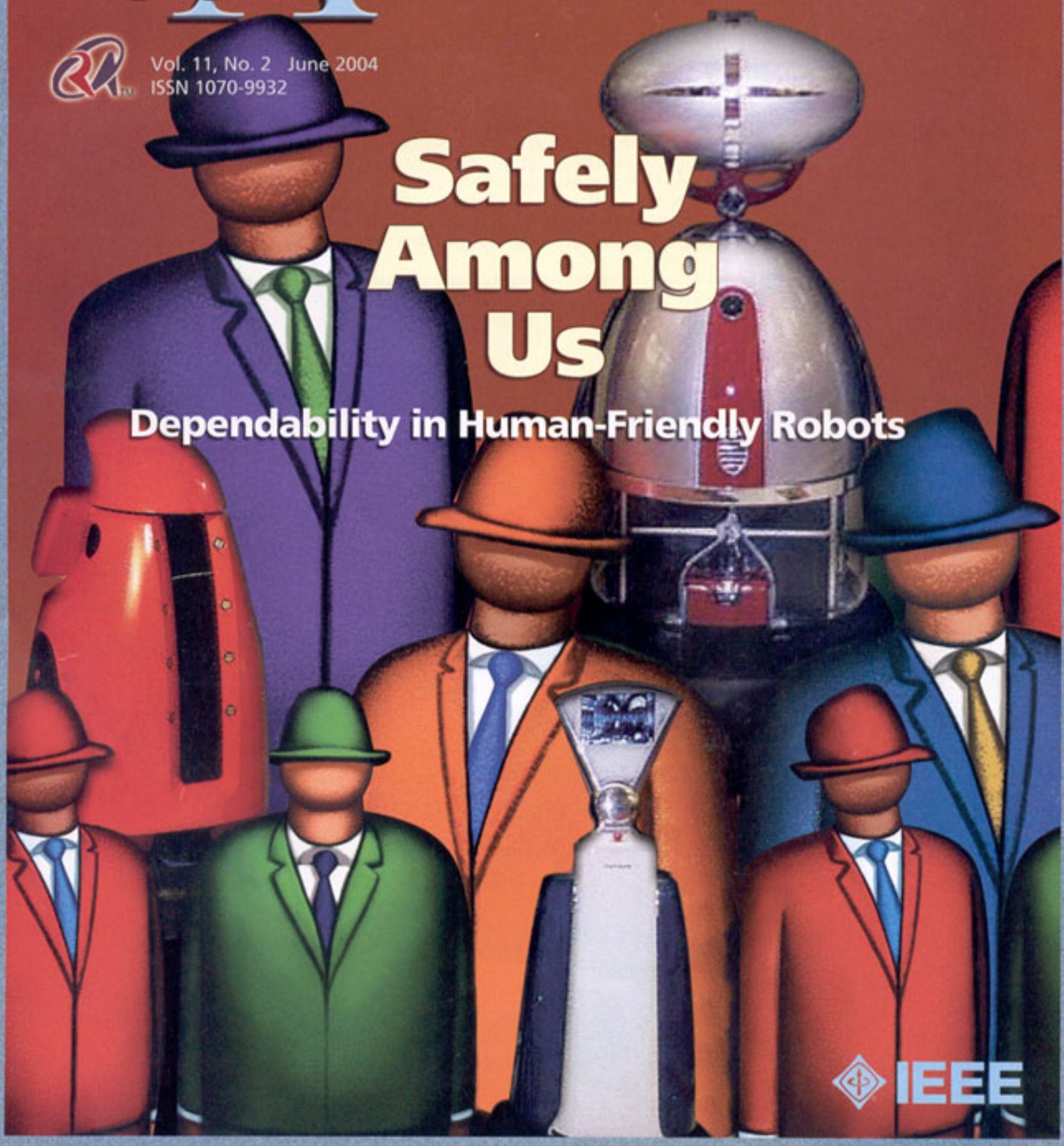
IEEE
Robotics & Automation
MAGAZINE

Vol. 11, No. 2 June 2004
ISSN 1070-9932

Safely Among Us

Dependability in Human-Friendly Robots



Kinematic Synthesis of Robotic Manipulators from Task Descriptions

Optimizing the Tasks at Hand

BY TAREK M. SOBH AND DANIEL Y. TOUNDYKOV

This article addresses the kinematic synthesis of robotic manipulators and presents a simple prototyping software tool. The tool, which runs under the Mathematica environment, automatically computes possible optimal parameters of robot arms by applying numerical optimization techniques to the manipulability function, combined with distances to the targets and restrictions on the dimensions of the robot. This work also discusses possible extensions of the proposed method of kinematic synthesis.

Solving Optimization Problems

Computing optimal geometry for robotic manipulators is one of the most intricate problems in contemporary kinematics. Mathematical equations that describe the behavior of kinematic chains are nonlinear; often contain thousands, sometimes even millions, of terms; and rarely have known closed-form solutions. Most of the existing analytical conclusions rely on rigorous analysis of some particular manipulator configurations, whereas attempts to generalize methods of kinematic synthesis usually end up in the domain of numerical analysis. The complexity of the optimal design problem remains a catalyst for the development of rapid prototyping, which allows engineers to determine structural flaws of the mechanisms by examining the behavior of their prototypes, as opposed to analyzing sophisticated mathematical models. Modern synthesis methods include minimization of cost functions [1], stochastic algorithms [2], distributed optimization [3], parameters space approach [4], and some other techniques [5]. This work concentrates on a popular algorithm for numerical optimization: minimizing functions with the *steepest descent* method.

A classical way of solving an optimization problem is to select several criteria that describe important aspects of the model, assign weight factors to them, and then find minima of the *cost function*, which is often the sum of the weighted criteria. Minima are located by examining the gradient of the function—the algorithm is known as the steepest descent method. In this project, instead of taking into account individual kinematic parameters, the cost function was composed from the expression for the manipulability measure and distances to the target points. A procedural package for Mathematica (v. 4.1, Wolfram Research Inc. 2002) [12] has been developed to test the new method of kinematic synthesis. The software uses a set of task-points, several weight factors, and produces a table of Denavit-Hartenberg parameters [6] describing a manipulator that attains high manipulability at each of the targets. The program actuates the Robotica package (v. 3.60, Copyright 1993 Board of Trustees, University of Illinois) [13] to display the results and employs a simple manipulability measure first defined by Yoshikawa in 1983 [7]. The framework can be extended to encompass more complicated and accurate models.

