimodal (*i.e.*, has more than one local minimum point), global optimization techniques are used.

Simulated Annealing and Genetic Algorithms are used to solve the concave minimum distance problem. In order to reduce the computational expense, it is proposed to replace the objects' geometry by a set of points on the surface of each body. This reduces the problem to an unconstrained combinatorial optimization problem where the combination of points (one on the surface of each body) that minimizes the distance will be the solution. Additionally, if the surface points are set as the nodes of a surface mesh, it is possible to accelerate the convergence of the global optimization algorithm by using a gradient-based local optimization algorithm. Some examples using these novel approaches are presented.

Keywords— **Distance determination, path planning, Genetic Algorithms applications, Simulated Annealing applications, Combinatorial optimization.**

I. Introduction

Probably the most popular use of the distance determination algorithms is in robot path planning [Liu and Mayne, 1990] where the trajectory of a robot through a workspace with objects is planned in order to have a collision free trajectory. Simulation of physical systems and virtual experimentation are other applications for the minimum distance problem. In many cases the simulation of physical systems requires the use of some form of contact dynamics model where the separation or interference distance between the objects is required [Nahon et al., 1998] (see Figure 1). Similarly, CAD / CAM applications, such as mechanical assembly, tool path planning, virtual reality and virtual prototyping require the use of distance determination algorithms or interference identification in order to make the task possible or the virtual models more realistic.

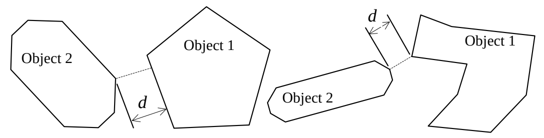

Fig. 1. Examples of the minimum separation distance between convex and concave objects.

Some of the above applications require the minimum distance problem to be solved at many instants as the scenario changes in time, *e.g.*, a robot moving a payload amongst objects. In other applications, only the fact that the objects are interfering or not is of interest, *i.e.*, no detailed information is needed about separation or interference (*e.g.*, mechanical assembly). On the other hand, in contact dynamics, a detailed knowledge of the interference geometry (interference distance and volume, *etc.*) is desirable.

A. Minimum distance for convex objects

Interference detection algorithms, where two objects are checked for interference, such as the one described by Maruyama [Maruyama, 1972], are the predecessors of some of the distance determination algorithms used nowadays. Maruyama's interference method exhaustively checks the boundaries of every pair of faces for interference. If any two faces interfere, then the objects are said to interfere.

Some analytical methods, *e.g.*, [Boyse, 1979], exhaustively compute the distance between edges and vertices of one object and those on the other object. The minimum distance is then determined by comparing all the obtained distances.

Other methods, such as the one described in [Meyer, 1986], compute the separation distance between bounding boxes. This method leads to an approximation but has the asset of being very fast.

Computational geometry theory has been applied to develop methods to solve the distance determination problem. These methods, also relying on an exhaustive feature to feature search, include ones using *Voronoi regions* [Lin and Canny, 1991] and the very popular *GJK* method relying on the *Minkowski subtraction* [Gilbert et al., 1988]. These methods have shown fast results especially when combined with effective pruning strategies [Gottschalk et al., 1996]. On the other hand, the benefit of pruning strategies decreases significantly in densely packed environments [Gilbert et al., 1988] such as the ones sometimes found in contact dynamics applications.

Finally, other methods rely on the use of numerical optimization techniques to find the solution [Bobrow, 1989]. In this case, a single point is located inside each object described by a set of inequality constraints. Then, the optimization proceeds where the objective is to find the set of coordinated of the pair of points that minimizes the distance between the points while satisfying all the constraints. Note that a point that satisfies all the constraints that define the geometry of the object is considered to be a point inside or on the object's surface. This method can also be applied to quadratically constrained objects [Ma and Nahon, 1992].

Recently, a method based on local descent and geometric projections was proposed in [Llanas et al., 2003] for disjoint convex polyhedra. The algorithm starts from a point on the surface of one of the two objects and projects it onto the second object. Once the projected point is found, the process is repeated by projecting it back onto the first object. In [Llanas et al., 2003], the authors claim that in most cases the algorithm converges within a couple of iterations. This is with the exception of cases when two faces are parallel or quasi-parallel. To overcome such problem, the authors have added some extra features to their algorithm and report some promising result for highly complicated environments with a combined total number of vertices of up $24,000$.

In brief, some fast algorithms sacrifice precision for speed while most are limited in the types of objects that can be handled (*e.g.* linearly bound objects, quadratically bound objects).

B. Minimum distance for concave objects

B.1 Partitioning methods

Most works have reported the use of convex partitioning of concave bodies (see Figure 2) to solve the distance determination problem between concave bodies. In this scenario, any concave object is partitioned into convex sub-objects, usually off-line. The distance problem is solved for every sub-object pair and the minimum of all the distances obtained is used as the solution of the concave problem [Ma and Nahon, 1992] [Nahon et al., 1998] [Gilbert et al., 1988] [Ponamgi et al., 1997].

While these methods work and rely on very fast and robust convex methods, the computational expense becomes a problem when dealing with complicated concave objects. Let us consider the following example: if the minimum distance between two concave objects, each of which is partitioned into ten convex sub-objects, is to be solved, then, a total of one hundred convex minimum distance problems would need to be solved. The number of convex sub-objects can be very large, particularly for higher order objects that need to linearized before they are partitioned into convex pieces.

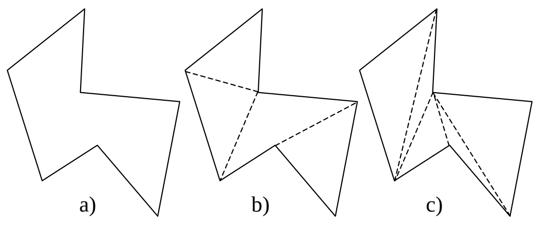

Fig. 2. Convex partitioning of a concave body, a) shows the original object, b) and c) show two different convex partitions of the same object.

Recently, face decomposition, as opposed to 3D solid decomposition as described earlier, has also been used for complicated concave polyhedral geometries. This, together with efficient pruning algorithms based on hierarchical representation of objects, have been used to solve the separation distance problem amongst complicated concave polyhedral geometries [Ehmann and Lin, 2001].

B.2 Non-partitioning methods

An interference detection scheme based on Boyse's scheme [Boyse, 1979] for convex polyhedra was extended to non-convex polyhedra in [Abdel-Malek and Burton, 1994], where every pair of surfaces is checked for interference. The line of intersection between pairs of planes (containing each of the faces in question) is first obtained. The line is divided in small segments that reach the faces' limits. If any segment is fully contained within the boundaries of both faces then the two faces are said to interfere except when one of the surfaces is a hole. If the face has a hole (a concavity) it is modeled as a negative entity, thus if a line segment is contained within the boundary of that face there is no interference. This method only returns a boolean result, *i.e.*, whether or not there is interference, but does not quantify the minimum distance.

The advantages of decomposition-free approaches include *i)* the minimization of the number of object pairs to be tested since the addition of extra features (surfaces, edges, vertices, *etc.*) is avoided and *ii)* the solution of the problem in a single run rather than solving the distance problem amongst all pairs of sub-objects. Both advantages can potentially reduce the number of computations needed to solve the concave problem.

In this paper, a novel optimization-based method for solving the minimum distance between any combination of objects is introduced. The proposed method makes use of combinatorial global optimization techniques such as Genetic Algorithms (GA) and Simulated Annealing (SA). Additionally, some specifics of the implementation of the proposed methods are presented paying particular attention to the genetic operators used in the GA implementation as well as the operators used in the Simulated Annealing process. Moreover, a local hill-climbing optimization technique used to improve the convergence speed of the global optimization algorithms is also described. Finally, the novel algorithms are used to solve the minimum distance problem in a few numerical examples where the capabilities of the method are illustrated.

