Proceedings of the 2004 IEEE
International Conference on Robotics & Automation
New Orleans, LA • April 2004

Solving Geometric Constraints by Iterative Projections and Backprojections

Federico Thomas
Institut de Robòtica i Informàtica Industrial (CSIC-UPC)
Llorens Artigas 4-6, 2 planta, 08028 Barcelona, Spain
Email: fthomas@iri.upc.es

Abstract— **Most geometric constraint problems can be reduced to give coordinates to a set of points from a subset of their pairwise distances. By exploiting this fact, this paper presents an algorithm that solves geometric constraint systems by iteratively reducing and expanding the dimension of the problem. In general, these projection/backprojection iterations permit tightening the ranges for the possible solutions but, if at a given point no progress is made, the algorithm bisects the search space and proceeds recursively for both subproblems. This branch-and-prune strategy is shown to converge to all solutions.**

I. Introduction

The resolution of systems of geometric constraints has aroused interest in many areas of Robotics (contact formation between polyhedra, assembly planning, forward/inverse kinematics of parallel/serial manipulators, path planning of closed-loop kinematic chains, etc.) and CAD/CAM (constraint-based sketching and design, interactive placement of objects, etc.). The solution of such problems entails finding object positions and orientations that satisfy all established constraints simultaneously. Reviews of the methods proposed to solve this problem in the context of Robotics, CAD/CAM and Molecular Conformation can be found in [14], [10], and [8], respectively.

Most of the proposed methods consist in translating the original geometric problem into a system of algebraic equations. In this paper, we depart from this usual formulation in that our algorithm does not rely on an algebrization of the problem. Contrarily, all operations have a direct interpretation in terms of geometric transformations in the embedding space of the problem and variables are *distances* instead of degrees of freedom linked to artificial reference frames.

Most direct and inverse kinematics problems can be expressed in terms of systems of distance constraints between points. Consider, for example, the problem of finding all valid configurations of a closed 6R linkage, a cycle of six binary links pairwise articulated with revolute joints Fig. 1a. A binary link can be modelled by taking two points on each of its two revolute axes and connecting them all with rigid bars to form a tetrahedron. By doing so, a 6R linkage is easily translated into a ring of six tetrahedra, pairwise articulated through a common edge (Fig. 1b) which can be simply regarded as a set of points that keep some pairwise distances. Observe that the valid configurations of this ring are in one-to-one correspondence with those of the original 6R linkage.

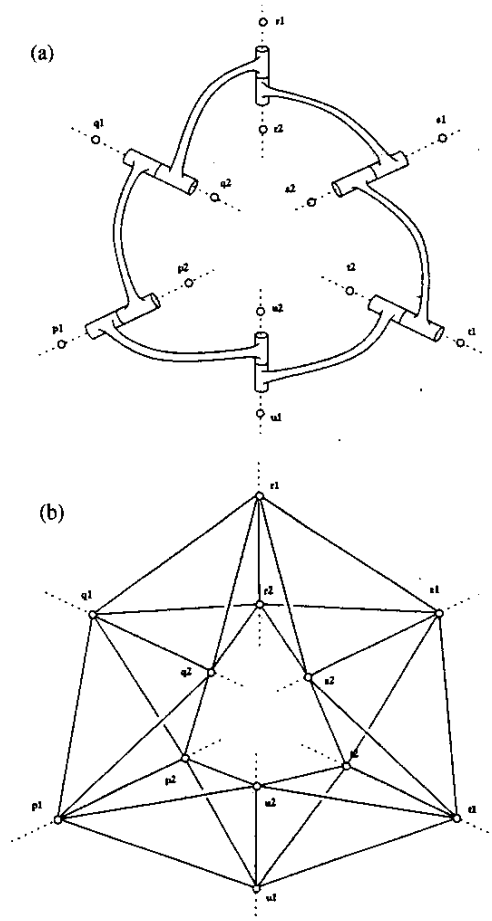

Fig. 1. A general 6R linkage (a) and its translation into a tetrahedral ring (b).

