Journal of Computing and Information Science in Engineering

Application Briefs

The Structure of an Extensible Java Applet for Spatial Linkage Synthesis

C. L. Collins, J. M. McCarthy, A. Perez, and H. Su
Department of Mechanical and Aerospace Engineering
University of California
Irvine, CA 92697

This paper describes a new approach to computer-aided spatial linkage design which uses the dynamic binding feature of Java to provide a extensible software system. The spatial linkages that we focus on are constructed from one or more spatial open chains that are connected to a single workpiece. Each open chain has less than six degrees of freedom and is termed a serial chain primitive. The goal of the design system is to determine the dimensions of a set of primitives each of which has a workspace that includes the set of specified workpiece positions. These chains are then assembled together to form candidate parallel linkage designs. There are many serial chain primitives each of which has a specialized constraint solver. These primitives can be assembled into many different parallel linkages, each of which needs an analysis routine. Our approach to the challenge of generating all of these routines is to abstract the design process and structure our computer-aided linkage design system to allow integration of specialize synthesis and analysis routines developed over time by user-collaborators. [DOI: 10.1115/1.1486217]

Introduction

Computer-aided linkage design software computes the form of a device from a function specified by the designer. Function-to-form linkage design tools have recently been commercialized by SyMech Inc., SyMech [1] and Heron Technologies, WATT [2]. These systems compute the dimensions for several planar linkage topologies, which are combinations of 4R closed chains and 2R open chains, given functional specifications provided by the user. Perhaps the first of this type of software system was KynSyn (Kaufman [3]) which focussed solely on the 4R planar topology which was dimensioned to guide a body through a specified set of positions—R denotes a revolute or hinged joint. RECSYN (Waldron and Song [4]) and LINCAGES (Erdman and Gustafson [5]) added features that simplified the design process but again focussed on planar 4R and later planar 6R linkages.

The linkage design software Sphinx (Larocelle et al. [6]) and later SphinxPC (Ruth and McCarthy [7]), extended Kaufman's strategy to 4R linkages that move on a sphere providing the first design tool for the design of linkages for spatial movement. Furlong et al. [8] used virtual reality to assist the designer's specification of desired function of a spherical 4R linkage as well as to evaluate the resulting computed device. The first design software for a true spatial linkage was Larochelle's SPADES software [9] which computed a spatial 4C closed chain to guide a body through four spatial positions—C denotes a cylindric joint which allows rotation about and sliding along a given axis.

The relative positions of links in a spatial chain have six degrees of freedom, as opposed to three for planar chains. This provides a richer array of linkage topologies that can be dimensioned to meet a desired functional specification. It is not enough to simply choose one of these topologies and develop a software system for this one alone. The designer needs the opportunity to see the result of fitting an array of topologies to a specified function. This means the system should compute a variety of devices ranging from a one degree of freedom spatial linkage to a five degree of freedom constrained robot or platform linkage. This is a significant expansion of the complexity of the design system.

The Design Process

Our goal is to develop a spatial linkage design system that is structured so that it can be extended by user-collaborators to include a broad range of spatial linkage topologies of interest to the designer. We have examined existing computer-aided linkage design systems and identified a fundamental structure that we believe is generic to the function-to-form process for spatial linkages.

The general linkage design methodology starts with the specification of the desired function. This can be done in the form of a discrete set of goal positions for the workpiece. A serial chain topology is then selected, which is fit to the positions using a constraint solver adapted to the geometry of the chain. These solvers are usually special purpose routines that rapidly compute either a finite set of solutions, or a parameterized solution space (see Suh and Radcliffe [10], or McCarthy [11]). For spatial linkages this process can be applied to a variety of serial chain primitives selected by the user. Then, the various serial chains are assembled to define candidate linkage designs. The analysis and simulation of each candidate design is required to evaluate the movement and mechanical advantage of the device as it performs the desired function.