Manipulability Measures

In order to analyze the efficiency of robots, one needs some quantitative measure of their performance. The theory of kinematic synthesis has significantly advanced during the past decade and various ways have been developed to describe the manipulability and dexterity of robots. Many of these approaches were derived from the definition of manipulability proposed by Yoshikawa [7]. Given a manipulator with N degrees of freedom, denote joint variables by an N-dimensional vector $\mathbf{q}$. Let $J(\mathbf{q})$ be the velocity jacobian of the manipulator. When $J(\mathbf{q})$ loses its full rank, the kinematic chain loses one of its degrees of freedom; hence, manipulability is defined as:

$$w = \sqrt{\det J(\mathbf{q}) J^T(\mathbf{q})}. \quad (1)$$

For nonredundant manipulators this expression reduces to

$$w = |\det J(\mathbf{q})| \quad (2)$$

By applying the singular value decomposition to the Jacobian: $J(\mathbf{q}) = U \Sigma V^T$ [8] it can be shown that w is proportional to

1070-9932/04/$20.00©2004 IEEE

the volume of the ellipsoid with principals axes $\sigma_1\mathbf{u}_1, \sigma_2\mathbf{u}_2, \ldots, \sigma_s\mathbf{u}_s$, where $s = \max \text{rank } J(\mathbf{q})$, σ_i are singular values of $J(\mathbf{q})$, and $\mathbf{u}_i$ is the ith column vector of U. When J loses rank the ellipsoid becomes *degenerate*; i.e., one or more of its principal axes have zero length.

During the past few years the manipulability ellipsoid approach has acquired significant enhancements. A number of manipulability measures for parallel mechanisms [9] have been derived; these equations included constraints on joint velocities and forces. J. Lee has developed a method of *manipulability polytopes* [10], which is more involute then the ellipsoid approach but provides a better assessment of the mechanism's efficiency.

Numerical Optimization

All of the previous manipulability measures involve lengthy nonlinear mathematical expressions in many variables. Contemporary mathematics does not possess generic techniques for obtaining closed-form solutions to nonlinear equations, and iterative methods still retain a firm position among the tools for solving complicated systems. Classical optimization usually refers to combining several criteria expressions into a single multivariable function, called the *cost function*, and then iteratively searching for solutions that minimize that function for a particular domain. The steepest-descent algorithm finds minima by always walking in the direction opposite to gradient of the function. This procedure is slow, if compared to locally convergent techniques such as Newton-Raphson method, for instance; however, steepest descent does not require the initial guess to be close to the actual solution. If the range of the function does not contain negative values, then the algorithm always converges, with the exception of some rare cases when the gradient vanishes—then the result may appear to be a maximum or a saddle point.

If $\mathbf{x}$ is the solution vector, then choose n criteria $f_i(\mathbf{x}) = 0, i = 1, 2 \ldots n$, and build the function for optimization:

$$F(\mathbf{x}) = \sum_i^n f_i^2(\mathbf{x}). \tag{3}$$

The expression (3) is then minimized, and if the discovered minimum lies close to 0 then the result yields a good approximation to the optimal value of $\mathbf{x}$, provided that such optimum exists.

This method has certain disadvantages: for instance, some criteria may be discontinuous or may involve complex numbers; also, there is no way to determine whether the encountered minimum is local or absolute. Nevertheless, the steepest-descent procedure, if carefully applied, provides a reliable method to quickly design kinematic chains, and the obtained results can always become initial approximations for other design techniques.

Constructing the Optimization Measure

The goal of the project was to develop a fast and simple synthesis tool for robotic manipulators. The objectives were:

- generality
- fast results
- ease of use.

It was intended that a designer would enter several spatial points into a computational module and within a reasonable amount of time receive a description of a robot that would be able to efficiently operate among the given targets. The