The total number of pairwise distances between n points are $\frac{n(n-1)}{2}$. The above example involves 12 points and 66 distances from which 30 are known. Likewise, it can be checked that the translation of the forward kinematics the general Gough-Stewart platform into distance constraints between points also

involves 12 points and 66 distances. In this case 36 of these distances are known. Therefore, if a computer program, able to obtain all sets of values for the unknown distances compatible with the known ones, would be available, obtaining all possible solutions of the inverse kinematics of a 6R robot or the forwards kinematics of a Gough-Stewart platform would become trivial.

Our problem can be more formally stated as follows: Given a graph $G = (V_n, E)$, with node set $V_n = \{1, \ldots, n\}$ and edge set E, and a real partial symmetric matrix $\mathbf{A} = (a_{ij})$ with pattern G and with zero diagonal entries, determine whether $\mathbf{A}$ can be completed to a *Euclidean distance matrix* [12]; that is, whether there exist vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \Re^d$ for a given $d \geq 1$ such that

$$a_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2 \quad \text{for all} \quad ij \in E,$$

where $\|\mathbf{x}\|$ denotes the Euclidean norm of $\mathbf{x} \in \Re^d$. The vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are then said to form a *realization* of $\mathbf{A}$. Unfortunately, this problem has been shown to be NP-complete if $d = 1$ and NP-hard if $d \geq 2$ [18]. Thus, the best we can expect is to come up with an algorithm that, despite its unavoidable exponential complexity, works well for problems of reasonable size.

Euclidean distance matrices are a central notion in the area of Distance Geometry [4]. Their study was initiated by Cayley in 1841 [5], but they were not systematically studied until 1928, when K. Menger showed how they could be used to study convexity and other basic geometric problems [13]. Nowadays, in dimension 3, these matrices play a fundamental role in the study of molecular conformations in chemistry [6].

Given an arbitrary symmetric matrix with 0 entries in its diagonal and strictly positive entries in the rest, two different criteria have been found to decide whether it corresponds to a Euclidean distance matrix. One derives from the theory of Cayley-Menger determinants and the other from the theory of positive semidefinite matrices. These two criteria lie at the innermost of the two main families of algorithms that have been proposed for solving the problem at hand. We will briefly comment on them in the next section. Nevertheless, it is worth saying here that all these algorithms lead to a rapid algebrization of the problem while the one proposed in this paper is based on elementary geometric operations, thus keeping the geometric flavor of the problem. Despite its geometric nature, it is closely related to the algorithm tools developed for positive semidefinitive matrices. This connection permits proving the correctness of the proposed algorithm.

This paper is structured as follows. Section II prepares the ground for the sections that follow. To keep it short we give the needed material in intuitive terms without going into mathematical details. Section III describes the two basic geometric operations in which our algorithm is based; namely, projection and backprojection. Section IV presents the algorithm and Section V gives simple examples to clarify the main points. Finally, section VI contains the conclusions and points that deserve further research.

II. Preliminaries

A. Incomplete distances matrices as interval matrices

The pairwise distances between n points, say $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$, will be arranged in a symmetric matrix of the form $\mathbf{D}^0 = (d_{ij}^2)$, $i, j = 1, \ldots, n$, where d_{ij} is the Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$. In our case, not all elements of $\mathbf{D}^0$ are known but lower and upper bounds on them can readily be obtained. For example, all unknown distances are necessarily smaller than the sum of all known distances, say σ. Then, all unknown distances can be bound to lie in the interval $[0, \sigma]$. But this trivial bound can be further tightened. In the literature this process is referred to as *bound-smoothing*. In this process, given the upper and lower bounds on a subset of pairwise distances, triangle and/or tetrangle inequalities are used to obtain the bounds to further tighten the existing ones [6, pp. 221-285].

Since any three points in the Euclidean space have to satisfy the triangle inequality, bounds could be tightened by applying this inequality to three points at a time [7]. The distance bounds thus obtained can still be tightened further by applying the tetrangle inequality —the limits imposed on the six pairwise distances among a set of four points (instead of three for the triangle inequalities) [17].