II. Solving the distance problem

As discussed earlier, one of the methods used to solve the minimum distance problem between convex objects is formulated as a constrained optimization problem [Bobrow, 1989]. The method locates one point inside or on the surface of each body. Then, using the Cartesian positions of each of these two points as search variables, the Euclidean distance between the two points is minimized subject to a set of inequality constraints (defining the geometry of each body). This problem can be expressed as follows [Ma and Nahon, 1992]:

$$\text{minimize} : \sqrt{(\mathbf{p}_1 - \mathbf{p}_2)^T(\mathbf{p}_1 - \mathbf{p}_2)} \tag{1a}$$

$$\begin{aligned} \text{subject to} : &\ \mathbf{g}_j(\mathbf{p}_1) \leq 0 \text{ for } j = 1, \ldots, k \qquad (1b)\\ : &\ \mathbf{g}_j(\mathbf{p}_2) \leq 0 \text{ for } j = k+1, \ldots, n_{constr} \end{aligned}$$

where $\mathbf{p}_i = [\ x_i \quad y_i \quad z_i\ ]^T$ corresponds to the Cartesian coordinates of the point on the i-th body, $\mathbf{g}_j(\mathbf{p}_i) \leq 0$ are the *low level geometry primitives* defining the geometry of both bodies. n_{constr} is the total number of constraints, that is, k constraints to describe object 1 and $n_{constr} - k$ to describe object 2. Each constraint represents a single linear or higher order surface of the object. Note that $\mathbf{p}_i$ and $\mathbf{g}_j$ must be expressed in the same coordinate frame.

In the convex case, the previous problem has a unique solution (*i.e.*, the objective function is unimodal) and can be solved using quadratic programming techniques when the constraints are linear [Ma and Nahon, 1992]. On the other hand, when at least one of the objects is concave [1], the objective function may no longer be unimodal, *i.e.*, more than one local minimum solution may exist.

It is important to notice that each time a point is generated it has to be checked for feasibility, *i.e.*, to ensure the point is inside or on the surface of a body. This entails evaluating all the constraints in equation (1b) at that particular trial point. For example, if one had two simple objects defined by 20 constraints each, the number of multiplications and additions (*i.e.*, flops) to verify feasibility would be 240, whereas only 11 flops are needed to evaluate the objective function. Thus, in this example, the number of flops required to evaluate the objective function in this simple example is about 22 times smaller than the number required to check the point's feasibility.

A. Combinatorial optimization approach

To avoid having to calculate the constraints at every iteration, it is proposed to replace the geometry of the objects by a finite number of points on the surface of the convex or concave object. Thus, if a fixed number of points is given, the minimum distance problem is reduced to a combinatorial optimization problem. That is: which is the best combination of two points (one on each body) which yields the minimum distance between them. Since the points on the surfaces of each body can be generated off-line, the number of algebraic calculations per iteration will be substantially reduced.

Although this method eliminates the computational expense involved in checking the feasibility of any point, it poses some accuracy problems. That is, the solution obtained will be an approximation to the real solution which will be governed by the number of points on the surface of each object and the distribution of these points. This method also poses implementation challenges such as the creation evenly-distributed points on the entire surface of both bodies. This issue can be addressed by using meshes to represent the surface of each body which can be obtained with several free or commercial software packages (see for instance [Joe, 1986] or [CIMNE, 2003]).

The use of meshes also allows the possibility of using a local minimum search. That is, at each iteration, any combination of points obtained at random, could be locally optimized by moving the points along the edges of the mesh using a hill-climbing based search. This is possible since any point on the mesh is 'aware' of its neighbours. An unstructured mesh allows motions of a single step at a time, *i.e.*, the rate of change can be calculated at every point in the direction of all its neighbours and the rate with the largest change in the negative direction is used as the direction where the point moves next.

A.1 Mesh storage

Typically, meshes are stored in matrices. The first of these matrices is the node matrix $\mathbf{N}$ that contains the Cartesian coordinates of all v mesh points (*a.k.a.* nodes) with respect to a body fixed frame. That is, each row of $\mathbf{N}$ corresponds to one point and the columns contain the x, y and z coordinates of that point. The second matrix is called the element or facet matrix $\mathbf{F}$ and gives a list of all the facets of the mesh. Thus, the matrix $\mathbf{F}$ has as many rows as the number of facets on the mesh. Each row on the matrix $\mathbf{F}$ contains m integer elements that correspond to the m vertices of the polygonal facet. Thus, a surface mesh with triangular facets would have an element matrix $\mathbf{F}$ with three columns. Note that the value of each entry in the facet matrix $\mathbf{F}$ corresponds to a particular point (*i.e.*, row) in the node matrix $\mathbf{N}$. That is, if the 10^{th} row of $\mathbf{F}$ is $[\ 8 \quad 5 \quad 18\ ]$, it means the tenth facet of the mesh has nodes 8, 5 and 18 as its vertices which coordinates are in rows 8, 5 and 18 of matrix $\mathbf{N}$.

In the present work, a third matrix, a connectivity matrix $\mathbf{\Pi}$, is also stored with each mesh. The matrix $\mathbf{\Pi}$ is a square matrix with as many rows and columns as node points on the mesh, *i.e.*, v. The connectivity matrix $\mathbf{\Pi}$ only contains binary elements which correspond to the particular elements of the mesh that are connected to each other.

[1]Special care must be taken when representing the geometry of a concave object since the simple union of inequality constraints method described in equation (1b) can only describe convex objects [Carretero and Nahon, 2001].

That is, if element $\mathbf{\Pi}_{[i,j]} = 1$ means that nodes n_i and n_j are connected to each other. Since $\mathbf{\Pi}$ is a sparse matrix, it can be stored more compactly using a cell array with v rows. Each row only contains the integer numbers corresponding to the adjacent nodes. That is, to see what all the connecting points to node n_i are, one has to look at row i of $\mathbf{\Pi}$.

A.2 Mesh distances (distance and predecessor matrices)

Several methods for obtaining distance in meshes/networks are known, see for instance [Tanenbaum, 1996]. In the present work, Dijkstra's algorithm [Dijkstra, 1959] to obtain the distance and predecessor matrices for the meshes is used. These two matrices will allow to obtain detailed information about the mesh required by the optimization algorithms. These two matrices determine the shortest distance and path between two elements of the same mesh. This process is computationally expensive but, in the framework of the work proposed here, this is not crucial since it is performed off-line. That is, these two matrices would only need to be obtained once for each object and the results stored for later use in the minimum distance algorithms.

Following is a brief description of the distance and predecessor matrices. For a mesh with v nodes, the distance and predecessor matrices ($\mathbf{D}$ and $\mathbf{P}$, respectively) will both be of size $v \times v$. For $\mathbf{D}$ and $\mathbf{P}$, the row indices correspond to source points and the columns indices to destination points. Thus, the distance δ between nodes n_s and n_t (*i.e.*, $\delta_{s\to t}$) is stored in $\mathbf{D}_{[s,t]}$, *i.e.*, $\delta_{s\to t} = \mathbf{D}_{[s,t]}$. Thus, the symmetric distance matrix $\mathbf{D}$ contains the shortest distance between every possible pair of points on the mesh.