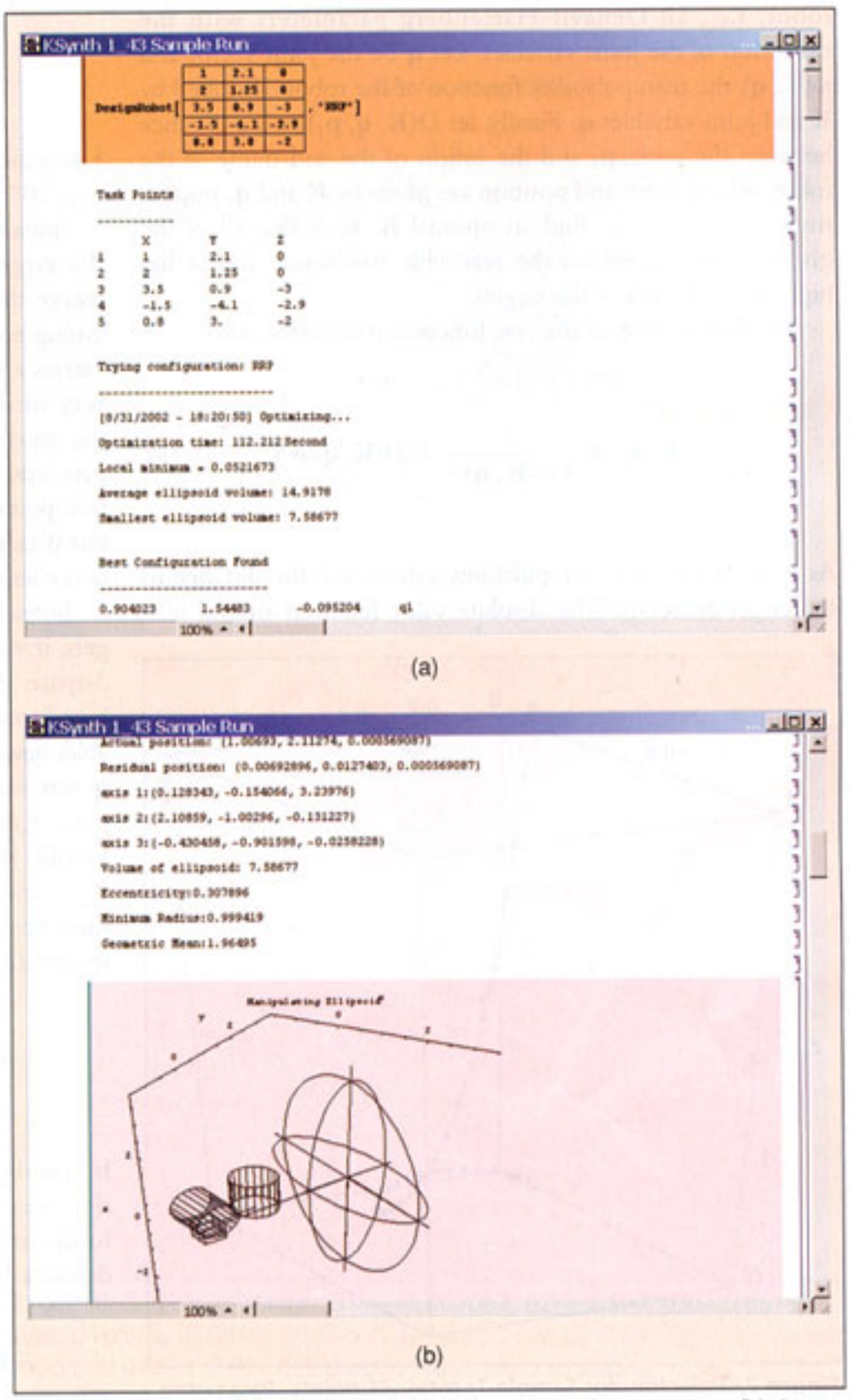


Figure 1. *(a) Mathematica environment executing the program. (b) Continuation of the program execution.*

The goal of the project was to develop a fast and simple synthesis tool for robotic manipulators. The objectives were generality, fast results, and ease of use.

synthesis algorithm was based on the manipulability measure, described by (1) and (2) and the steepest-descent method.

Suppose there are m task-points: $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \ldots, \mathbf{p}_m$. Let $\mathbf{K}$ be the set of constant parameters in the definition of the robot; i.e., all Denavit-Hartenberg parameters with the exception of the joint variables. Let $\mathbf{q}$ be the joint vector and $w(\mathbf{K}, \mathbf{q})$ the manipulability function of the robot described by $\mathbf{K}$ and joint variables $\mathbf{q}$. Finally, let $D(\mathbf{K}, \mathbf{q}, \mathbf{p}_i)$be the distance between the point $\mathbf{p}_i$ and the origin of the end-frame of the robot, whose form and position are given by $\mathbf{K}$ and $\mathbf{q}$, respectively. The task is to find an optimal $\mathbf{K}$, such that all of the given points fall within the reachable workspace and w has high values at each of the targets.

The first version of the cost function considered was

$$F_i(\mathbf{K}, \mathbf{q}) = \frac{1}{|\, w(\mathbf{K}, \mathbf{q})\,|} + D(\mathbf{K}, \mathbf{q}, \mathbf{p}_i). \qquad (4)$$

As F_i decreases, the manipulability grows, and the distance to the target decreases. The absolute value function around w is

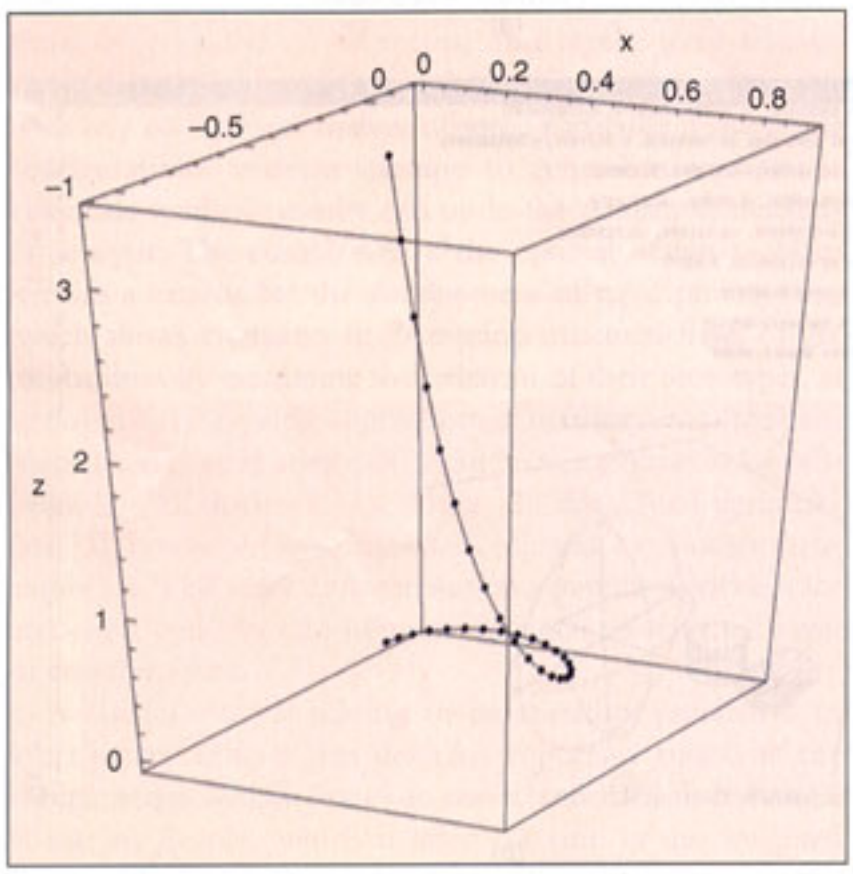


Figure 2. *Trajectory for Sample 1; total: 31 points. Parametric representation: $x = t\sin 2t$, $y = t\cos 3t$, $z = t^3$; t ranges from 0 to 1.5 with the step of 0.05.*

difficult to work with, and D contains a square root: $\sqrt{x_1^2 + x_2^2 + x_3^2 + \ldots}$ (here x_i are the components of the residual vector), which only increases the complexity of the expression. So, (4) was transformed into

$$F_i(\mathbf{K}, \mathbf{q}) = \frac{1}{w^2(\mathbf{K}, \mathbf{q})} + D^2(\mathbf{K}, \mathbf{q}, \mathbf{p}_i). \qquad (5)$$

After minimization, $\mathbf{q}$ becomes the inverse kinematic solution for the point $\mathbf{p}_i$ and $\mathbf{K}$ describes a robot that attains high manipulability at that point. If during the optimization phase the algorithm encounters a singular Jacobian, the value of F becomes infinite, therefore an extra term b is needed to eliminate the singularities.

$$F_i(\mathbf{K}, \mathbf{q}) = \frac{1}{w^2(\mathbf{K}, \mathbf{q}) + b} + D^2(\mathbf{K}, \mathbf{q}, \mathbf{p}_i) \qquad (6)$$

b is typically very small so that it does not distort the results; e.g., 10^{-10}.