As a consequence, in what follows, it is convenient to assume that all entries in vectors and matrices are real compact intervals, and that all ordinary arithmetic operations on them are carried out according to the standard interval arithmetics [1]. That is, intervals are added, subtracted, multiplied, etc. in such a way that each computed interval is generated to contain the unknown value of the quantity it represents.

Although it introduces a slight abuse of language, vectors and matrices will be treated as sets which, under certain circumstances, could be operated as such. For example, two matrices of the same size can be intersected provided that the result is also a matrix with real compact intervals.

B. Necessary and sufficient conditions of realizability

The elements of $\mathbf{D}^0$ have to satisfy certain algebraic conditions to be the set of pairwise squared Euclidean distances between n points in $\Re^d$. These conditions are equalities, due to Cayley, and inequalities —such as the triangular and tetrangle inequalities— due to Menger. Subsets of these equalities and inequalities can be chosen to complete sets of necessary and sufficient conditions for $\mathbf{D}^0$ be an Euclidean distance matrix [6]. Unfortunately, this characterization of realizability does not seem useful for our purposes because it

leads to a rapid algebrization of the problem and we are interested on obtaining an algorithm based on purely geometric constructions. For more details on an algorithm for solving the problem at hand based on this characterization of realizability, the reader is addressed to [16].

To find an alternative characterization of the realizability, let us organize, without loss of generality, the coordinates of $\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}$ with reference to $\mathbf{x}_n$ in the following $(n-1) \times d$ matrix:

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 - \mathbf{x}_n \\ \mathbf{x}_2 - \mathbf{x}_n \\ \vdots \\ \mathbf{x}_{n-1} - \mathbf{x}_n \end{pmatrix},$$

where row i contains the coordinates of the vector pointing from $\mathbf{x}_n$ to $\mathbf{x}_i$. Then, the element (i,j) of the $(n-1)\times(n-1)$ matrix $\mathbf{X}\mathbf{X}^t$ contains the inner product $\langle \mathbf{x}_i - \mathbf{x}_n, \mathbf{x}_j - \mathbf{x}_n \rangle$. Actually, $\mathbf{G} = \mathbf{X}\mathbf{X}^t$ is known as a *Gram matrix*, a positive semidefinite matrix whose rank is equal to the dimension of the space in which the n points are embedded (in our case d).

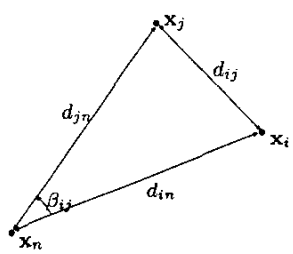

Fig. 2. The triangle formed by $\mathbf{x}_i$, $\mathbf{x}_j$ and $\mathbf{x}_n$.

According to Fig. 2, and using the cosine theorem, we can directly obtain the elements of $\mathbf{G}$ from those of $\mathbf{D}^0$ as follows:

$$\begin{aligned} \mathbf{G}[i,j] &= \langle \mathbf{x}_i - \mathbf{x}_n, \mathbf{x}_j - \mathbf{x}_n \rangle \\ &= d_{in} d_{jn} \cos\beta_{ij} = \frac{1}{2}(d_{in}^2 + d_{jn}^2 - d_{ij}^2). \end{aligned} \tag{1}$$

Schoenberg showed that $\mathbf{D}^0$ is a proper Euclidean matrix if, and only if, $\mathbf{G}$, obtained using (1), is positive semidefinite and its rank gives the dimension of the space in which the points can be embedded [19]. This characterization of realizability has led to several algorithms for solving the matrix completion problem by semidefinite programming and variations [3], [12], [15]. The problem with these algorithms is that they are only able to find a realization within the provided ranges for the distances, while we are actually interested in all possible realizations.

What is important for us is that the Schoenberg's characterization of realizability permits concluding that the Gram matrix $\mathbf{G}$ can be uniquely factorized into the product $\mathbf{L}\mathbf{L}^t$, where $\mathbf{L}$ is an $(n-1)\times d$ lower triangular matrix because it is positive semidefinite of rank d. This factorization can be obtained by the application of d steps of the Cholesky factorization algorithm (see [20, p. 188] for a standard presentation and [2] for the analysis of the interval version of this algorithm). Then, the rows of $\mathbf{L}$ can be directly seen as the coordinates of $\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}$ and therefore it is equivalent to $\mathbf{X}$ up to rotations. This algebraic fact has a nice geometric interpretation in which the algorithm given below is based but, before we explain it, let us introduce the two basic operation in which it is based.