On the other hand, in order to obtain the shortest path between two nodes on the mesh, namely n_s and n_t, element $[s,t]$ of matrix $\mathbf{P}$ contains the last element to be reached in the path from point n_s to point n_t, namely n_k. That is, in order to reach the destination point n_t from the source point n_s, it is necessary to first reach node n_k (see Figure 3). Then, to get the element in the path before n_k that need to be reached, it is necessary to look at $\mathbf{P}_{[s,k]}$ which contains another element n_{k-1}. The process would be repeated until the node n_s is found as the entry $\mathbf{P}_{[s,k-j+1]}$. Thus, the path $\mathbf{p}$ to go from n_s to n_t is $\mathbf{p}_{n_s\to n_t} = [n_s, n_{k-j+1}, \ldots, n_{k-1}, n_k, n_t]$. It follows that the distance $\delta_{s\to t}$ is equal to the addition of the distance between the intermediate nodes, that is, $\delta_{s\to t} = \delta_{s\to k-j+1} + \ldots \delta_{k-1\to k} + \delta_{k\to t}$.

A.3 Objective function

Since the constraints have been eliminated by the use of meshes, we are now left with an unconstrained optimization problem which is formulated as

$$\min : d^2 = (\mathbf{p}_1 - \mathbf{p}_2)^T(\mathbf{p}_1 - \mathbf{p}_2) \tag{2}$$

Notice that, in equation (2), it is the square of the distance that is minimized and not the distance d itself. This is done to avoid the computational expense of calculating the square root in equation (1a) which can be done since minimizing a is equivalent to minimizing $\sqrt{a}$ if a is a positive number.

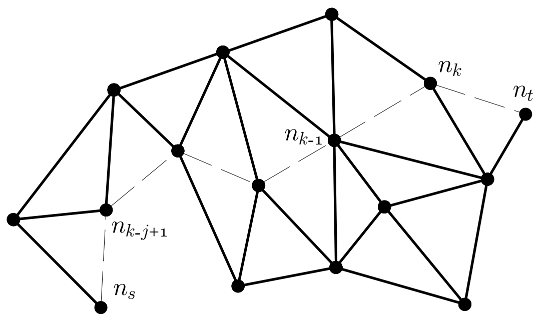

Fig. 3. Minimum distance path to reach destination node n_t from source node n_s.

As in [Carretero and Nahon, 2001] and [Carretero et al., 2001], the formulation proposed here uses Genetic Algorithms and Simulated Annealing to solve the minimum distance problem. As will be seen in further sections, these methods were selected over other global optimization methods due to their proven capabilities in multimodal function optimization (*e.g.* [Goldberg, 1989] and [Aarts and Korst, 1989]).

Note that the method proposed here, as opposed to the other optimization based approaches presented in [Bobrow, 1989] and [Ma and Nahon, 1992], can handle minimum distance problems with concave objects without the need of partitioning the concave objects into convex sub-pieces. As mentioned earlier, this can have the advantage of not having to introduce unnecessary fictitious surfaces. Additionally, the proposed method solves an unconstrained optimization problem whereas in [Bobrow, 1989] and [Ma and Nahon, 1992] the minimum distance problem is formulated as a constrained optimization problem thus having to evaluate the constraints at run time. As a result, these two advantages would certainly prove powerful particularly in complex scenarios as complicated concave objects would not need to be partitioned into large number of pieces.

B. Global optimization

If a function having more than one local minimum (*a.k.a.* multimodal function) is to be optimized using a gradient-based search algorithm, the solution obtained may or may not be the global minimum. When the start point for a gradient based method is close to a local minimum, the algorithm will quickly move toward it and eventually converge there, *i.e.* the algorithm is trapped in the local minimum. This means that if a different start point is used, a different solution may be found. Figure 4 illustrates this with a single variable function. Often, a multi start search is used to bolster the confidence in the minimum value obtained using gradient-based approaches (see for instance [Carretero et al., 2000]).

As an alternative to gradient-based approaches, a variety of different *global* optimization methods have been proposed in the literature. Amongst the most popular

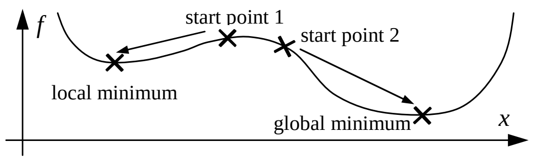

Fig. 4. Local versus global optimum.

techniques, one can find the Genetic Algorithms (GA) and Simulated Annealing (SA) [Davis and Steenstrup, 1987]. These two methods are briefly described here as they were both used to solve the minimum distance problem. Later, in Section IV, result obtained from both algorithms are compared.

B.1 Simulated annealing

Simulated Annealing (SA) is a global optimization method which is based on the analogy of the cooling process of molten metals [Kirkpatrick et al., 1983]. Briefly, this method works on the principle of randomly generating new points that are accepted if the objective function is improved. On the other hand, if the objective function is not improved the new point is rejected unless a randomly generated number is inferior to a probability inversely proportional to the iteration number of getting accepted. This randomness is included to allow the search to escape from local minima [Rutenbar, 1989]. This method has proven to be powerful in several applications, see for instance [Buchal et al., 1989] [Martinez-Alfaro et al., 1998].

The basic algorithm is described in point form as follows [Kirkpatrick et al., 1983]:
1. Randomly create an initial guess and evaluate the objective function.
2. Randomly generate a new guess and evaluate the objective function.
3. If the objective function at the new guess is improved, increase the iteration number (in SA parlance: decrease the temperature) and go to step 2 unless the algorithm has converged or the maximum iteration number is exceeded. If the objective function at the new point is worse, continue to step 4.
4. Generate a normally distributed random number ($0 \leqslant \rho \leqslant 1$) and compare it to a Boltzmann probability function. If the random number is smaller than the probability factor (*i.e.*, $\rho < p_b$) accept the new point and return to step 2, otherwise return to step 2. This acceptance criterion is referred to as the *Metropolis criterion.*

The most important aspect of the SA process is the *cooling schedule*: *i.e.* the expression defining the Boltzmann probability function (step number 4 described above). A Boltzmann probability function is one that decreases as the iteration number increases such as

$$p_b = e^{\frac{f(\mathbf{x}_{i-1}) - f(\mathbf{x}_i)}{k_B t}} \tag{3}$$

where $f(\mathbf{x}_i)$ is the objective function evaluated at trial point $\mathbf{x}_i$, k_B is Boltzmann constant and t refers to the temperature which decreases as the iteration number increases.

A highly exploratory algorithm has a very slow cooling schedule accepting points with poor objective function even after many iterations have passed. Once enough iterations have passed, only a few moves that worsen the objective function are accepted. By the end of the SA process, only moves that improve the objective function are accepted.

B.2 Genetic algorithms

Another popular global optimization technique is Genetic Algorithms (GAs). These algorithms mimic the natural evolution processes by natural selection and ‘survival of the fittest’ methods outlined by Charles Darwin in the 19th century. The first complete work published stating the basic principles of Genetic Algorithms was presented by Holland in 1975 [Holland, 1975]. Although little theory has been developed in the area of why GAs work so well in many problems, their use in different areas is growing quickly. As described in [Goldberg, 1989], the most common application of GAs is in the areas of function optimization, machine learning, robot path planning and artificial intelligence [Grefenstette, 1987]. A good introduction to GAs is given in [Beasley et al., 1993a].

The main steps in the original GA described by Holland (often referred to as the *canonical genetic algorithm*) are described as follows:
1. *Initialization*: Randomly create an initial population of size N_{pop}
2. *Evaluation*: Compute the fitness of all individuals in the population
3. *Selection*: Based on their fitness, select individuals that will be used for mating
4. *Mating* (*crossover*): Recombine population to produce N_{pop} offspring
5. *Mutation*: Randomly mutate the population
6. Repeat steps 2 through 6 until a maximum number of generations (N_{gen}) is reached or convergence is detected.

