DISTANCE CONSTRAINTS SOLVED GEOMETRICALLY

F. Thomas, J.M. Porta, and L. Ros
Institut de Robòtica i Informàtica Industrial (CSIC-UPC)
Llorens i Artigas 4-6, 08028-Barcelona, Spain
{fthomas,jporta,llros}@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 distance 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.

Keywords: Distance constraints, distance geometry, geometric constraint systems.

1. 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. A review of the methods proposed to solve this problem in the context of Robotics can be found in (Nielsen and Roth, 1997).

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 of six binary links pairwise articulated with revolute joints (Fig.

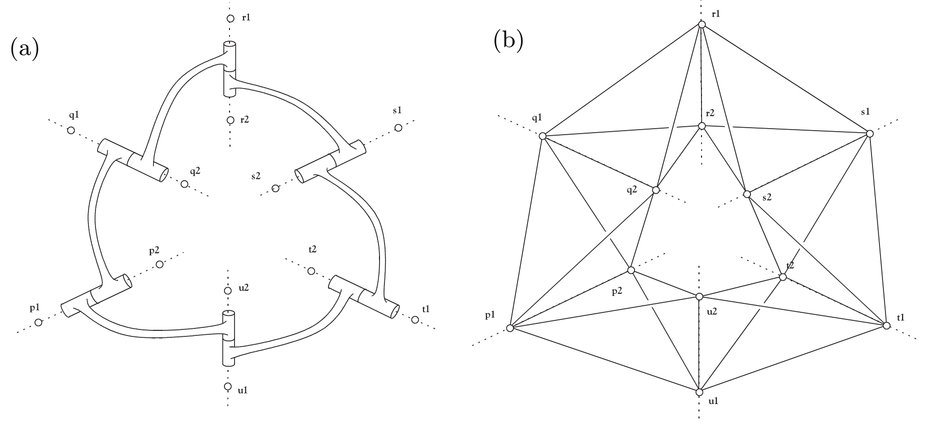

Figure 1. A general 6R linkage (a) and its translation into a tetrahedral ring (b).

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.

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 of the general Gough-Stewart platform into distance constraints between points also involves 12 points and 66 distances, 36 of which 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* (Laurent, 1998); that is, whether there exist vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \Re^d$ for a given $d \geq 1$ such that $a_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2$ for all $ij \in E$, 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

$d \geq 2$ (Saxe, 1979). 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.

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. 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, though, it is closely related to the algorithmic tools developed for positive semidefinitive matrices. This connection permits proving the correctness of the proposed algorithm.

This paper is structured as follows. Section 2 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 3 describes the two basic geometric operations in which our algorithm is based; namely, projection and backprojection. Section 4 presents the algorithm and examples to clarify the main points. Finally, section 5 contains the conclusions.

2. Preliminaries

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 $[0, \sigma^2]$. But this trivial bound can be further tightened. In the literature this process is referred to as *bound-smoothing*: given the upper and lower bounds on a subset of pairwise distances, triangle and/or tetrangle inequalities are used to further tighten these bounds (Crippen and Havel, 1988). 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. 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.

The elements of $\mathbf{D}^0$ have to satisfy certain algebraic conditions, derived from the theory of Cayley-Menger determinants, to be the set of pairwise squared Euclidean distances between n points in $\Re^d$ (Porta et al., 2004). Unfortunately, this characterization of realizability leads to a rapid algebrization of the problem and we are here interested on obtaining an algorithm based on purely geometric constructions. To find an alternative characterization of 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). Then, using the cosine theorem, we can directly obtain the elements of $\mathbf{G}$ from those of $\mathbf{D}^0$ as follows:

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

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 (Schoenberg, 1935). This characterization of realizability has led to several algorithms for solving the matrix completion problem by semidefinite programming and variations (Alfakih et al., 1999), (Laurent, 1998), (Nikitopoulos and Emiris, 2002). The problem with these algorithms is that they are only able to find *one* realization within the provided ranges for the distances, while we are actually interested in *all* possible realizations.