Initially it was intended to solve the problem for each of the target-vectors individually and then use heuristics to merge the solutions into a single parameter table. Yet, combining results usually implies some form of averaging, which carries a negative aspect: when trying to merge very large and very small values of a particular parameter, say, a link length, the outcome is smaller than the larger value; hence, some targets may become unreachable. Averaging works only when task-points lie on, or near, some sphere centered at the origin, but if targets are arbitrarily distributed in space, this approach becomes unacceptable.

Instead of computing a separate solution for each of the targets, it was proposed to treat the point-set as a single object in 3-space. All manipulability functions and distances were combined into a single expression for optimization. The joint variables have unique values at every task-point, so the joint vector $\mathbf{q}$ was made different for each pair {*manipulability_at_point_i, distance_to_point_i*}. For example, if the first angle parameter is variable then at point $\mathbf{p}_1$ it is called $\theta_{[1,1]}$, at the second point: $\theta_{[2,1]}$, and so on. So for each of the m points there is a separate joint vector $\mathbf{q}_i$. Now, writing $w(\mathbf{K}, \mathbf{q}_i)$ and $D(\mathbf{K}, \mathbf{q}_i, \mathbf{p}_i)$ as w_i and D_i, respectively, one can reformulate (6) as follows:

$$F(\mathbf{K}, \mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_m) = \sum_i^m \frac{1}{w_i^2 + b} + D_i^2. \qquad (7)$$

It was discovered that (7) sometimes resulted in poor precision; i.e., the positioning error for some targets could go up to the order of 10^{-1}. So a weight factor ε was attached to the distance D:

$$F(\mathbf{K}, \mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_m) = \sum_i^m \frac{1}{w_i^2 + b} + \varepsilon \times D_i^2. \qquad (8)$$

By increasing ε one can increase the precision of the manipulator.

Another issue was that (7) and (8) always produced very large arms, often several times longer than the distance to the remotest target. Therefore, it was decided to introduce another term $\xi \times L$:

$$\xi \times L = \xi \times \sum_{\substack{j \\ d_j \notin \mathbf{q}}}^{N} (a_j + d_j). \tag{9}$$

Here a_j and d_j are the length and offset Denavit-Hartenberg parameters, respectively; the restriction $d_j \notin \mathbf{q}$ ensures that only invariant offsets are included. ξ is the size dumping factor: values of order higher than 10^{-2} notably reduce the dimensions of the manipulator; however, they also decrease manipulability, because the available workspace shrinks. The final expression for optimization is

This tool can significantly aid in robot design and prototyping and is another significant step toward the automated generation of optimal robotic mechanisms from task descriptions.