Current linkage design systems are structured to apply this design process to either the planar 4R and 6R linkage topologies, the spherical 4R, or the spatial 4C topologies. In what follows, we describe a Java-based system that allows this process to be abstracted for open chain primitives, which are then assembled to form various spatial linkage topologies, including those of the current systems. Of importance is the fact that the various primitives can be integrated independently into the design system as they are developed by user-collaborators. Similarly the spatial linkage analysis routines that are fundamental to the evaluation of various candidates and often challenging to develop can be in-

Contributed by the Engineering Simulation and Visualization Committee for publication in the JOURNAL OF COMPUTING AND INFORMATION SCIENCE IN ENGINEERING. Manuscript received November 2001; Revised April 2002. Associate Editor: S. Szykman.

cluded as needed. The variety of spatial linkage topologies provides the motivation for our abstraction of the linkage design process and our development of a topology independent software structure for spatial linkage synthesis.

Software Functionality

We have called our prototype computer-aided design system Synthetica. Its functionality is based on our model of the linkage design process. We confine its scope to the synthesis of serial chain primitives (planar, spherical and spatial) and the analysis of parallel linkages assembled from these primitives.

The key components and flow of information for Synthetica are shown in Fig. 1. At every level, the architecture supports multiple linkage topologies and multiple analysis/synthesis routines. To accommodate this approach, a common interface is available for linkage analysis and synthesis routines which allows them to be dynamically integrated with the program. This feature is represented by the Class Loader element in the block diagram.

We separate the task specification from the linkage topology using a Design Matrix element that presents all available synthesis routines arranged according to the number of design positions, and the associated serial chain topology. This allows the user to direct the design process from either the task size or linkage topology perspective. It also provides a high level view of the available implemented routines.

Once a linkage topology and task size is selected, the task positions and other constraints can be specified. Since the number of positions and constraints required by a synthesis routine is topology dependent, the user interface for entering this information is dynamically generated according to the information provided by the specific routine. In addition, it is also desirable to be able to directly specify a set of linkage dimensions. Again, to accommodate multiple topologies, this interface is also dynamically generated for the specified the serial chain topology.

Whether a set of serial chains is synthesized, or defined directly, the next stage of the process is to allow the designer to view and manipulate the solution space in a variety of ways, as represented by the Solution Browser element. The solution browser allows the designer to change the solution space by modifying the task and topology, solving for additional serial chain topologies, or combining serial chain solutions into platform topologies.

Two components are provided to aid the designer in selecting a particular design solution. A linkage can be displayed and animated using the Linkage Viewer, or a range of linkages can be evaluated for a specific quantitative property by sending them to a Design Evaluator. Analysis routines can be general purpose or topology specific. For example, a general purpose kinematics solver has been implemented for animating any serial or platform linkage, while a specialized type evaluator has been implemented for RRSS architectures.

At any time, the task, topology, constraint, and solution information can be saved for future use, or for use in external programs. Thus a design solution can be saved along with the specifications used to synthesize it. In addition, link and joint geometries which are dynamically generated for 3D visualization purposes and can be exported for use in CAD packages.

Synthetica Packages

The first version of Synthetica is in the form of a Java applet which is designed to run under a web browser or applet viewer on Windows, and Macintosh operating systems. Java has a number of unique features that make it well suited for implementing our linkage design system. The Java language and API specifications have become fairly stable, and provide a flexible and powerful set of GUI elements to enhance user interaction (see the Java Tutorial [12,13] for an introduction). Java has networking capabilities built in to the Java API, and provides a facility for utilizing dynamically linked libraries on the client machine. This not only provides a means to more efficiently implement particularly complex computations, but it also allows Java programs to use existing core libraries such as OpenGL [14].

The object-oriented nature of Java also allows us to design a set of class specifications that enables collaborative development of special purpose synthesis algorithms and linkage analysis routines. The platform independent nature of Java allows these classes to be developed independently by researchers with their own software development resources and dynamically integrated with the applet/application at run-time.