What is important for us is that 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 (Steward, 1998, p. 188). 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 on which the algorithm given below is based but, before we explain it, let us introduce the basic operations in which it relies.

3. Basic operations: projection and backprojection

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, the projection $\overline{d}_{in}$ of d_{in} on this axis is, using the cosine theorem,

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

Hence,

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

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

$$(d_{ij}^{\perp})^2 = d_{ij}^2 - \overline{d}_{ij}^2 = d_{ij}^2 - \frac{1}{4d_{1n}^2}(d_{in}^2 - d_{jn}^2 + d_{j1}^2 - d_{i1}^2)^2. \tag{4}$$

Then, we can construct the vector

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

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. After $d-1$ iterations, if our points can be embedded in $\Re^d$, $\widetilde{\mathbf{D}}^{d-1}$ can be embedded on a line. After d iterations, $\widetilde{\mathbf{D}}^d$ should be identically null. Space limitations prevent 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).

Now, 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 may be 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 $\mathbf{D}^0$.

4. The algorithm and examples

To find all realizations in $\Re^d$ contained in a given distance matrix $\mathbf{D}^0$, the proposed algorithm projects this matrix $d-1$ times yielding $\mathbf{D}^{d-1}$. As a byproduct of these projections, we get a set of coordinate vectors that permit recovering $\mathbf{D}^0$ from $\mathbf{D}^{d-1}$. Then, all elements in $\mathbf{D}^{d-1}$ that cannot be embedded on a line can be ruled out because they correspond to realizations that cannot be embedded in $\Re^d$. Thus, $\mathbf{D}^{d-1}$ is pruned accordingly and 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 permuting the indices of $\mathbf{D}^0$ until either (1) one of the entries in the obtained distance matrix gets empty, in which case we can conclude that our distance matrix contain no realization, or (2) the matrix is "sufficiently" small, in which case it is considered a solution, or (3) the matrix cannot be "significantly" reduced, in which case it is split into two matrices by bisection. If the latter 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.

The presented algorithm has been implemented in MATLAB using the INTLAB toolbox (Hargreaves, 2002) that implements the standard interval arithmetics. As an example, let us consider four points in $\Re^2$ so that all but one distances between them are known. In general, this leads to two possible realizations, as shown in Fig. 2, up to congruences.

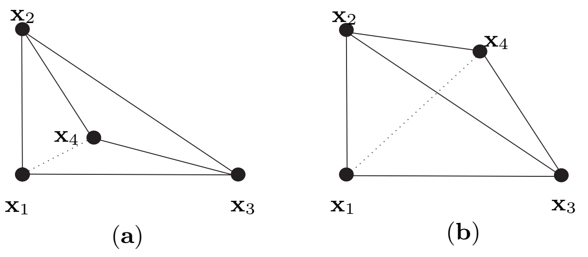

Figure 2. If there is only an unknown distance between four points in $\Re^2$ – represented here by a dotted segment–, two possible realizations arise, up to congruences.

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_{14}^2 of 5 and 23.461.

Let us suppose that $d_{14}^2 = [4.9, 5.1]$. This interval contains the realization in Fig.2a. The application of two consecutive projections leads 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. Nevertheless, note that after a single projection, the result can be efficiently pruned by embedding it on a line. Then, let us assume that $d_{14}^2 = [0, 100]$. By iterating for all possible projections, and after a single bisection and 64 projections, our algorithm is able to find the two solutions with an error bounded to 0.001. The number of projections is highly dependant on the chosen ordering for all possible projections. In this example, this number drops to 12 by properly choosing this ordering.