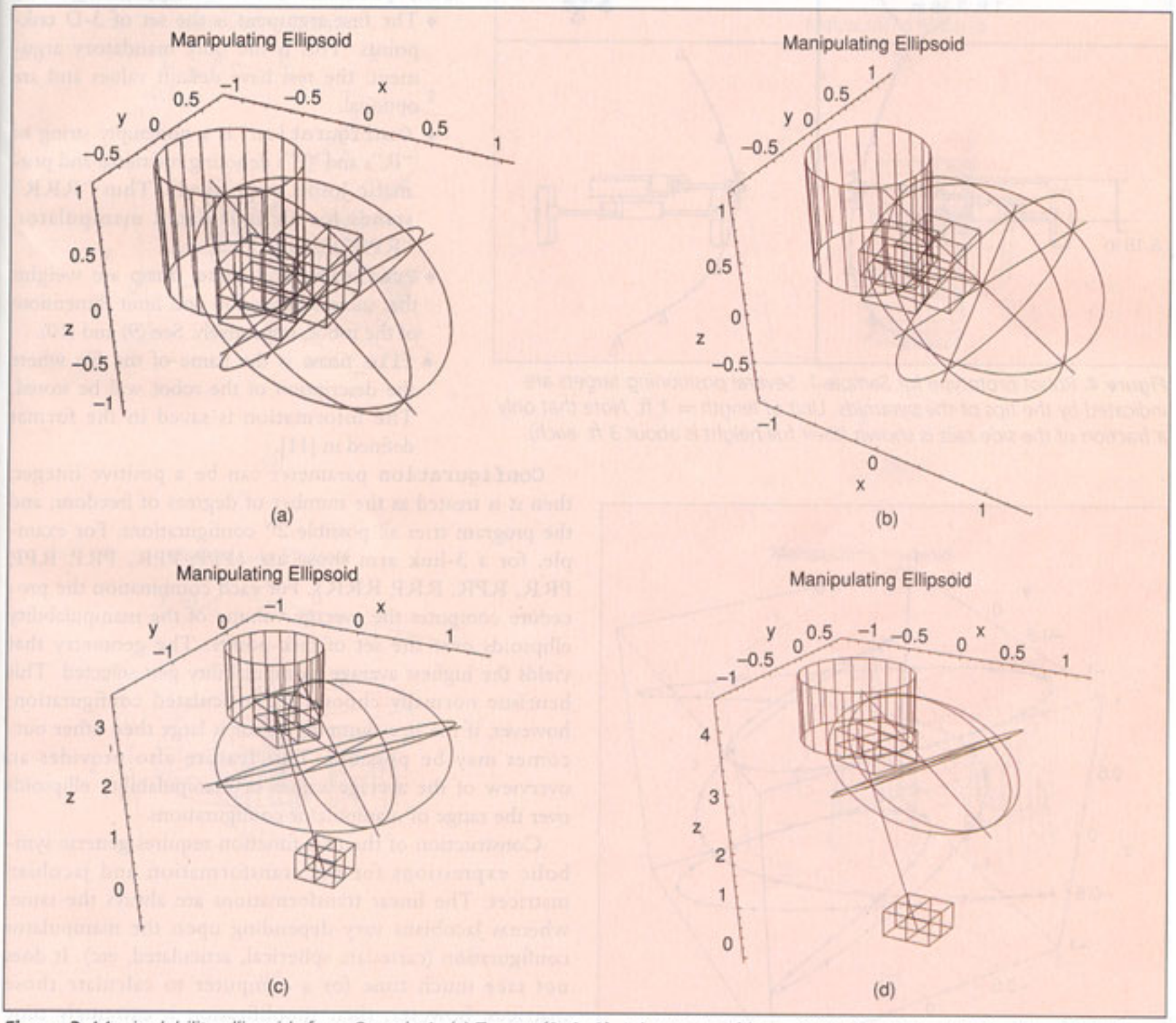


Figure 3. *Manipulability ellipsoids from Sample 1. (a) Target: {0, 0, 0}. Joint vector: {d1→–0.002083, q2→1.574, d3→0.6519}. Residual position: {0.03634, 0.001228, 0.0002705}. Ellipsoid volume: 0.6519. (b) Target: {0.3525, 0.09855, 0.09113}. Joint vector: {d1→0.08826, q2→1.661, d3→0.9854}. Residual position: {0.01008, 0.001385, 0.00006366}. Ellipsoid volume: 0.9855. (c) Target: {0.46898, –0.68637, 2.744}. Joint vector: {d1→2.738, q2→0.9861, d3→1.275}. Residual pos.: {0.00404, –0.002133, –0.0002591}. Ellipsoid vol.: 1.275. (d) Target: {0.2117, –0.31619, 3.375}. Joint vector: {d1→3.372, q2→1.195, d3→0.8913}. Residual pos.: {0.01338, –0.004412, 0.0001901}. Ellipsoid vol.: 0.8912.*

$$F(\mathbf{K}, \mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_m) = \xi \times L + \sum_i^m \frac{1}{w_i^2 + b} + \varepsilon \times D_i^2 \quad . \tag{10}$$

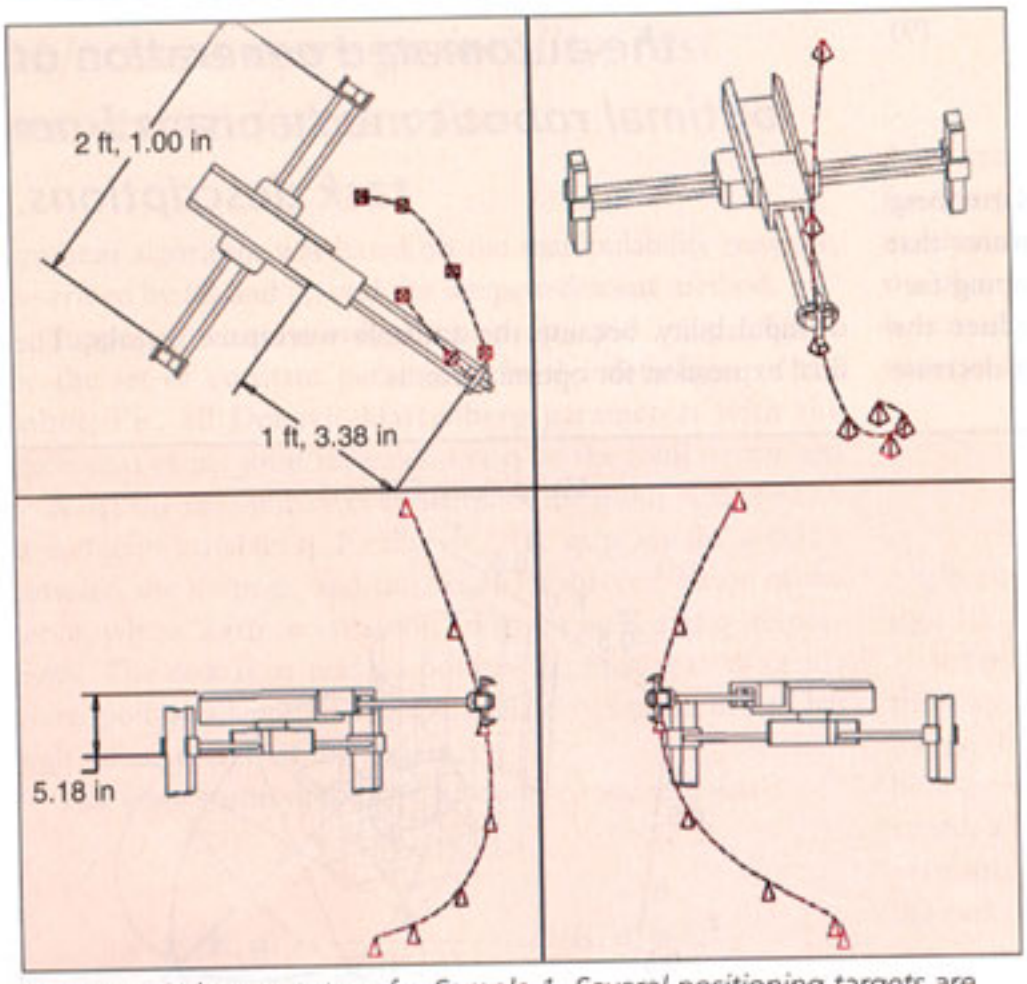


Figure 4. *Robot prototype for Sample 1. Several positioning targets are indicated by the tips of the pyramids. Unit of length = 1 ft. Note that only a fraction of the side rails is shown (their full height is about 3 ft each).*