A good part of the research on this area involves the development of new and more efficient mating, mutation and selection methods for particular applications (see for instance [Davis and Steenstrup, 1987] and [Beasley et al., 1993b]).

III. Implementation

Although many examples of SA and GAs can be found in literature, these stochastic methods have to be tailored to the particular problem in order to solve it more efficiently than with a generic implementation [Beasley et al., 1993b]. This section describes the most relevant aspects of the implemented algorithms.

A. Encoding

One of the most important aspects of the implementation of any SA or GA optimization is the encoding of the trial points, *i.e.* the search variables. In the present problem, the encoding was done by having only a set of two indices each referring to a particular point on the surface of each body. That is, trial point i is expressed as $\mathbf{x}_i = [\ n_1^i \quad n_2^i\ ]$ where n_j^i contains the row number of the node matrix $\mathbf{N}_j$ of body j where the coordinates of point n_j^i are stored.

It should be noted that to calculate d^2 (equation (2)), both points must be expressed with respect to a common frame. In order to reduce the computational expense of expressing both points with respect to the inertial frame, it is possible to express both points with respect to the same body fixed frame, namely frame Σ_1 attached to body 1. This transformation is possible without affecting the distance calculation since the Euclidean distance is frame invariant. Thus, only the points extracted from the database of body 2 have to be multiplied by the homogeneous transform describing body 2 with respect to body 1 as follows

$$\mathbf{T}_2^1 \begin{bmatrix} (\mathbf{N}_{2_j})^T \\ 1 \end{bmatrix} = \mathbf{T}_0^1 \mathbf{T}_2^0 \begin{bmatrix} (\mathbf{N}_{2_j})^T \\ 1 \end{bmatrix} \tag{4}$$

where $\mathbf{T}_b^a$ corresponds to the homogeneous transform that describes frame b with respect to frame a and $\mathbf{N}_{2_j}$ corresponds to the point $\mathbf{p}_2$ stored in the node matrix $\mathbf{N}_2$, *i.e.*, row j of matrix $\mathbf{N}_2$. The extra row containing 1 used to express $\mathbf{N}_{2_j}$ is required in order to keep the dimensions consistent. The use of this method was found to reduce the overall computational expense between 30 to 50% resulting in a total of 26 flops per d^2 evaluation.

B. Genetic operations

B.1 Selection

The selection process in a GA is essential in order to exploit promising regions of the space. At the same time, by randomly selecting some poor individuals to proceed to the next generation, it creates the population diversity needed for the exploration of the entire search space. For this combinatorial approach, it is proposed to use Stochastic Universal Sampling [Baker, 1987] (*a.k.a.* modified roulette wheel selection) which is a simple selection method that has shown good results in many applications such as function optimization.

B.2 Mesh mating

The mating process of a GA is the mechanism that allows it to pass to the next generations the traits of the selected individuals.

In the present work, *mesh mating* is used, which involves manipulation of the distance and predecessor matrices. Once two individuals $\mathbf{x}_1 = [\ n_1^1 \quad n_2^1\ ]$ and $\mathbf{x}_2 = [\ n_1^2 \quad n_2^2\ ]$ have been picked for mating, their nodes are mated pair wise[2]. That is, the first node of $\mathbf{x}_1$ is paired with the first node of $\mathbf{x}_2$ and similarly for the second node of each parent. Thus, the rest of this analysis only considers the first gene (node) of each solution point to describe the *mesh mating* method.

The path between the selected nodes n_1^1 and n_1^2 ($\mathbf{p}_{n_1^1 \to n_1^2}$) is obtained using the method described in Section II-A.2

[2]Note that the superscript in n_j^i denotes the trial point $\mathbf{x}_i$ to which n_j belongs to. That is, n_i^j is the n-th node in the object's j mesh which belongs to trial point $\mathbf{x}_i$.

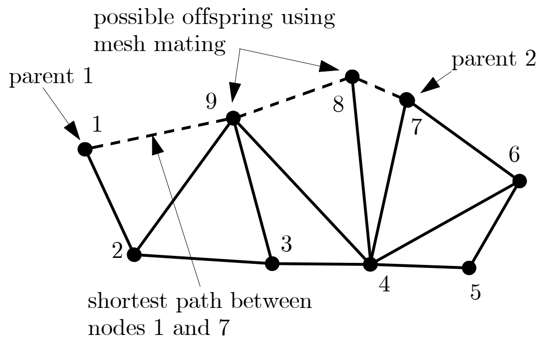

Fig. 5. Mesh mating example for a simple mesh.

and all nodes within the path are considered to be eligible as offspring. To select the offspring, a particular radius from one of the parents is used to determine the maximum mesh distance between that parent and the offspring. This radius is centered around the parent with best fitness, namely $\mathbf{x}_1$, and its size r is weighted to give preference to the parent with better fitness as follows:

$$r = \delta_{1\to 2} \frac{d_1^2}{d_1^2 + d_2^2} \tag{5}$$

where $\delta_{1\to 2}$ is the mesh distance from node n_1^1 to node n_1^2 and d_i^2 is the square of the distance associated with individual $\mathbf{x}_i$. Figure 5 shows an example where two parents (nodes 1 and 8) are mated to produce an offspring (node 9).

B.3 Mutation

The mutation of trial point $\mathbf{x}$ is little more than a random displacement of the trial point $\mathbf{x}$. For this purpose, the use of the *random move* is proposed here.

The method uses mesh information such as the distance and predecessor matrices described in previous sections. First, if a point n_i on mesh i is to be modified, a new random point ($n_{i_{\text{rand}}}$) is generated. Next, the path between n_i and $n_{i_{\text{rand}}}$ is determined using the predecessor matrix $\mathbf{P}$, that is $\mathbf{p}_{n_i \to n_{i_{\text{rand}}}}$ is obtained. Finally, one of the points along the path $\mathbf{p}_{n_i \to n_{i_{\text{rand}}}}$ within a user defined radius $r_{\max}$ from n_i is selected as the trial point $n_{i_{new}}$. The radius $r_{\max}$ is set as a function of the maximum mesh distance in the entire mesh which itself is a function of the object's size. Note that the present method uses the mesh distance δ between n_i and $n_{i_{new}}$ (*i.e.*, $\delta_{n_i \longrightarrow n_{i_{new}}}$) and not the Cartesian distance since it is the mesh distance that is stored in the distance matrix $\mathbf{D}$. In order to allow points in early generations to explore new regions of the solution space, the maximum move radius $r_{\max}$ could be set relatively large. Then, as the generation number increases, $r_{\max}$ could be reduced using a linear function or a Boltzmann probability factor. In the present case $r_{\max}$ was fixed to $\max(\mathbf{D})/2$.

C. Simulated annealing operations

C.1 Cooling schedule

A Boltzmann probability function expressed as in equation (3) was used. The Boltzmann constant is usually found by experimentation, *i.e.* by allowing the SA process run a few times and making sure there is no premature or excessively slow convergence [Kirkpatrick et al., 1983]. Additionally, the choice of the initial temperature t_{ini} determines for how long the algorithms is allowed to explore new regions rather than concentrating on improving the local solution.

C.2 Generation of new points

The exploration and exploitation in the SA algorithm is achieved by generating new points in the solution space. Thus, the development of mechanisms to generate new points is of vital importance to the SA algorithm.