Synthetica has four major components organized into Java packages as shown in Fig. 2. The main program integrates and coordinates the flow of information using the four underlying packages. The packages are briefly described as follows.

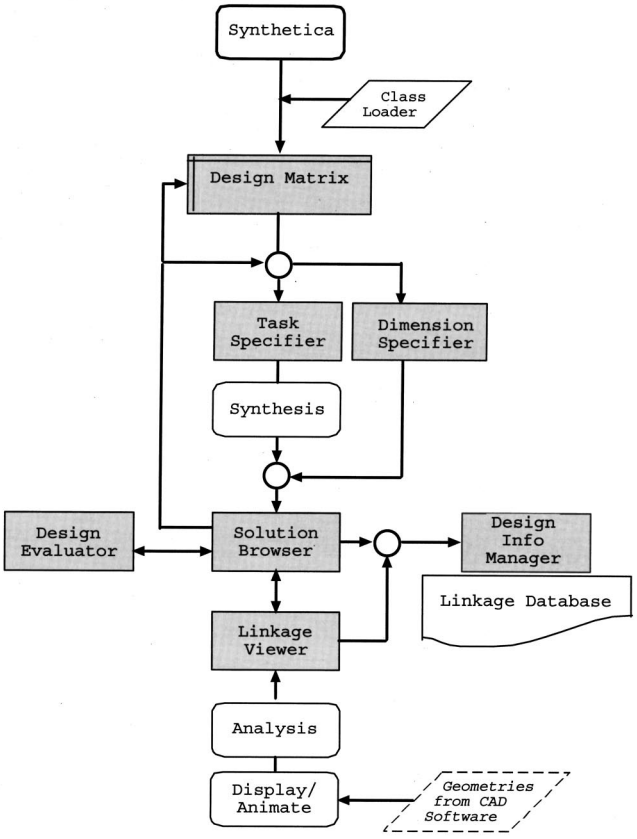

Fig. 1 Software Architecture

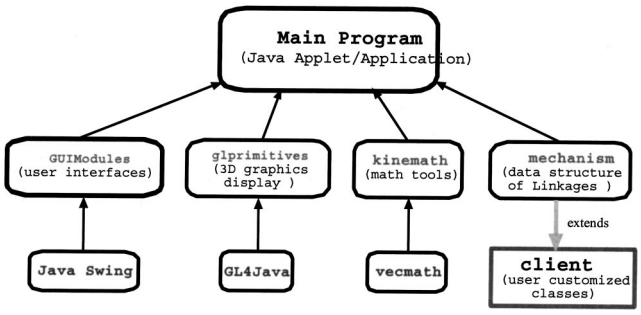

Fig. 2 Synthetica Package Organization

The GUIModules package provides classes that generate the graphical user interfaces for designer input and interaction. The Java 2 Swing library is used extensively in this package.

The glprimitives package provides the 3D graphics engine for the main program. It uses GL4Java (http://www.jausoft.com/gl4java/), a JNI encapsulation of the OpenGL graphics library, to generate and display mechanism models in real-time. This allows the applet to access the OpenGL DLL's on the client machine.

The kinemath package provides basic mathematical resources used by other parts of the program. It allows programmers to conveniently access mathematics operations such as vector and matrix manipulation, and kinematics operations such as motion interpolation. The javax.vecmath package is used extensively; this package is distributed with Java3D, but is independent of the Java3D specification and implementation.

The mechanism package provides the base classes and interface specifications for defining linkages. It is designed to allow a programmer to easily implement new specialized linkage analysis and synthesis routines for integration with the Synthetica design environment.

The GUIModules and mechanism packages encapsulate the primary functionality of Synthetica by defining the user interface modules and the linkage routines respectively. We now provide a more detailed discussion of these two packages.

GUIModules Package. The GUIModules package creates the windows and panels that allow users to modify/view the design data. The three key components are the Design Matrix, the Task Specifier, and the Solution Browser. The functionality of these modules is dependent on the data structures and class definitions found in the mechanism package, and on the standard Java 2 packages.