III. The Two Basic Operations

A. Projection

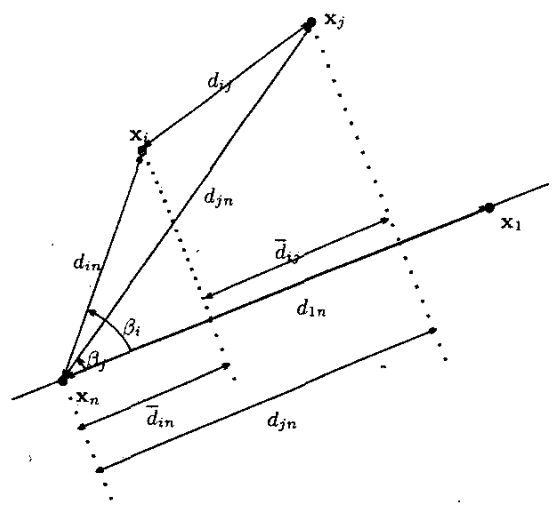

Fig. 3. Projecting distances onto the axis defined by $\mathbf{x}_n\mathbf{x}_1$.

Let us take as a reference the coordinate axis defined by $\mathbf{x}_n\mathbf{x}_1$ with origin at $\mathbf{x}_n$ and pointing to $\mathbf{x}_1$ in the positive direction. Then, according to Fig. 3, the projection $\bar{d}_{in}$ of d_{in} on this axis is, using the cosine theorem,

$$\bar{d}_{in} = d_{in} \cos\beta_i = \frac{1}{2d_{1n}}(d_{in}^2 + d_{1n}^2 - d_{i1}^2). \tag{2}$$

Hence,

$$\begin{aligned} \bar{d}_{ij} &= \bar{d}_{in} - \bar{d}_{jn} \\ &= \frac{1}{2d_{1n}}(d_{in}^2 - d_{jn}^2 + d_{j1}^2 - d_{i1}^2), \end{aligned} \tag{3}$$

and the orthogonal component $d_{ij}^{\perp}$ of d_{ij} is

$$\begin{aligned} (d_{ij}^{\perp})^2 &= d_{ij}^2 - \bar{d}_{ij}^2 \\ &= d_{ij}^2 - \frac{1}{4d_{1n}^2}(d_{in}^2 - d_{jn}^2 + d_{j1}^2 - d_{i1}^2)^2 \end{aligned} \tag{4}$$

Then, we can construct the vector

$$\mathbf{v}^1[i] = \bar{d}_{in}, \quad i = 1, \ldots, n \tag{5}$$

that contains the projections of d_{in} on the axis defined by $\mathbf{x}_n\mathbf{x}_1$, and the matrix

$$\mathbf{D}^1[i,j] = (d_{i,j}^{\perp})^2, \quad i,j = 1, \cdots, n \tag{6}$$

which is a distance matrix containing all squared distances involved in $\mathbf{D}^0$ projected onto the hyperplane orthogonal to the axis defined by $\mathbf{x}_n\mathbf{x}_1$. Note that $\mathbf{D}^1[1,n] = 0$, and, as a consequence, $\mathbf{D}^1[i,1] = \mathbf{D}^1[i,n]$, $i = 1,\ldots,n$, so that we can define $\widetilde{\mathbf{D}}^1[i,j] = \mathbf{D}^1[i,j]$, $i,j = 1,\cdots,n-1$, from which $\mathbf{D}^1$ can be straightforwardly recovered. Then, deciding whether n points can be embedded in $\Re^d$ from their interpoint distances boils down to decide whether $n-1$ points can be embedded in $\Re^{d-1}$ from a new set of interpoint distances obtained by projection. The above projection process can be iteratively repeated so that $\widetilde{\mathbf{D}}^1$ yields $\mathbf{D}^2$ and $\mathbf{v}^2$ and so on. Note that, after d iterations, if our points can be embedded in $\Re^d$, $\widetilde{\mathbf{D}}^d$ must be identically null. Space limitations prevents us from showing that, if $\mathbf{D}^0$ is a point matrix, this is a necessary and sufficient condition of realizability because $\mathbf{v}^1,\ldots,\mathbf{v}^d$ provide a Cholesky factorization of the Gram matrix associated with $\mathbf{D}^0$.