In the present work, the mutation method presented earlier is used to generate new points during the SA process. As for the mutation operation, the maximum radius which the points are allowed to be displaced is scaled down as the iteration number increases (*i.e.*, the temperature decreases) using a Boltzmann type variation.

D. Initial guess

Numerical optimization algorithms require an initial guess or start point for the optimization routine to start the search. In most cases a good initial guess accelerates the convergence of the optimization routine. If the distance determination problem is solved repeatedly over time, as is common in dynamics simulation, it is possible to use the solution of the previous time step as the initial guess for the new time step [Ma and Nahon, 1992]. It is also possible to include velocity information to further improve the initial guess [Nahon, 1993].

On the other hand, during the first run of the minimum distance algorithm, *i.e.*, at the start of a simulation, the initial guess is not known. In this work, it is assumed that there is no previous knowledge of the solution and so, the initial guess has to be generated by the algorithm.

For the GA and SA implementations, one method to generate the initial trial points is at random where all node points are possible candidates and the pairs are selected at random. Due to the nature of the minimum distance problem, the salient points of the objects are potentially good candidates as initial guess points. That is, a better initial guess could be made by picking pairs of points that include points on vertices, edges and centroid of the faces of the objects.

In the present case, in order to evaluate the worst case scenario, the initial trial points are generated at random at every single run.

E. Local optimization

Based on the information that can be extracted from the unstructured mesh, a local optimization procedure was implemented. As mentioned in Section II-A.1, a connectivity matrix $\mathbf{\Pi}$ is obtained when the mesh is created that provides the point index of all the neighbouring points to a particular node.

Using the connectivity matrix this method tried to reduce the distance between the two selected mesh nodes by moving them along the edges of the mesh, *i.e.* to a neighbouring node. First the selected node on object one (n_i) is moved successively to each of its neighbours while the node on object two (n_j) is fixed. The neighbour of n_i that minimizes the distance between n_i and n_j is accepted as the new point n_i and the search for a closer neighbouring point starts again. Once the best n_i is found, the process is repeated by fixing n_i and moving n_j. This procedure is repeated, alternating the moving and the fixed node until the distance between the nodes is minimized, *i.e.*, until no neighbouring nodes can be found that reduce the distance between n_i and n_j.

Since this is a hill-climbing based method, it is clear that, on the local scale, the solution will greatly depend on the start point. However, in the larger scale, the SA and GA algorithms are expected to introduce the necessary ability to jump out of local minima 'traps'.

Note that, if the objects in question are convex, the local optimization algorithm suffices to obtain the global solution since for convex problems there is a unique local solution that is also the global solution.

Also note that, Genetic Algorithms that make use of local optimization techniques at every iteration are referred to as Memetic Algorithms which evolve using what in GA parlance is referred to as Lamarckian evolution (see for instance [Whitley et al., 1994]).

IV. Examples

This section presents three numerical examples where the distance determination algorithms described in this paper are applied to solve the minimum distance problem between a pair of objects. Each example uses a different set of objects. The first two examples were chosen since they demonstrate the algorithms' basic capabilities. On the other hand, the third example offers a more challenging problem as it has a much larger number of minima. Other examples can bee seen in [Carretero, 2002].

To evaluate the capabilities of the proposed algorithms under normal conditions, the following examples solve the minimum distance problem using randomly generated starting points. This allows us to evaluate, in terms of computational efficiency, the worst case scenario since no prior knowledge of the location of the minimum solution is assumed. Additionally, for every example presented here, the random number generator was reset before each set of tests.

Note that, in order to evaluate and compare the algorithms' performance, the number of flops required for each test are presented with each set of results. The number of computations was estimated using special built-in functions and commands within the interpreter for the programming language that was used in the numerical implementation.

In order to classify the algorithms evaluated here, the fol-

Parameter	Symbol	Value
population size	N_{pop}	50
probability of mutation	p_m	0.02
probability of crossover	p_x	0.6
max. number of generations	G_{max}	50
Boltzmann constant	k_B	0.0001
Initial temperature	t_{ini}	2500

TABLE I
Parameters for all comb-SA and comb-GA runs.

Algorithm	Position 1 minimum	Global region	kflops	Position 2 minimum	G re
comb-GA-wl	0.0757	✓	1 852	0.0997	
comb-GA-nl	0.2345	×	147	0.244	
comb-SA-wl	0.0757	✓	8 209	0.0997	
comb-SA-nl	0.1138	✓	113	0.1746	
enumeration	0.0757			0.0997	
exact (± 0.00005)	0.0747			0.0987	

TABLE III
Results for the battery and fixture example.

Object name	Battery	Fixture
overall dimensions (cm)	$92 \times 74 \times 31$	$62 \times 86 \times 99$
grid size (cm)	0.05	0.05
number of nodes	1241	1842
number of facets	2478	3680
convex sub-pieces	14	9

TABLE II
Battery and fixture geometry details.

lowing naming technique was adopted. First, the first three letters of the approach name are used (*i.e.*, 'comb' for combinatorial), second the optimization algorithm acronym is used (*i.e.*, GA or SA) and finally two letters identifying if local optimization is used or not are added (*i.e.*, 'wl' for **w**ith **l**ocal optimization and 'nl' for **n**o **l**ocal optimization). This results in algorithm names such as: comb-GA-wl, which states that a genetic algorithm without local optimization using the combinatorial approach is used.

In the following examples four algorithms are tested, these are the GA and SA implementations of the proposed minimum distance algorithm each with two variants depending on the use of local optimization or not. The SA and GA parameters used in the examples are listed in Table I. Although during the course of the project many different sets of parameters were tested [Carretero, 2002], only the ones listed in Table I are presented here since they were found to be a good balance between exploration and exploitation. Note that, in order to allow a comparison between the results of the SA and GA algorithms, the initial temperature (t_{ini}) for the SA examples was set equal to the total number of trials the GA performs, *i.e.*, $t_{ini} = G_{max} \times N_{pop}$ where G_{max} is the maximum number of generations and N_{pop} is the population size.

A. Battery and fixture

In this first example, relatively simple geometries are used. The objects shown in Figure 6 illustrate a simplified version of the Orbital Replacement Unit (ORU) battery and its fixture of the International Space Station [Nahon et al., 1998]. Several relative positions of the objects were tried of which only the results from two are presented here. In both poses analysed here, the battery is close to the fixture and both objects are near to an insertion position.

Surface meshes were created on the two objects and their dimensions are listed in Table II. A visual inspection of this test scenario revealed approximately 9 local minima from which a single one is the global minimum. Note that, as stated in Table II, if the battery and fixture were to be partitioned into convex sub-pieces they will result in approximately 14 and 9 convex pieces, respectively[3]. This, in a more conventional minimum distance algorithms as the one described in [Ma and Nahon, 1992], would result in a total number of sub-object pairs equal to 126.

Figures 7 and 8 show the time history of the solution for the GA and SA algorithms, respectively, for the first position. Note that in both figures, the objective function decreases monotonically. This is due to the fact that at all times the algorithm keeps track of the best set of points obtained throughout the optimization run to make sure the best set is never lost.

Table III lists the final results for the battery and fixture example using comb-GA and comb-SA. Additionally, in order to verify the results from the optimization algorithms, Table III lists the global minimum obtained by enumeration and the exact solution obtained semi-analytically[4]. Note that the column labelled as 'Global region' denotes whether the region where the global minimum is identified was located by the algorithm or not. In both positions, the comb-GA-wl and comb-SA-wl algorithms were capable of obtaining the global minimum. By contrast, the algorithms not using the local optimization had troubles locating the global solution.