To accommodate the incorporation of multiple linkage and synthesis routines, we have taken advantage of Java's built-in capabilities to dynamically inspect class contents at runtime. This feature is represented in Fig. 1 by the Class Loader. When the Synthetica applet loads, the available mechanism and synthesis classes are inspected to determine the mechanism topology, and task characteristics. This information is saved in a map and presented to the user in the Design Matrix. The ability to load and view the contents of an arbitrary class allows us include classes generated by other programmers. In particular, the java.net.URLClassLoader class allows us to load any user defined class given its URL. The class specifications defined in the mechanism package ensures that the classes contain all the information we need to integrate them into our list of mechanisms and synthesis routines. The Task Specifier directs the user to enter all the information required by the particular synthesis routine. The Task Specifier is dynamically generated based on the information found in the synthesis routine implementation. Position and constraint defaults are provided along with a listing of constraint names.

The Solution Browser is a dynamically generated window containing a panel for each distinct serial chain topology under consideration. Within each panel, the user can select and view the parameters for any of solution generated by the synthesis routine. When multiple serial chain primitives are being considered, the user can view different combinations of them in the same window and select to display the resulting closed chain linkage in the Linkage Viewer.

Mechanism Package. In order to support multiple mechanism topologies and synthesis routines, a detailed package has been developed for programmers who wish to extend Synthetica. The mechanism package provides the classes which define the data structure of spatial linkages as well as a number of key interfaces for implementing special purpose kinematics and synthesis routines. The package is designed using the subclassing architecture of the Java language specification.

The mechanism package defines a DesignTask which contains the position and constraint specifications for the design. It also defines the Mechanism class which is the key component of the system. A mechanism is defined as a set of one or more serial chains composed of joints and links that act between a common gripper (or platform) and a base. The package also defines a Geometric−Object that is used to parameterize the geometry associated with the elements that make up the linkage.

We have also defined four Java interfaces. The ForwardKinematics and InverseKinematics interfaces specify the methods required for linkage position analysis. The Synthesizable interface specifies a set of methods for defining default tasks, constraint names, and for performing finite position synthesis. The Drawable interface provides the programmer with the ability to create customized geometry for the mechanism joints and links.

Implementing Mechanism and Synthesis Classes

Synthetica is structured to serve two roles. It can run as a standalone application/applet which causal users execute to synthesize and analyze implemented mechanisms. It is also a programming package. Interested users can define their own mechanisms and implement their own synthesis or kinematics algorithms. At run−time, the program examines the contents of these classes to determine the mechanism topology and DesignTask information needed to construct the Design Matrix.

All mechanism classes and synthesis routines are implemented in the same way using the following basic procedure:

1. Create a new class by extending a base mechanism class, for example, SerialMechanism.
2. Implement any interface methods you require, such as Synthesizable and InverseKinematics.
3. Compile the source together with the Synthetica API, to obtain a java.class file.
4. Run Synthetica, and load the new class file using the Class Loader.
5. The information found in the new class will be mapped into the Design Matrix and made available for use.

For the example design discussed in the next section, we implemented five classes. When the Class Loader encounters these classes it can inspect them to find that the first class defines and RR mechanism, the second class defines a TS mechanism, the third class contains a synthesis method for 3 position RR synthesis, the fourth class contains a synthesis method for 3 position TS synthesis, and the fifth class contains a method for 3 position synthesis, with a different constraint list than the fourth. In general, we have found it convenient to implement one class per mechanism topology, and one class per synthesis routine.