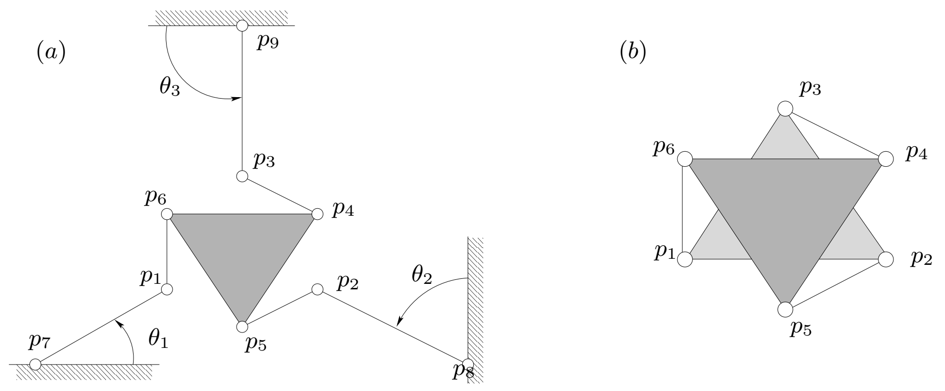

Figure 3. The forward analysis of the 3RRR planar manipulator (a) reduces to that of a 3RPR device (b).

As another example, let us consider the forward kinematics of the 3RRR planar manipulator in Fig. 3a which can be straightforwardly reduced to that of the 3RPR planar device shown in Fig. 3b. Using our formulation, this problem can be translated into 9 pairwise distance constraints between 6 points, and the solution entails finding the 6 unknown distances. For generic dimensions, it can have up to 8 different discrete solutions. An eight-degree polynomial to compute them all is derived in (Tsai, 1999, Section 3.4), and a numerical method of solution can be found in (Gosselin and Sefrioui, 1991). For particular dimensions, a one-dimensional solution set can arise for which standard numerical approaches may fail to provide a proper description. For example, if the distances between the three points forming the two triangles in Fig. 3b are 1 and the three leg lengths are set to $\sqrt{3/2}$, one gets the solution space shown in Fig. 4 using the presented algorithm. It has 6 discrete solutions, contained in 6 isolated boxes, and a one-dimensional solution set, obtained here as a discretization of 538 boxes. Since the shown solution space is actually a projection of a solution space embedded in $\Re^6$, these boxes overlap at some points. This space has been obtained after 50054 projections and 1108 bisections for a maximum accepted error of 0.05.

5. Conclusions

An algorithm for solving systems of pairwise distance constraints between points, based on two simple geometric operations, has been presented. The cluster effect near singularities has not been observed, contrarily to what happened in (Porta et al., 2004), because this algorithm is