It is important to notice that, although in the present example the algorithm comb-GA-wl was capable of finding the global solution in the first iteration, this may not be the case in more complicated scenarios. Thus the use of GAs, rather than multi point local search, will bolster the confidence in finding the global solution. This is illustrated in the following examples.

B. Hand and bowl

The objects shown in Figure 9 illustrate a simplified version of hand which is about to touch a bowl. The mesh

[3]The round alignment pegs (fixture) and holes (battery) where linearized using only 6 surfaces.

[4]The closest regions are identified using the enumeration method. Then, CAD regular drawing tools are used to approximate the shortest line intersecting the two objects. The length of such line is then measured and reported as the 'exact' minimum distance.

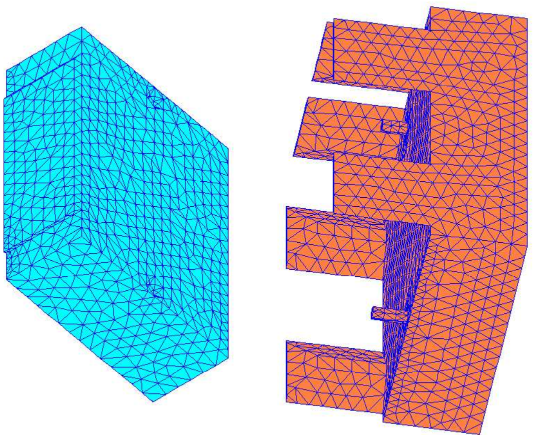

Fig. 6. Mesh representation of the battery (1241 nodes) and fixture (1842 nodes).

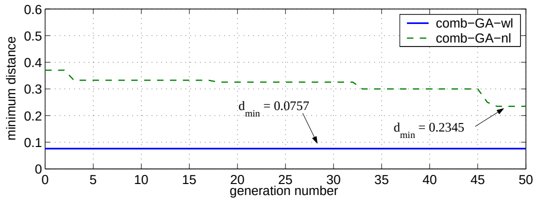

Fig. 7. Time history of the best individual in the population for comb-GA-wl and comb-GA-nl (ORU battery and fixture example).

Object name	Hand	Bowl
overall dimensions (mm)	130 × 185 × 90	∅300 × 60
grid size (mm)	[3 – 30]	10
number of nodes	2236	3855
number of facets	4362	7706
convex sub-pieces	196	73

TABLE IV
Hand and bowl geometry details.

Algorithm	Position 1 minimum	Global region	kflops
comb-GA-wl	4.8295	✓	2 430
comb-GA-nl	12.433	×	155
comb-SA-wl	4.8295	✓	11 228
comb-SA-nl	16.221	×	115
enumeration	4.8295		
exact (±0.00005)	4.7527		

TABLE V
Results for the hand and bowl example.

for the hand was created with a variable size as more precise distance calculations where desired near the fingertips whereas the mesh on the back of the hand was less dense. Conversely, the bowl's mesh was created using an evenly distributed grid size. More details about the test geometries are listed on Table IV. It is expected that in the current problem, as many as seven local minima would occur from which a single one is the global.

Note that, according to the data listed on Table IV, the current models of the hand and bowl would be partitioned into approximately 196 and 73 convex pieces, respectively[5]. This, in more conventional minimum distance algorithms would result on a total number of sub-object pairs to be checked exceeding 14 thousand.

A few relative positions of the objects were analysed with satisfactory results and only one of them is presented here (the one pictured in Figure 9). In such pose, the global minimum region is located where the tip of the middle finger is the closest to the upper section of the bowl's rim.

Figures 10 and 11 show the time history of the solution for the GA and SA algorithms, respectively. Additionally, Table V lists the final results for the hand and bowl example using comb-GA and comb-SA. In this case, the comb-GA-wl and comb-SA-wl algorithms were both capable of obtaining the global minimum. By contrast, as demonstrated also in the earlier example, the algorithms not using the local optimization had troubles locating the region where global solution is located.

[5]The bowl was linearized using 36 radial segments.

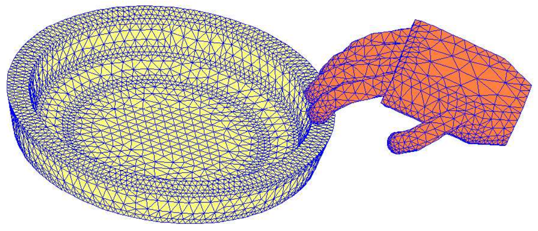

Fig. 9. Mesh representation of the hand (2236 nodes) and bowl (3855 nodes).

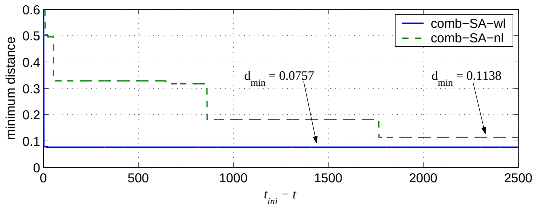

Fig. 8. Time history of the minimum distance of the trial point for comb-SA-wl and comb-SA-nl (ORU battery and fixture example).

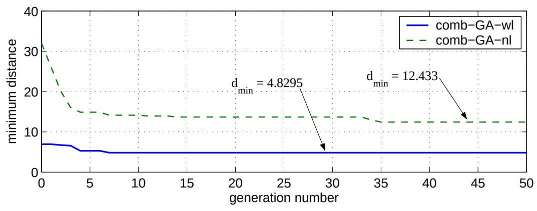

Fig. 10. Time history of the best individual in the population for comb-GA-wl and comb-GA-nl (hand and bowl example).

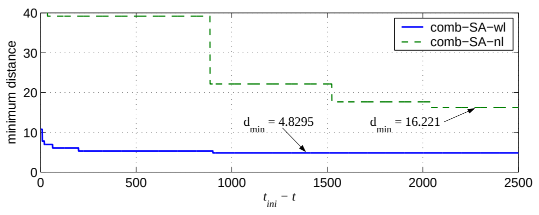

Fig. 11. Time history of the minimum distance of the trial point for comb-SA-wl and comb-SA-nl (hand and bowl example).

Algorithm	Position 1 minimum	Global region	kflops
comb-GA-wl	19.3163	✓	2 140
comb-GA-nl	28.8090	×	152
comb-SA-wl	19.3163	✓	9 716
comb-SA-nl	24.9761	×	112
enumeration	19.3163		
exact	˜18.95		

TABLE VI

Results for the handshake example.

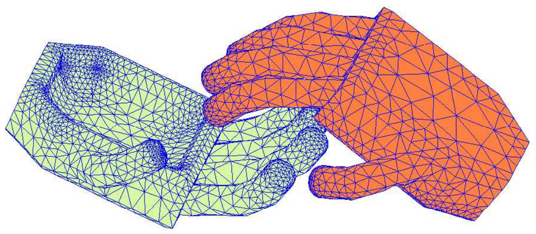

Fig. 12. Mesh representation of the handshake example (2236 nodes on each hand).

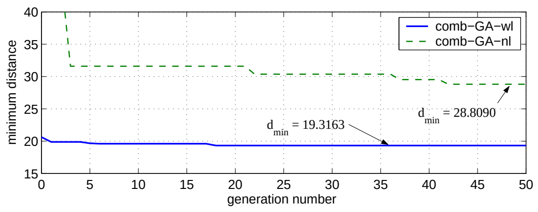

Fig. 13. Time history of the best individual in the population for comb-GA-wl and comb-GA-nl (handshake example).