Example Design Session: RRTS Linkage Synthesis

In this section we provide an example design session for specifying and synthesizing an RRTS linkage. Since we have already defined our mechanism and synthesis classes, we can run the Synthetica applet and have it automatically inspect the classes. The resulting Design Matrix is shown in Fig. 3. The top portion of the Design Matrix window is a table of buttons. The columns are labeled with the available linkage primitives, and the rows are labeled with the number of design positions. Within the table, there are buttons used to select a particular synthesis method. The number that appears on the button indicates now many synthesis methods are available for that particular combination of positions and linkage primitive. In this case, we have one 3 position synthesis method for the RR chain, and two for the TS chain. When we press the button, information associated with the available synthesis routines is displayed below the table. This allows the designer to choose between different synthesis methods. We have also provided an additional button for manual specification of the linkage dimensions.

In this example, we wish to proceed to synthesize an RR chain. Pressing the Synthesize button leads to the generation of the Task Specifier, shown in Fig. 4. The Task Specifier window allows us

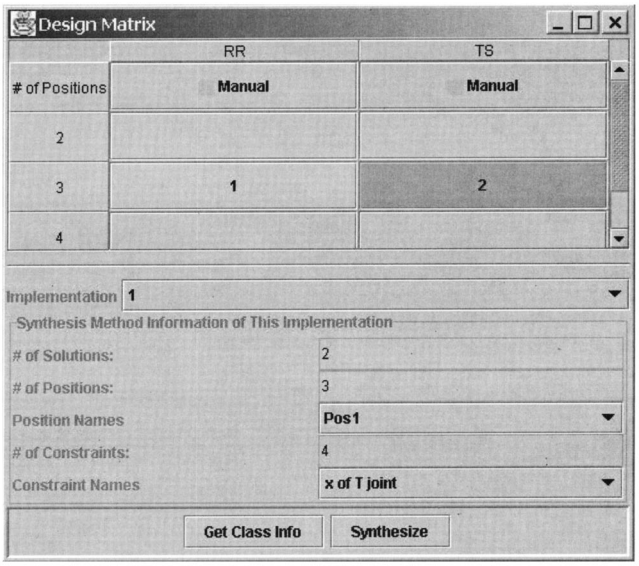

Fig. 3 Sample Design Matrix

to enter the design positions along with any additional constraints required by the synthesis method. It is customized for the particular linkage and synthesis routine. For the RR chain, it will only allow the user to specify three positions. When we design a TS chain, the names of the additional required constraints are provided along with the position information.

Once the design task is completely defined, the program executes the synthesis method and displays the solutions in the Solution Browser, shown in Fig. 5. At this point we have a number of options. We can change the entire design selection, we can revise the task, we an edit the linkage parameters, we can add an additional chain to the design, or we can generate a 3D model of the linkage solution.

In this case we want to add a second chain to the design. Pressing the Add button brings us back to the Design Matrix from which we select a TS chain synthesis method. We proceed through the Task Specifier and generate a set of TS linkage solutions. Now, the Solution Browser contains a panel for each set of chains. On the left we have the solutions for the RR linkage, and on the right we find the solutions for the TS linkage. We can now view