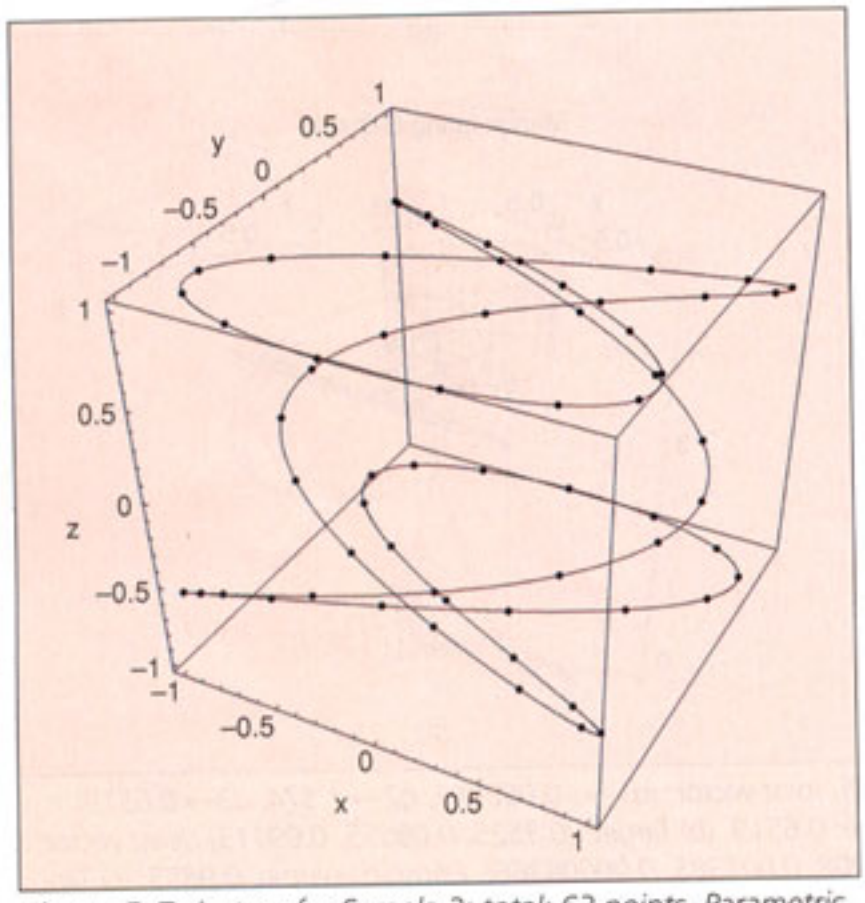


Figure 5. *Trajectory for Sample 2; total: 63 points. Parametric representation: $x = \cos 5t$, $y = \sin 3t$, $z = \sin t$; t ranges from 0 to 2π with the step of 0.1.*

A Simple Design Tool

A set of procedures for Mathematica (v. 4.1) have been written to automate the kinematic synthesis of robot arms.

The source code, along with instructions, can be downloaded from http://www.bridgeport.edu/~sobh/ieeeram2002.html.

A sample run of the program is shown on Figure 1(a) and (b). The software uses the traditional manipulability ellipsoid measures (1) and (2). Detailed descriptions of available procedures and their parameters can be found in the program itself. This section will focus only on the main module that triggers the optimization:

`DesignRobot[task_points, configuration, precision, size_dump, file_name]`

- The first argument is the set of 3-D task-points. This is the only mandatory argument: the rest have default values and are optional.
- `Configuration` is a nonempty string of "R"s and "P"s denoting rotational and prismatic joints respectively. Thus "RRR" stands for an articulated manipulator, "RPP" for cylindrical, etc.
- `Precision` and `size_dump` are weights that increase precision and limit dimensions of the robot, respectively. See (9) and (10).
- `file_name` is the name of the file where the description of the robot will be stored. The information is saved in the format defined in [11].

`Configuration` parameter can be a positive integer, then it is treated as the number of degrees of freedom, and the program tries all possible 2^N configurations. For example, for a 3-link arm those are {PPP, PPR, PRP, RPP, PRR, RPR, RRP, RRR}. For each combination the procedure computes the average volume of the manipulability ellipsoids over the set of task-points. The geometry that yields the highest average manipulability gets selected. This heuristic normally chooses the articulated configuration; however, if the size-dumping factor is large then other outcomes may be possible. This feature also provides an overview of the average values of manipulability ellipsoids over the range of manipulator configurations.

Construction of the cost function requires generic symbolic expressions for the transformation and jacobian matrices. The linear transformations are always the same, whereas Jacobians vary depending upon the manipulator configuration (cartesian, spherical, articulated, etc). It does not take much time for a computer to calculate those matrices; however, their simplification is extremely time consuming and for redundant manipulators may take a few hours. Nevertheless, since the matrices are completely generic, they need to be derived only once. After the very first computation the expressions are saved and are reloaded for subsequent operations.

Results

This section contains results from two sample program runs. Each case presents some parametric trajectory $\{x(t), y(t), z(t)\}$, $t \in [a, b]$ and demonstrates what happens when a number of points that belong to the curve are submitted to the optimization module. The precision and size-dumping factors vary for the two cases. The volume of the manipulability ellipsoids relates to the unit of length used to measure the dimensions of the robot. In the first sample the types of joints were selected by the software, whereas in the other the configuration was forced by the user.

More case studies are available at: http://www.bridgeport.edu/~sobh/ieeeram2002.html.

Substituting different symbols in place of joint variables for each of the task-points greatly increases the number of unknowns to compute.