C. Handshake

As seen in Figure 12, this last example uses two hands about to do a handshake. The mesh for both hands was used identical to the hand used in the previous example which details are listed in Table IV. Additionally, all optimization parameters were kept identical to the ones used in previous examples (see Table I). In the present case, a visual inspection of the problem revealed as many as 50 local minimum from which only one is the global (the combination of the tip of the upper hand's thumb with the tip of the lower hand's index finger). As for the previous example, each hand would be partitioned into around 196 convex sub-objects resulting in a total number of convex sub-object pairs of slightly under 40 thousand.

Figures 13 and 14 show the time history of the solution for the GA and SA algorithms, respectively, whereas Table VI[6] lists the final results. As seen for previous examples, the algorithms not using the local optimization had troubles locating the global minimum region whereas the algorithms making use of the local optimization algorithms where much more successful.

D. Summary of results

In 100 independent runs, it was noticed that the comb-GA-wl was always successful at finding the global solution for all three examples. Additionally the comb-GA-wl was seen to converge in as many as 40 generations where the mean was around 2 generations for the first example, 4 generations for the second example and around 8 generations for the third example. On the other hand, The comb-SA-wl algorithm was seen to have a fair success rate but did not have the robustness of the comb-GA-wl algorithm, particularly for more complicated geometries as the ones presented in the third example (failed on 12 out of 100 runs). In most cases comb-SA-wl converged within the first 100 iterations. Conversely, the comb-GA-nl and comb-SA-nl algorithms improved the solution up until the last few iterations but did not have enough time to reach the global solution (in many cases not even the region where the global solution is located).

It is interesting to notice that, due to the stochastic nature of the optimization methods (namely GA and SA), the algorithms presented here do not guarantee to always converge to the true closest point, particularly when no local optimization is used. However, for the geometries in the previous examples and the ones tested in [Carretero, 2002], the global minimum point was found in the vast majority of the tests where the local optimization was used regardless of the optimization method (*i.e.*, GA or SA). Moreover, when the local optimization is combined with the GA, the solution returned by comb-GA-wl for the examples considered here and in [Carretero, 2002] was always the global minimum, regardless of the initial point used in the simulation. Further robustness tests can be found in [Carretero, 2002] where most tests were performed up to 2000 times per pose and per geometry.

[6]Due to the complexity of the geometries in this example, the minimum distance was only approximated. The reported distance is believed to be accurate within ± 0.05 units.

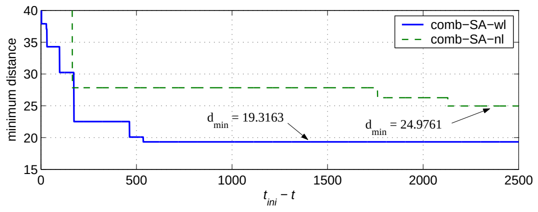

Fig. 14. Time history of the minimum distance of the trial point for comb-SA-wl and comb-SA-nl (handshake example).

According to the data given in Tables III, V and VI for the algorithms using local optimization and assuming the computations were performed at a rate of 1414 Mflops/s (PC at 3.06 GHz [Dongarra, 2003]), the minimum distance calculation would take under 1.5 ms for comb-GA-wl on the complicated scenario whereas comb-SA-wl would take up to 4.5 to 5 times more for the same test case. The main reason for this large difference in computational expense between the GA and SA implementation is due to the fact that all trial points in the SA are generated semi-randomly thus the local optimization algorithm needs to take many more steps to locally optimize each trial point. The GA implementation, on the other hand, will generate most trial points by using the mating process which will tend to produce offspring in potentially good regions of the search space allowing the local optimization algorithm to converge in a few steps.

Also worthwhile of noting is that a careful analysis of the computational expense taken by each of part of the minimum distance algorithm revealed that for the GA implementation around 80% of the computations were due to the local optimization algorithm. On the other hand, for the SA implementation the percentage was substantially higher reaching in many instances close to 98% (more details available in [Carretero, 2002]).

The results given in Tables III, V and VI are fairly similar in terms of the number of flops required to execute the program. This is mainly due to the fact that all SA and GA parameters were kept identical between the three examples. That is, the main factor that changes in the problem is the total number of nodes which varies between 3 083 and 6 091. As a matter of fact, it is the total number of nodes on the surface meshes the main factor that determines the amount of computations for the algorithm to converge. This, of course, does not come as a surprise as the current problem is a combinatorial one

V. Conclusions

A new method to solve the minimum distance problem between concave objects was presented. The method involves the use of global optimization techniques such as Genetic Algorithms and Simulated Annealing and does not require the concave objects to be partitioned into convex sub-pieces. I addition of not having to introduce unnecessary fictitious surfaces resulting from the partition, the new method has the advantage of being able to solve the minimum distance problem in a single run.

It was shown that the computational efficiency of this method is greatly improved by substituting the continuous geometry by a number of points on the surface of the body. As a result, the optimization process is converted to a combinatorial process where the objective is to find the pair of points, one on the surface of each body, that minimizes the Euclidean distance between them.

Furthermore, if the points are distributed on a mesh, the convergence process is greatly improved by using a local optimization (based on a hill-climbing strategy) inside the global optimization method. Such method uses the distance and predecessor matrices obtained off-line only once or every object.

Through a series of numerical tests, some of them reported in this paper, it was demonstrated that the GA implementation of the minimum distance algorithm outperforms, in terms of computational efficiency and robustness, the SA implementation. Additionally, the possibility of parallelizing the code makes the GA implementation more attractive for a real time application.

Currently, the authors are working on improving the accuracy of the results which is now a function of the mesh size and the distribution of its nodes. One of the proposed solutions is to use the global optimization method presented in this paper as a pre-processing algorithm at each iteration. The solution point of the global search, which locates the region where the exact minimum occurs, would then be passed to a local minimum distance method which will find the exact solution, within machine precision, of the minimum distance problem.

Nonetheless, the proposed algorithm could be used, as is, in most path planning and virtual reality applications where only an approximate solution of the minimum distance problem is required. Note that, although the proposed method would give an approximation, its solution would still be much more precise that a bounding box approach unless a very coarse mesh is generated on the surface of the objects.

Acknowledgment

The authors would like to acknowledge financial support received from NSERC, MD Robotics and the Canadian Space Agency in the development of this work through an NSERC/CSA Partnership Agreement grant.

References

[Aarts and Korst, 1989] Aarts, E. and Korst, J. (1989). *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing.* Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley and Sons, Chichester.

[Abdel-Malek and Burton, 1994] Abdel-Malek, K. A. and Burton, P. (1994). Interference detection of non-convex solids for manipulators. In *Proceedings of ASME Conference on Advances in Design Automation, Flexible Assembly Systems*, volume DE-Vol. 73, pages 85–95, Minneapolis, Minnesota, USA.

[Baker, 1987] Baker, J. E. (1987). Adaptive selection methods for genetic algorithms. In Grefenstette, J. J., editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 100–111, MIT, Cambridge, MA. Lawrence Erlbaum Associates, Publishers.

[Beasley et al., 1993a] Beasley, D., Bull, D. R., and Martin, R. R. (1993a). An overview of genetic algorithms: Part 1, fundamentals. *University Computing*, 15(2):58–69.

[Beasley et al., 1993b] Beasley, D., Bull, D. R., and Martin, R. R. (1993b). An overview of genetic algorithms: Part 2, research topics. *University Computing*, 15(4):170–181.

[Bobrow, 1989] Bobrow, J. E. (1989). A direct minimization approach for obtaining the distance between convex polyhedra. *International Journal of Robotics Research*, 8(3):65–76.

[Boyse, 1979] Boyse, J. W. (1979). Interference detection among solids and surfaces. *Communications of the ACM*, 22(1):3–9.