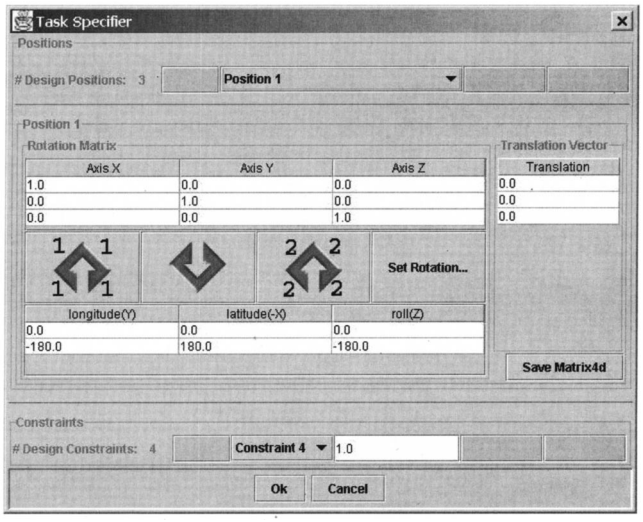

Fig. 4 Sample Task Specifier

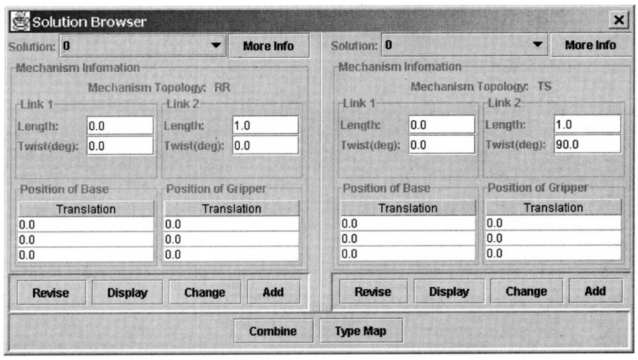

Fig. 5 Sample Solution Browser

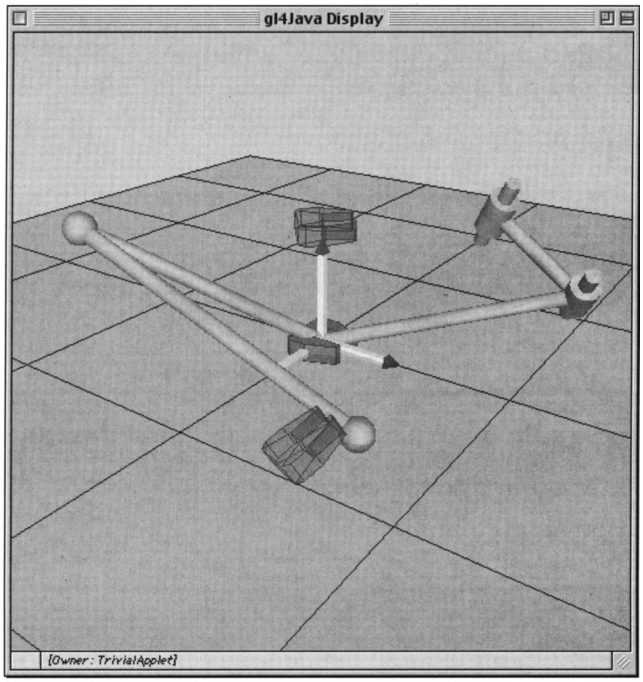

Fig. 6 Sample Mechanism Viewer

each solution individually, or combine the serial chains into a parallel linkage. In either case, the program takes the linkage data associated with the selected solution and displays an interactive 3D model of the design as shown in Fig. 6.

Depending on what kinematics routines are available, the designer can then interactively move the linkage through the various positions, or change the joint variables to animate the mechanism. It is important to note that any serial chains designed for the same positions can be assembled at those positions. This does not guarantee that the mechanism can move in a desirable way, thus motivating the need for post-synthesis evaluation.

Discussion

The current version of Synthetica is an applet that runs in a web browser or an applet viewer. The applet has been tested under Windows 95/98/NT, Mac OS 9, and Mac OS X. Cross platform compatibility is complicated by two factors; (i) different web browsers accommodate different Java Virtual Machines, and (ii) OpenGL links to Java currently are platform dependent. However, once the proper libraries are installed, the same Java code can be compiled to run on of the supported operating systems.

Once the Java applet loads, all computations are performed on the client machine. The applet is less than 100K in size, and loads and initializes in seconds. The majority of the applet functionality comes from the Java 2 packages and OpenGL libraries which reside on the client machine. The synthesis routines for the RR and TS mechanisms have analytical solutions and therefore execute almost instantly. The 3D graphics are initialized and generated directly within the applet itself and passed to the resident OpenGL implementation which can be hardware accelerated. In our case, the RRTS linkage was colored, textured, manipulated, and viewed in real-time.

