

DETC02/MECH-34371

AN EXTENSIBLE JAVA APPLLET FOR SPATIAL LINKAGE SYNTHESIS

Haijun Su, Curtis Collins, and J. Michael McCarthy *
Robotics and Automation Laboratory
Department of Mechanical and Aerospace Engineering
University of California, Irvine
Irvine, California 92697

ABSTRACT

A spatial linkage is defined by a workpiece supported by several serial chains. The task of the linkage is defined by specifying rigid positions for the workpiece that approximate a desired workspace. Spatial serial chains can be synthesized and combined to form more complicated linkages. Because there are many serial chains available for this design process, we have developed a Java-based architecture for a computer-aided-design system for spatial linkages that is organized to allow incorporation of individually defined serial chain classes prepared by users/collaborators. Using the RR and TS chains as an example, we present the general features common to dimensional synthesis methodologies, and identify the types of analysis, synthesis, and evaluation routines that are required for displaying, animating, and designing spatial linkages formed from serial chain primitives. We present the basic components of the design system as well as specifications for the serial chain analysis and synthesis classes. An example is provided to demonstrate how the system can be used to perform three position synthesis of spatial RR, TS, and RRTS linkages.

INTRODUCTION

The focus of commercial computer-aided design (CAD) software has been on tools for shape representation and manipulation, assembly and detail drawing creation, and integration with computer-aided manufacturing (CAM) and analysis packages. This focus is clearly represented in high end systems such

as Dassault's CATIA, and PTC's Pro/ENGINEER. Mid-range modeling applications such as SolidWorks, AutoCAD, IronCad, and Vellum Solids have export capabilities or additional plug-in interfaces to CAM packages such as SurfCam or MasterCam, FEA packages such as Algor or Nastran, and dynamic simulation packages such as WorkingModel or ADAMS.

While this software is extremely powerful and versatile, it does not help automate the step in the design process described as the function-to-form transformation. This is the creative phase in the product development cycle where the designer synthesizes the geometric model of individual parts and assemblies to realize the particular function and satisfy the particular requirements.

One specific area of design research that has well developed tools for this function-to-form transformation is the theory of mechanism synthesis. Mechanism synthesis theory can be divided into methodologies for type and number synthesis, and dimensional synthesis (Sandor and Erdman 1997). Type synthesis allows the designer to match the input requirements to a space of possible mechanism types, such as gear systems, cam systems, or linkage systems. Once a mechanism type is identified, number synthesis can be used to generate the set of potential topological configurations available for further consideration. For example if an input specification is compatible with a planar linkage, number synthesis would generate a variety of linkage topologies with single and multiple loops. Once a mechanism topology is identified, the designer can use the tools of dimensional synthesis to identify the values of geometric parameters that satisfy the motion requirements.

*Address all correspondence to this author (email: jmmccart@uci.edu)

Linkage Design Software

For linkages, the dimensional synthesis problem is particularly well defined. The input specification, or task, describes the desired workspace or motion capabilities, and is typically represented by a set of discrete positions. From this description, the geometric constraints associated with the linkage architecture are resolved to determine its physical dimensions. This requires the solution of a complex set of nonlinear equations. These problems range in size and complexity depending on the number of specifications, and the type of mechanism being synthesized. Mechanism synthesis research well established and solution algorithms have been presented for a variety of mechanism architectures and input conditions (see for example, Suh and Radcliffe (1978) and McCarthy (2000)).

While there are numerous special purpose algorithms for synthesizing a variety of mechanism architectures, no general purpose design tool exists that integrates them. Each synthesis problem typically has different input specifications and different output requirements. This characteristic has led to the development of special-purpose software systems for linkage design. Research software has been developed to aide the design of planar (Erdman and Gustafson (1977), Ruth and McCarthy (1997)), spherical (Ruth and McCarthy (1997), Furlong, Vance and Larochelle (1998)), and spatial (Larochelle (1998), Kihonge, Vance and Larochelle (2001)) linkages. These packages have a number of features in common. They all provide an interface for specifying discrete workspace positions, they solve for the dimensions of a single mechanism architecture, and they provide some kind of graphical visualization of the resulting linkage designs. In general, they are developed for specific computing platforms and have no export capabilities.