Remind that all distances should be treated as intervals and, as a consequence, all expressions above should be evaluated using interval arithmetics. Therefore, the following facts must be taken into account:

1) The interval resulting from evaluating (4) can contain negative values. Since they do not correspond to real solutions, they can be ruled out from the result. If the resulting interval is empty, the matrix being projected is not a proper Euclidean matrix.
2) If the interval for the denominator in (4) contains the origin, the projection cannot be performed. Another projection should be chosen to reduce the dimension of the problem.
3) The direct interval evaluation of all projected distances does not necessarily lead to a interval symmetric matrix. Then, projected matrices must be regularized: their diagonal entries must be set to zero and all other entries substituted by their ranges intersected with their symmetric ones.

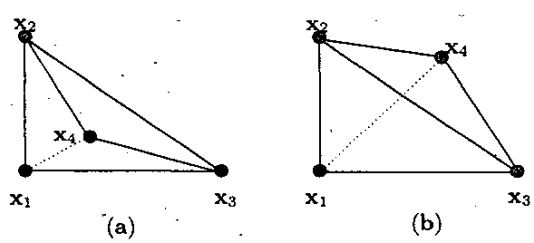

Fig. 4. If there is only an unknown distance between four points in $\Re^2$ –represented here by a dotted segment–, two possible realizations arise, up to congruences.

As an example, let us consider four points in $\Re^2$ so that all distances between them but one are known. In general, this leads to two possible realizations, as shown in Fig. 4, up to congruences.

If the corresponding distance matrix is

$$\mathbf{D}^0 = \begin{pmatrix} 0 & 16 & 36 & d_{14}^2 \\ 16 & 0 & 52 & 13 \\ 36 & 52 & 0 & 17 \\ d_{14}^2 & 13 & 17 & 0 \end{pmatrix},$$

the two realizations correspond to the values for d_{12}^2 of 5 and 23.461.

Let us suppose that $d_{14}^2 = [4.9, 5.1]$. This interval contains the realization in Fig.4a. The application of the first and second projections lead to:

$$\widetilde{\mathbf{D}}^1 = \begin{pmatrix} [0,0] & [12.77,12.82] & [6.86,7.53] \\ [12.77,12.82] & [0,0] & [38.94,39.45] \\ [6.86,7.53] & [38.94,39.45] & [0,0] \end{pmatrix},$$

and

$$\widetilde{\mathbf{D}}^2 = \begin{pmatrix} [0,0] & [0,1.35] \\ [0,1.35] & [0,0] \end{pmatrix},$$

respectively. The last projection contains the null matrix. In other words, $\mathbf{D}^0$ might contain a realization in $\Re^2$ within the analyzed interval.

Now, let us suppose that $d_{14}^2 = [5.1, 5.2]$ where no realization exists. This is readily detected because, after the same two projections, one gets

$$\widetilde{\mathbf{D}}^2 = \begin{pmatrix} [0,0] & [0.26,1.50] \\ [0.26,1.5] & [0,0] \end{pmatrix}$$

which does not contain the null matrix.

Finally, let us assume that $d_{14}^2 = [0.0001, 21.0]$. In this case, the second projection can no be performed because the denominator in expression (4) includes the origin. To proceed, we can change the projecting axis by permuting rows and columns in the input distance matrix.