Conclusions

This paper presents a new software structure for computer-aided spatial linkage design that is intended to support extension by user-collaborators. It is based on an abstraction of the function-to-form computer-aided design process for linkages that has evolved over the past decades. The complexity of spatial linkages requires a topology independent structure that allows user-collaborators to develop and integrate new serial chain primitives and parallel linkage analysis routines as needed. The structure consists of four primary modules: a Design Matrix, a Task Specifier, a Solution Browser, and a Linkage Viewer. These modules are supported by four packages, one of which, the mechanism package, can be tailored as needed to new spatial linkage topologies.

The resulting system will allow the designer to investigate a large number of spatial linkage topologies, as well as extend the system, if necessary, with new linkage definitions and synthesis routines. A prototype of this system defines spatial RR and TS open chains and closed chain topologies that can be constructed from them. It shows that OpenGL, GL4Java, and Java 2 combine to provided a convenient cross-platform development environment. Future research will seek to involve multiple design laboratories in a collaborative development effort for the computer-aided design of spatial linkages and robotic systems.

Acknowledgment

The authors gratefully acknowledge the support of the National Science Foundation and the University of California, Riverside.

References

[1] SyMech, software, http://www.symech.com/
[2] WATT, software, http://www.heron-technologies.com
[3] Kaufman, R. E., 1978, ''Mechanism Design by Computer,'' Mach. Des., October, pp. 94–100.
[4] Waldron, K. J., and Song, S. M., 1981, ''Theoretical and Numerical Improvements to an Interactive Linkage Design Program, RECSYN,'' Proc. of the Seventh Applied Mechanisms Conference, Kansas City, MO, Dec: 8.1–8.8.
[5] Erdman, A., and Gustafson, J., 1977, ''LINCAGES: Linkage INteractive Computer Analysis and Graphically Enhanced Synthesis Packages,'' Technical Report 77-DET-5, American Society of Mechanical Engineers. (See also: http://www.me.umn.edu/divisions/design/lincages/)
[6] Larochelle, P., Dooley, J., Murray, A., McCarthy J.M., 1993, ''SPHINX: Software for Synthesizing Spherical 4R Mechanisms,'' Proceedings of the 1993 NSF Design and Manufacturing Systems Conference, University of North Carolina at Charlotte, January 1993, **1**, pp. 607–611.
[7] Ruth, D. A., and McCarthy, J. M., 1997, ''SphinxPC: An Implementation of Four Position Synthesis for Planar and Spherical 4R Linkages,'' CD-ROM Proc. of the ASME DETC'97, paper no. DETC97/DAC-3860, Sept. 14–17, Sacramento, CA.
[8] Furlong, T. J., Vance, J. M., and Larochelle, P. M., 1998, ''Spherical Mechanism Synthesis in Virtual Reality,'' CD-ROM Proc. of the ASME DETC'98, Paper No. DETC98/DAC-5584, Sept. 13–16, Atlanta, GA.
[9] Larochelle, P. M., 1998, ''Spades: Software for Synthesizing Spatial 4C Linkages,'' CD-ROM Proc. of the ASME DETC'98, Paper No. DETC98/Mech-588w9, Sept. 13–16, Atlanta, GA.
[10] Suh, C.H., and Radcliffe, C. W., 1978, Kinematics and Mechanisms Design, John Wiley and Sons, New York.
[11] McCarthy, J. M., 2000, Geometric Design of Linkages, Springer-Verlag, New York.
[12] Campione, M., Walrath, K., and Huml, A., 2000, The Java(TM) Tutorial: A Short Course on the Basics, 3rd Ed., Addison-Wesley, Pub Co., San Francisco.
[13] Walrath, K., and Campione, M., 1999, The JFC Swing Tutorial: A Guide to Constructing GUI's, Addison-Wesley Pub Co., San Francisco.
[14] Woo, M., Neider, J., Davis, T., and Shreiner, D., 1999, OpenGL Programming Guide, Third Edition: The official Guide to Learning OpenGL, Version 1.2, Addison-Wesley, Menlo Park, CA.