[Buchal et al., 1989] Buchal, R. O., Cherchas, D. B., Sassani, F., and Duncan, J. P. (1989). Simulated off-line programming of welding robots. *International Journal of Robotics Research*, 8(3):31–43.

[Carretero, 2002] Carretero, J. A. (2002). *Distance determination algorithms for convex and concave objects.* PhD Dissertation, Department of Mechanical Engineering, University of Victoria, Victoria, British Columbia, Canada.

[Carretero and Nahon, 2001] Carretero, J. A. and Nahon, M. A. (2001). A genetic algorithm for calculating minimum distance between convex and concave bodies. In *Proceedings of The 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space: A New Space Odyssey*, Montréal, Québec, Canada.

[Carretero et al., 2001] Carretero, J. A., Nahon, M. A., and Ma, O. (2001). Solving the minimum distance problem between convex or concave bodies using simulated annealing. In *Proceedings of the 2001 IEEE International Conference on Intelligent Robots and Systems*, Maui, Hawaii, USA.

[Carretero et al., 2000] Carretero, J. A., Podhorodeski, R. P., Nahon, M. A., and Gosselin, C. M. (2000). Kinematic analysis and optimization of a new three degree-of-freedom spatial parallel manipulator. *Journal of Mechanical Design*, 122(1):17–24.

[CIMNE, 2003] CIMNE (2003). *GiD - The personal pre and post-processor, http://gid.cimne.upc.es/.* International Center for Numerical Methods in Engineering, Barcelona, Spain.

[Davis and Steenstrup, 1987] Davis, L. and Steenstrup, M. (1987). Genetic algorithms and simulated annealing: An overview. In Davis, L., editor, *Genetic Algorithms and Simulated Annealing*, Research Notes in Artificial Intelligence, pages 1–11, London. Pitman Publishing.

[Dijkstra, 1959] Dijkstra, E. W. (1959). A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271.

[Dongarra, 2003] Dongarra, J. J. (2003). Performance of various computers using standard linear equations software. Technical Report CS-89-85, Computer Science Department, University of Tennessee.

[Ehmann and Lin, 2001] Ehmann, S. A. and Lin, M. C. (2001). Accurate and fast proximity queries between polyhedra using surface decomposition. In *Proc. of Eurographics (Computer Graphics Forum)*, volume 20 (3), page 11, Manchester, UK.

[Gilbert et al., 1988] Gilbert, E. G., Johnson, D. W., and Keerthi, S. S. (1988). A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation*, 4(2):193–203.

[Goldberg, 1989] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison Wesley.

[Gottschalk et al., 1996] Gottschalk, S., Lin, M., and Manocha, D. (1996). OBB-tree: A hierarchical structure for rapid interference detection. *Computer Graphics, Annual Conference Series (SIGGRAPH '96 Proceedings)*, 30:171–180.

[Grefenstette, 1987] Grefenstette, J. J. (1987). *Genetic Algorithms and their Applications (ICGA'87).* Lawrence Erlbaum Associates, Publishers. Editor.

[Holland, 1975] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor.

[Joe, 1986] Joe, B. (1986). Triangular meshes for regions of complicated shape. *International Journal for Numerical Methods in Engineering*, 23:751–778. http://members.attcanada.ca/˜bjoe/index.htm.

[Kirkpatrick et al., 1983] Kirkpatrick, S., Gelatt Jr., C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 4598(220):671–680.

[Lin and Canny, 1991] Lin, M. C. and Canny, J. F. (1991). A fast algorithm for incremental distance determination. In *Proceedings of the 1991 IEEE Conference on Robotics and Automation*, pages 1008–1014, Sacramento, California, USA.

[Liu and Mayne, 1990] Liu, C. Y. and Mayne, R. W. (1990). Distance calculations in motion planning problems with interference situations. In *Proceedings of the 1990 ASME DETC – 16th Design Automation Conference*, pages 145–152, Chicago, Illinois.

[Llanas et al., 2003] Llanas, B., Fernández de Sevilla, M., and Feliú, V. (2003). Minimum distance between the faces of two convex polyhedra: A sufficient condition. *Journal of Global Optimization*, 26:361–385.

[Ma and Nahon, 1992] Ma, O. and Nahon, M. (1992). A general method for computing the distance between two moving objects using optimization techniques. In *Proceedings of ASME Conference on Advances in Design and Automation*, volume 1, pages 109–117, Phoenix, Arizona, USA.

[Martinez-Alfaro et al., 1998] Martinez-Alfaro, H., Valdez, H., and Ortega, J. (1998). Linkage synthesis of a four bar mechanism for n precision points using simulated annealing. In *Proceedings of 1998 ASME DETC Conference*, page 7, Atlanta, Georgia.

[Maruyama, 1972] Maruyama, K. A. (1972). A procedure to determine intersection between polyhedral objects. *International Journal of Computer and Information Sciences*, 1(3):255–266.

[Meyer, 1986] Meyer, W. (1986). Distance between boxes: Applications to collision detection and clipping. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 597–602, San Francisco, California, USA.

[Nahon, 1993] Nahon, M. (1993). Determination of the minimum distance between moving objects including velocity information. In *Proceedings of ASME Conference on Advances in Design Automation*, volume 1, pages 693–698.

[Nahon et al., 1998] Nahon, M., Ma, O., Piedbœuf, J.-C., and DeCarufel, J. (1998). A distance determination algorithm for real-time simulation of contact dynamics. In *Proceedings of the 7th CASI Conference on Astronautics*, volume 1, pages 1–7, Ottawa, ON, Canada.

[Ponamgi et al., 1997] Ponamgi, M. K., Manocha, D., and Lin, M. C. (1997). Incremental algorithms for collision detection between polygonal models. *IEEE Transactions on Visualization and Computer Graphics*, 3(1):51–64.

[Rutenbar, 1989] Rutenbar, R. A. (1989). Simulated annealing algorithms: An overview. *IEEE Circuits and Devices Magazine*, 5(1):19–26.

[Tanenbaum, 1996] Tanenbaum, A. S. (1996). *Computer Networks.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 3rd edition.

[Whitley et al., 1994] Whitley, D., Gordon, V. S., and Mathias, K. (1994). Lamarckian evolution, the Baldwin effect and function optimization. In *Parallel Problem Solving from Nature, PPSN III: International Conference on Evolutionary Computation*, pages 6–15, Jerusalem, Israel. Springer-Verlag.

Juan A. Carretero received his B. Eng. degree in electromechanical engineering from the Universidad Nacional Autónoma de México (UNAM), graduating with hounours in 1996. He received his M.A.Sc. and Ph.D. degrees in mechanical engineering at the University of Victoria (Canada) in 1998 and 2002, respectively. Dr. Carretero is currently an assistant professor at the Department of Mechanical Engineering at the University of New Brunswick (Canada). His research interests include the design and optimization of parallel mechanisms for applications in machining and low-cost flight simulation. Also, he is developing advanced distance determination algorithms for complex concave environments based on the use of Genetic Algorithms.

Meyer Nahon is an Associate Professor in the Department of Mechanical Engineering at McGill University. His present research deals with various aspects of robotics including: dynamics and control of aerial and undersea vehicles; mechanics of parallel mechanisms; and distance determination algorithms. Dr. Nahon has acted as consultant to a number of companies, including CAE Electronics, MD Robotics, Sikorsky Aircraft and Evans and Sutherland Ltd. He has received awards from the American Institute of Aeronautics and Astronautics and from the Canadian Aeronautics and Space Institute for his work on flight simulator motion systems and on space-based robotics.