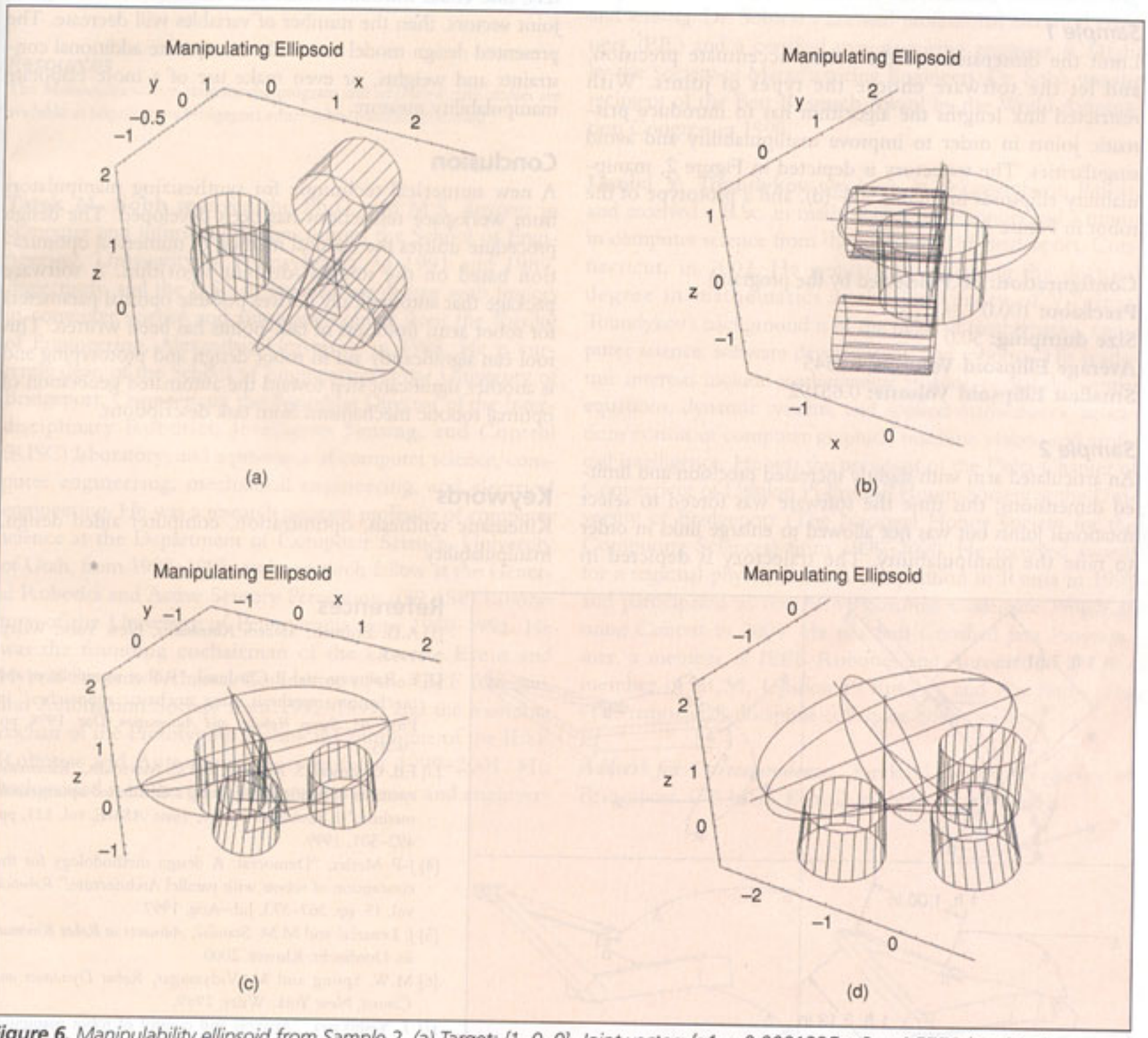


Figure 6. *Manipulability ellipsoid from Sample 2. (a) Target: {1, 0, 0}. Joint vector: {q1→0.0001335, q2→4.5756 (mod 2π), q3→–2.725}. Residual position: {–0.01909, –0.0006325, 0.005302}. Ellipsoid volume: 0.9705. (b) Target: {–0.4161, 0.9320, 0.3894}. Joint vector: {q1→33.41 (mod 2π), q2→21.30 (mod 2π), q3→8.937 (mod 2π)}. Residual position: {–0.0002037, –0.001812, 0.01891}. Ellipsoid volume: 1.1671. (c) Target: {0.2837, 0.1411, 0.8415}. Joint vector: {q1→6.752, q2→2.712, q3→2.140}. Residual position: {0.09413, 0.04726, 0.01501}. Ellipsoid vol.: 0.8692. (d) Target: {–0.9972, –0.5507, 0.9463}. Joint vector: {q1→–15.20 (mod 2π), q2→–9.513 (mod 2π), q3→–3.864}. Residual position: {–0.005700, –0.003291, 0.009052}. Ellipsoid vol.: 1.853.*

The presented design model can easily incorporate additional constraints and weights, or even make use of a more elaborate manipulability measure.

Sample 1

Limit the dimensions of the robot, accentuate precision, and let the software choose the types of joints. With restricted link lengths the algorithm has to introduce prismatic joints in order to improve manipulability and avoid singularities. The trajectory is depicted in Figure 2, manipulability ellipsoids in Figure 3(a)–(d), and a prototype of the robot in Figure 4.

Configuration: PRP (selected by the program)
Precision: 100.0
Size dumping: 50.0
Average Ellipsoid Volume: 1.2545
Smallest Ellipsoid Volume: 0.65192

Sample 2

An articulated arm with slightly increased precision and limited dimensions; this time the software was forced to select rotational joints but was not allowed to enlarge links in order to raise the manipulability. The trajectory is depicted in Figure 5, manipulability ellipsoids in Figure 6(a)–(d), and a prototype of the robot in Figure 7.

Configuration: RRR (selected by the user).
Precision: 25
Size dumping: 12
Average Ellipsoid Volume: 1.451
Smallest Ellipsoid Volume: 0.81001

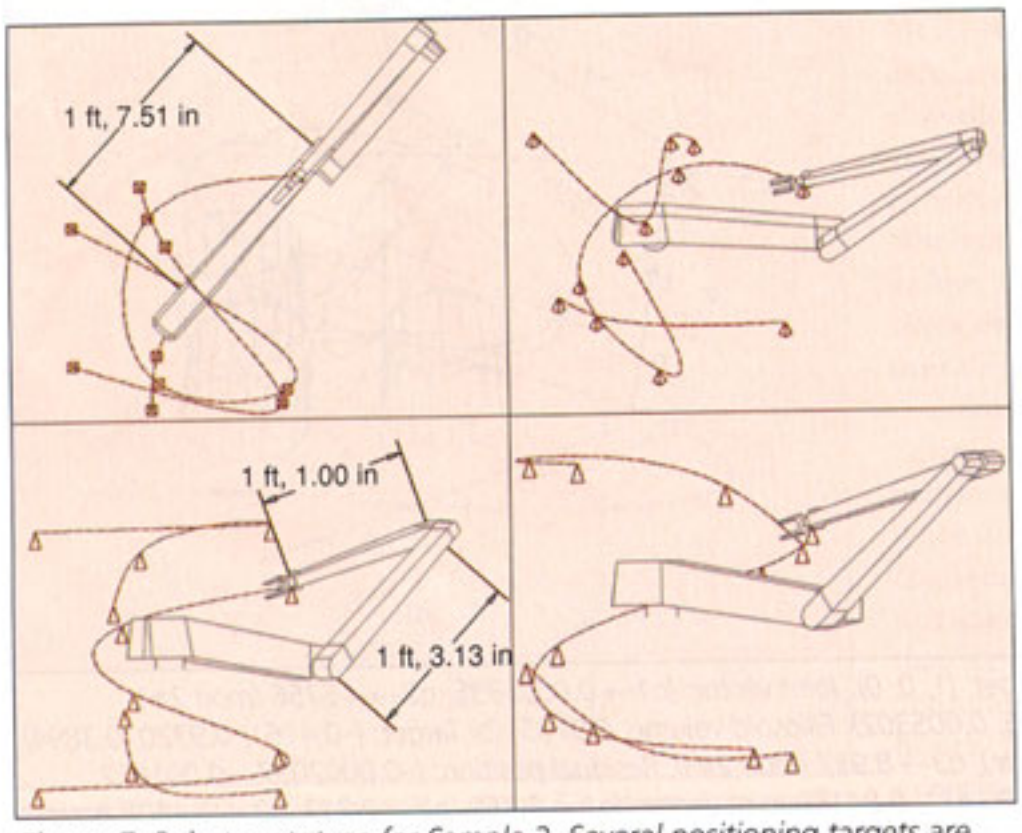


Figure 7. *Robot prototype for Sample 2. Several positioning targets are indicated by the tips of the pyramids. Unit of length = 1 ft.*

Future Developments

Substituting different symbols in place of joint variables for each of the task-points greatly increases the number of unknowns to compute. If, instead of using distinct parameters, one could introduce functional relations connecting the joint vectors, then the number of variables will decrease. The presented design model can easily incorporate additional constraints and weights, or even make use of a more elaborate manipulability measure.