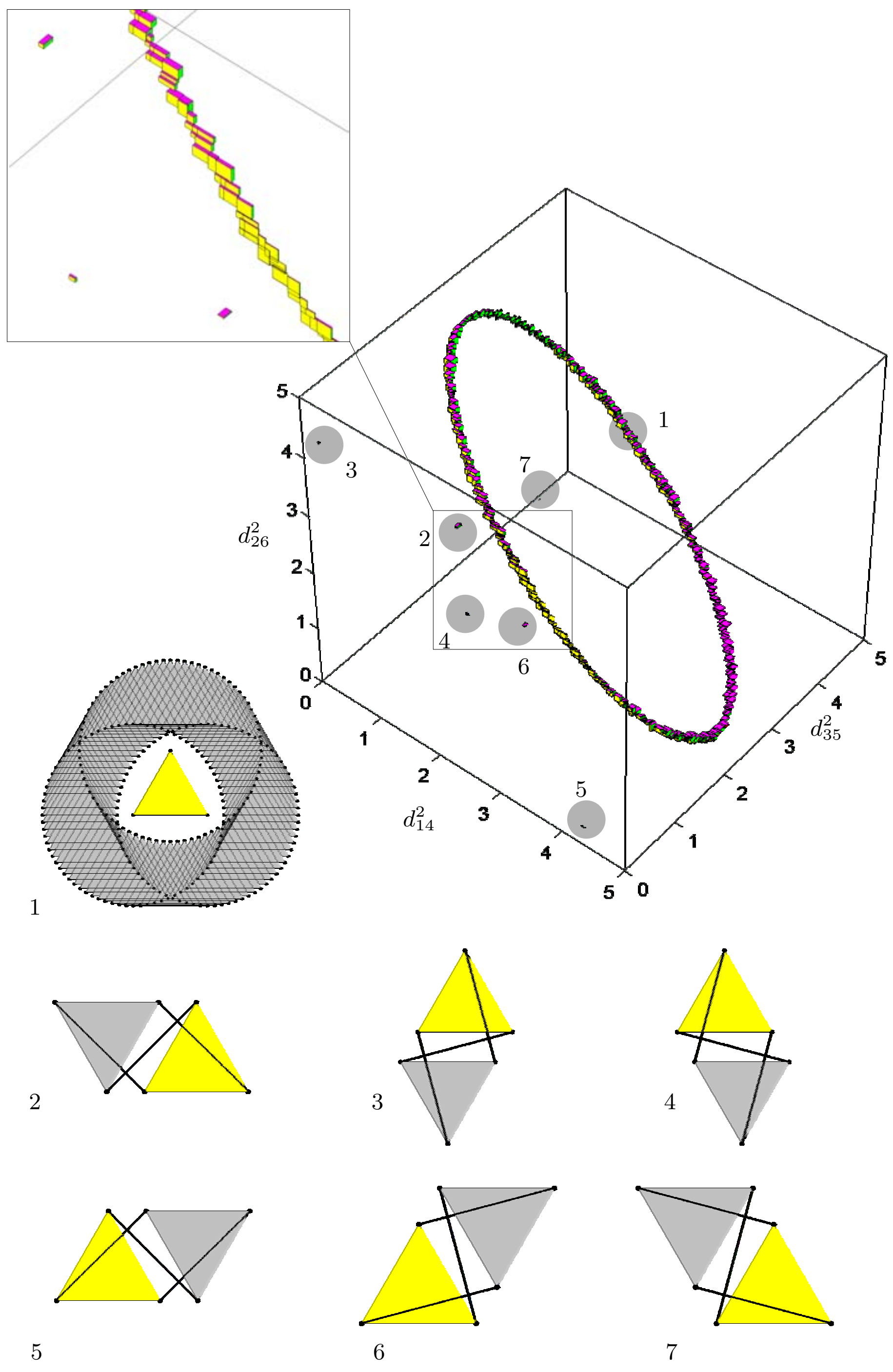

Figure 4. Feasible configurations of the 3RPR planar manipulator. Each configuration corresponds to the same-numbered box in the plot above. See the text for details.

based on the application of a sequence of projection/backprojection operations, each of them derived from a necessary and sufficient condition of realizability. The speed of convergence to the solutions heavily depends on the chosen sequence of projections that are iteratively repeated. The development of heuristics for properly choosing this sequence is a point that concentrates our current efforts.

References

Alfakih, A.Y., Khandani, A., and Wolkowicz, H. (1999),"Solving Euclidean Distance matrix completion problems via Semidefinite Programming," *Computational Optimization and Applications*, Vol. 12, No. 1-3, pp. 13-30.

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

Gosselin, C., and Sefrioui, J. (1991), "Polynomial Solutions for the Direct Kinematic Problem of Planar Three-Degree-of-Freedom Parallel Manipulators," in *Proc. International Conf. on Advanced Robotics*, Pisa, Italy, pp. 1124-1129, 1991.

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

Laurent, M. (1998), "A Connection Between Positive Semidefinite and Euclidean Distance Matrix Completion Problems," *Linear Algebra and its Applications*, Vol. 273, No. 1-3, pp. 9-22.

Nielsen, J., and Roth, B. (1997), "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.

Nikitopoulos, T.G., and Emiris, I.Z. (2003), "Molecular Conformation Search by Matrix Perturbations," *preprint.*

Porta, J.M., Ros, L., Thomas, F., and Torras, C. (2004), "A Branch-and-Prune Algorithm for Solving Systems of Distance Constraints," *IEEE Trans. on Robotics and Automation*, to appear.

Saxe, J.B. (1979), "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.

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

Steward, G.W. (1998), *Matrix Algorithms. Volume I: Basic Decompositions*, SIAM, Philadelphia.

Tsai, L.-W. (1999) *Robot Analysis. The Machanics of Serial and Parallel Manipulators*, John Wiley and Sons Inc., 1999.