B. Backprojection

It can be checked that

$$\widetilde{\mathbf{D}}^{k-1}[i,j] = \mathbf{D}^k[i,j] + (\mathbf{v}^k[i] - \mathbf{v}^k[j])^2,$$

for $i,j = 1,\ldots,n-1$. Therefore, it is clear that $\mathbf{D}^{d-1}$ can be recovered from $\mathbf{D}^d$ and $\mathbf{v}^d$. This backprojection operation can take as input $\mathbf{D}^d$ and repeated till $\mathbf{D}^0$ is recovered. Since $\mathbf{D}^d$ must be identically null for solutions embedded in $\Re^d$, one concludes that $\mathbf{v}^1,\ldots,\mathbf{v}^d$ encodes all the information required to recover $\mathbf{D}^0$. Actually, it can be shown that the rows of the $(n-1)\times d$ matrix

$$\begin{pmatrix} \mathbf{v}^1[1] & 0 & \cdots & 0 \\ \mathbf{v}^1[2] & \mathbf{v}^2[1] & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{v}^1[n-1] & \mathbf{v}^2[n-2] & \cdots & \mathbf{v}^d[n-d] \end{pmatrix}$$

provides a set of coordinates for $\mathbf{x}_1,\ldots,\mathbf{x}_{n-1}$, respectively, taking $\mathbf{x}_n$ at the origin.

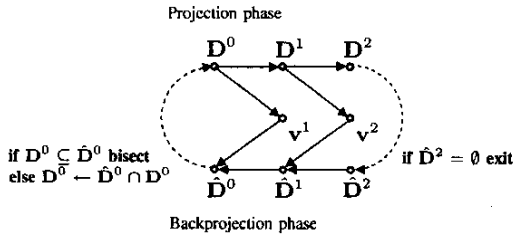

Fig. 5. The proposed algorithm iteratively projects and backprojects the input distance matrix thus tightening the bounds for the unknown distances. The process is repeated till it is proved that the input matrix cannot contain any realization or no relevant bound improvement is observed.

IV. The Algorithm

If we want to find all the realizations in $\Re^d$ contained in a given distance matrix $\mathbf{D}^0$, the proposed algorithm first projects it $d-1$ times yielding $\mathbf{D}^{d-1}$. As a byproduct of these projections, we get $d-1$ coordinate vectors that permit recovering $\mathbf{D}^0$ from $\mathbf{D}^{d-1}$.

All elements in $\mathbf{D}^{d-1}$ should correspond to pairwise distances between points on a line. When working in $\Re^1$, the triangle inequality becomes an equality. Then, we can tighten the involve interval distances by imposing this triangle equality to all subsets of three points. This can be done by adapting the distance smoothing algorithm proposed in [7]. The following algorithm, with $O(n^3)$ complexity, is thus obtained:

```
for j=1 to n
    for i=1 to n-1
        for k=i+1 to n
            d²_ik := hull ((d²_ik ∩ (d_ij − d_jk)²),
                           (d²_ik ∩ (d_ij + d_jk)²)) ;
        end
    end
end
```

where the hull function returns the smallest interval containing two given intervals.

Let $\hat{\mathbf{D}}^{d-1}$ denote the result of applying the above algorithm to $\mathbf{D}^{d-1}$. Now, $\hat{\mathbf{D}}^{d-1}$ can be backprojected, using the obtained coordinate vectors, yielding $\hat{\mathbf{D}}^0$. Then, if $\hat{\mathbf{D}}^0 \cap \mathbf{D}^0$ provides a reduction in the original bounds, this process can be iterated for all possible projections (by simply permuting indices) until either one of the entries in the obtained distance matrix gets empty, in which case we can conclude that our distance matrix contains no realization, or the matrix is "sufficiently" small, in which case it is considered a solution, or the matrix cannot be "significantly" reduced, in which case it is split into two matrices by bisection. If the latter case occurs, the whole process is repeated onto the newly created matrices, and to the matrices recursively created thereafter, until we end up with a collection of "small" matrices containing all solutions. Fig. 5 provides an schematic representation of this process for the three dimensional case.

The presented algorithm has been implemented in MATLAB using the INTLAB toolbox [9] that implements the standard interval arithmetics.