Conclusion

A new numerical technique for synthesizing manipulators from workspace restrictions has been developed. The design procedure utilizes the classical method of numerical optimization based on the steepest-descent algorithm. A software package that automatically derives possible optimal parameters for robot arms from sets of task-points has been written. This tool can significantly aid in robot design and prototyping and is another significant step toward the automated generation of optimal robotic mechanisms from task descriptions.

Keywords

Kinematic synthesis, optimization, computer aided design, manipulability.

References

[1] A.G. Erdman, *Modern Kinematics*. New York: Wiley, 1993.

[2] E. Ramstein and P. Chedmail, "Robot manipulators and mechanisms synthesis using stochastic approaches," in Proc. *Int. Symp. Robotics and Automation*, Dec. 1998, pp. 79–85.

[3] F.B. Ouezdou, S. Regnier, and C. Mavroidis, "Kinematic synthesis of manipulators using a distributed optimization method," *J. Mechanical Design, Trans. ASME*, vol. 121, pp. 492–501, 1999.

[4] J-P. Merlet, "Democrat: A design methodology for the conception of robots with parallel Architecture," *Robotica*, vol. 15, pp. 367–373, Jul.–Aug. 1997.

[5] J. Lenarčič and M.M. Stanišić, *Advances in Robot Kinematics*. Dordrecht: Kluwer, 2000.

[6] M.W. Spong and M. Vidyasagar, *Robot Dynamics and Control*, New York: Wiley, 1989.

[7] T. Yoshikawa, "Analysis and control of robot manipulators with redundancy," in *Robotics Research, The First International Symp.* Cambridge, MA: MIT Press, 1984, pp. 735–747.

[8] V.C. Klema and A.T. Laub, "The singular value decomposition: Its computation and some applications," *IEEE Trans. Automatic Contr.*, vol. AC25, no. 2, pp. 164–176, 1980.

[9] H. Keum-Shik, K. Jeom-Goo, and L. Seung-Hwan, "Optimal link design of a parallel machine tool based on manipulability analysis," in *Proc. IEEE Int. Symp. on Industrial Electronics '01*, 2001, pp. 1747–1752.
[10] L. Jihong, "A study on the manipulability measures for robot manipulators," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems '97*, 1997, pp. 1458–1465.
[11] M.W. Spong, "The robotica manual," [online] 1993 [downloaded 21 Dec. 2001]. Available http://robot0.ge.uiuc.edu/~spong/Robotica/newman.ps.Z

Computer Software

[12] *Mathematica* v. 4.1. Wolfram Research Inc, 2002. www.wolfram.org
[13] *Robotica* v. 3.60 [online] University of Illinois, 1993 [downloaded 21 Dec. 2001]. Available http://robot0.ge.uiuc.edu/~spong/ Robotica/robotica.m

Resources

The *Mathematica* source code for the program and additional case studies are available at http://www.bridgeport.edu/~sobh/ieeeram2002.html

Tarek M. Sobh received the Ph.D. and M.S. degrees in computer and information science from the School of Engineering, University of Pennsylvania, in 1991 and 1989, respectively, and the B.Sc. in engineering degree with honors in computer science and automatic control from the Faculty of Engineering, Alexandria University, in 1988. He is currently dean of the School of Engineering at the University of Bridgeport, Connecticut; the Founding Director of the Interdisciplinary Robotics, Intelligent Sensing, and Control (RISC) laboratory; and a professor of computer science, computer engineering, mechanical engineering, and electrical engineering. He was a research assistant professor of computer science at the Department of Computer Science, University of Utah, from 1992–1995, and a research fellow at the General Robotics and Active Sensory Perception (GRASP) Laboratory of the University of Pennsylvania from 1989–1991. He was the founding cochairman of the Discrete Event and Hybrid Systems Technical Committee of the IEEE Robotics and Automation Society from 1992–1999, and the founding cochair of the Prototyping Technical Committee of the IEEE Robotics and Automation Society from 1999–2001. His background is in the fields of computer science and engineering, control theory, robotics, automation, manufacturing, AI, computer vision, and signal processing. Dr. Sobh's current research interests include active sensing/imaging under uncertainty, robots and electromechanical systems prototyping, and sensor-based distributed control schemes. He has published over 130 refereed journal and conference papers and book chapters in these and other areas. Dr. Sobh is active in consulting and providing service to many industrial organizations and companies. He has consulted for many companies and institutions in the United States, Switzerland, India, Malaysia, UAE, and Egypt to support projects in robotics, automation, manufacturing, sensing, numerical analysis, control and engineering education. Dr. Sobh has been awarded many grants to pursue his work in robotics, automation, manufacturing, and sensing. Dr. Sobh is a licensed professional electrical engineer (P.E.) and a certified manufacturing engineer (CMfgE) by the Society of Manufacturing Engineers. Dr. Sobh was the recipient of the Best Research Award by the World Automation Congress in 1998.

Daniel Y. Toundykov was born in Yekaterinburg, Russia, and received a B.Sc. in mathematics with honors and a minor in computer science from the University of Bridgeport, Connecticut, in 2002. He is currently pursuing the doctoral degree in mathematics at the University of Virginia. Toundykov's background is in the fields of mathematics, computer science, software development, and robotics. His academic interests include mathematical analysis, partial differential equations, dynamic systems, and applied mathematics; avocations consist of computer graphics, machine vision, and artificial intelligence. He was the president of the Delta Chapter of Connecticut of Upsilon Pi Epsilon Honor Society at the University of Bridgeport (The National Honor Society for the Computing Sciences) from 1999–2000. He received awards for a regional physics research competition in Russia in 1998 and participated in the ACM National Collegiate Programming Contest in 2000. He is a Sun Certified Java Programmer, a member of IEEE Robotics and Automation Society, member of ACM, Upsilon Pi Epsilon, and Phi Kappa Phi (The national all-discipline collegiate honor society).

Address for Correspondence: Tarek M. Sobh, University of Bridgeport, CT, USA. E-mail: sobh@bridgeport.edu.