V. Examples

Let us apply the above algorithm so that, besides iterating for all possible projections, the same projection/backprojection iteration is repeated while a reduction in the range for at least one variable is higher than 5%. When, during these iterations, the ranges of all matrix entries go below 0.001, it is considered that a valid realization has been reached. Under this circumstances, for the example in Fig. 4 with $d_{14}^2 = [5.001, 23.461]$, the algorithm detects that no realization is contained in this interval in just 4 projection/backprojection iterations. If $d_{14}^2 = [0, 20]$, the algorithm converges to the realization in Fig. 4a ($d_{14}^2 = 5$) after 47 iterations without bisecting. Only 4 iterations do not contribute to any improvement in the ranges. If $d_{14}^2 = [0, 100]$, the two possible realizations are obtained after 64 iterations and a single bisection. In this case 15 iterations do not contribute to any improvement. It is important noticing that the number of iterations needed to obtain these solutions is highly variable with the chosen sequence of projections/backprojections. In this case, for example, the number of iterations drops to 12 by choosing a different ordering.

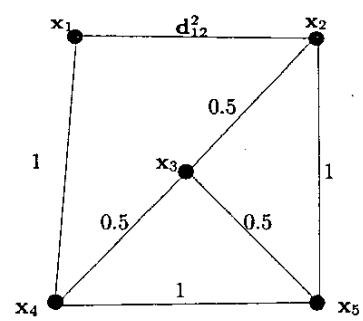

Fig. 6. Five points whose known pairwise squared distances are shown close to the corresponding edge.

As another example, let us consider the five point planar configuration shown in Fig. 6. This example is interesting for analyzing the behavior of the proposed algorithm near singularities. Observe that, for $0.1715 < d_{12}^2 < 1$, two possible realizations in $\Re^2$ satisfy the set of distance constraints. For $d_{12}^2 = 0.1715$, there is only one realization. For $d_{12}^2 = 1$, an infinite number of realizations arise because $\mathbf{x}_2$ and $\mathbf{x}_4$ can become coincident and $\mathbf{x}_1$ can then freely rotate around these two points. Setting $d_{12}^2 = 0.5$ and all unknown distances to the interval $[0, 100]$, two realizations are obtained, as expected, after 650 iterations and 7 bisections (357 iterations do not contribute to any improvement). Setting $d_{12}^2 = 0.999$, i.e. quite near to one of the two singularities, the two realizations are

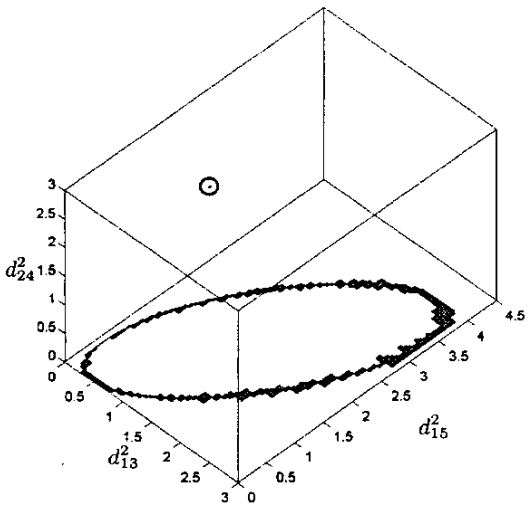

Fig. 7. The solution space for the example in Fig. 6 with $\mathbf{d}^2_{12} = 1$. Besides the encircled isolated realization at $d^2_{24} = 2$, a continuum arises at $d^2_{24} = 0$.

still obtained after 331 iterations and 10 bisections. In this case, 166 of these iterations do not provide any improvement. Setting $d^2_{12} = 0.1716$, i.e. near to the other singularity, the algorithm finds the two realizations after 379 projections and only 1 bisection. In this case, 165 iterations are useless. For $d^2_{12} = 1$, one isolated realization and a one-dimensional continuum of realizations arise as shown in Fig. 6 thus proving that the presented algorithm is immune to singularities and that it can also be used to discretized continuous sets of realizations.

VI. Conclusions

An algorithm for solving systems of pairwise distances constraints between points, based on two simple geometric operations, has been presented. For all analyzed problems, a single box is obtained for each isolated realization. In other words, the cluster effect has not been observed contrarily to what happened in [16]. This is due to the fact that the presented algorithm is based on a sequence of projection/backprojection operations, each of them being a necessary and sufficient condition of realizability, instead of applying a sequence of necessary conditions, one at a time. The speed of convergence to the solutions greatly depends on the chosen sequence of projections that are iteratively repeated. The development of heuristics for choosing this sequence is a point that deserves further efforts.

The algorithm can be easily parallelized on a cluster-like network of computers where each node work on a different sequence of distance projections and all nodes interchange the obtained bounds.

References

[1] G. Alefeld and J. Herzberger, *Introduction to Interval Computations, Academis Press*, New York, 1983.

[2] G. Alefeld and G. Mayer, "The Cholesky method for interval data," *Linear Algebra and its applications*, Vol. 194, pp. 161-182, 1993.

[3] A.Y. Alfakih, A. Khandani, and H. Wolkowicz, "Solving Euclidean distance matrix completion problems via Semideinite Programming," *Computational Optimization and Applications*, Vol. 12, No. 1-3, pp. 13-30, 1999.

[4] L.M. Blumenthal, *Theory and Applications of Distance Geometry*, Oxford University Press, 1953.

[5] A. Cayley, "A theorem in the Geometry of Position," *Cambridge Mathematical Journal*, Vol. II, pp. 267-271, 1841. (Also in *Collected Mathematical Papers of Arthur Cayley*, Cambridge University Press, 1963).

[6] G.M. Crippen and T. Havel, *Distance Geometry and Molecular Conformation*, John Wiley and Sons Inc., New York, 1988.

[7] A.W.M. Dress and T.F. Havel, "Shortest-path problems and molecular conformation," *Discrete Applied Mathematics*, Vol. 19, pp. 129-144, 1988.

[8] I.Z. Emiris and B. Mourrain, "Computer algebra methods for studying and computing molecular conformations," *Algorithmica*, No. 25, pp. 372-402, 1999.

[9] G.I. Hargreaves, "Interval Analysis in MATLAB," Numerical Analysis Report No. 416, Manchester Center for Computational Mathematics, 2002.

[10] C.H. Hoffmann and B. Yuan, "On spatial constraint solving approaches," in *Automated Deduction in Geometry: Third International Workshop*, LNCS No. 2061, Springer Verlag, 2000.

[11] M.K. Kim, G.S. Chirikjian, and R.L. Jernigan, "Elastic models of conformational transitions in macromolecules," *Journal of Molecular Graphics and Modelling*,

[12] M. Laurent, "A connection between positive semidefinite and Euclidean distance matrix completion problems," *Lineal Algebra and its Applications*, Vol. 273, No. 1-3, pp. 9-22, 1998.

[13] K. Menger, "New foundation for Euclidean geometry," *American Journal of Mathematics*, No. 53, pp. 721-745, 1931.

[14] J. Nielsen and B. Roth, "Formulation and solution for the direct and inverse kinematics problems for mechanisms and mechatronic systems," *Proc. of the NATO Advanced Institute on Computational Methods in Mechanisms*, J. Angeles and E. Zakhariev, Eds., Vol. I, pp. 233-252, 1997.

[15] T.G. Nikitopoulos and I.Z. Emiris, "Molecular conformation search by matrix perturbations," *preprint*.

[16] J.M. Porta, L. Ros, F. Thomas, and C. Torras, "A Branch-and-Prune Algorithm for Solving Systems of Distance Constraints," *IEEE Trans. on Robotics and Automation*, to appear.

[17] K. Rajan and N. Deo, "A parallel algorithm for bound-smoothing using tetrangle inequality," *Proc. of the IPPS/SPDP*, pp. 645-652, 1998.

[18] J.B. Saxe, "Embedadbility of weighted graphs in k-space is strongly NP-hard," *Proc. of the 17th Allerton Conference in Communications, Control and Computing*, pp 480-489, 1979.

[19] I.J. Schoenberg, "Remarks to a M. Fréchet's article," *Annals of Mathematics*, Vol. 36, pp. 724-732, 1935.

[20] G.W. Steward, *Matrix Algorithms. Volume I: Basic Decompositions*, SIAM, Philadelphia, 1998.