Two commercial packages for planar linkage design are also available. SyMech software (2002) allows the designer to synthesize planar mechanisms from within Pro/E. Wireframe kinematic diagrams are generated that can be used to create parts. WATT software (2002) is a stand-alone package that can synthesize 4, 6 and 8 bar planar linkages and display them as stick models; it has very limited import and export capabilities.

Paper Overview

The goal of this paper is the definition of a software architecture that can incorporate a variety of mechanism synthesis routines into an interactive web-based design system. This architecture is modeled on the engineering design process, and provides a graphical user interface to integrate features common to linkage design systems. We begin with a discussion of the dimensional synthesis problem for RRTS linkages and highlight how it can serve as a template for our interactive design system. We then provide an overview of the functionality and implementation of Synthetica, an extensible Java applet for linkage synthesis. The flexibility and interactive capabilities of the

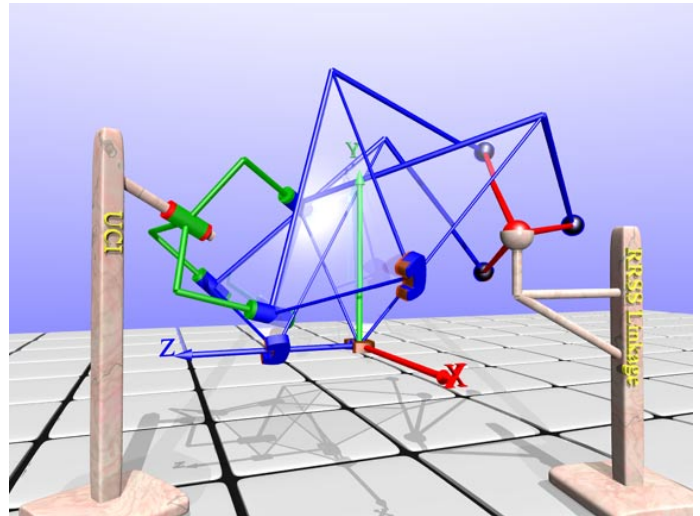


Figure 1. RRTS linkage designed to pass through three positions

applet arise from three key components: the Java programming language and platform specification, the OpenGL libraries and GL4Java bindings, and the architecture and interface specifications that we have developed. We then address how a user can utilize Synthetica as a design environment, or extend Synthetica to include new mechanism architectures and synthesis routines.

LINKAGE DESIGN METHODOLOGY

We have adopted a focused design methodology, that while not the most general approach, applies to a large variety of mechanism architectures. It is basically a procedure for dimensional synthesis with a finite number of rigid body positions. We consider a set of positions as a discrete representation of the desired workspace of a linkage. When formulating the design problem, we typically specify a set of positions and identify a particular type of linkage we wish to design. We then compare the number of positions to the number of free parameters in the design to determine what additional constraints are needed to fully specify a solution to the problem. In general, we synthesize open chains, which we call *linkage primitives*, and combine them to form closed chains. The primitives can have the same topology, or different. If they are designed to reach the same set of positions, the combined closed chain is guaranteed to be assembled at those positions.

Typically, serial chain synthesis routines lead to a parameterized set of solutions. We form a discrete set of closed chains using every combination of open chains from our solution sets. In order to determine what resulting closed chain satisfies our quantitative and qualitative specifications we need then analyze them for their properties. For the 4R planar or spherical linkage, for example, we can calculate each mechanism type and display

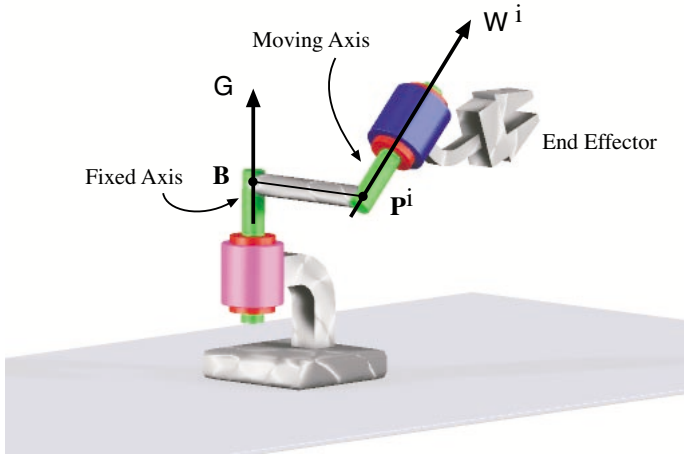


Figure 2. The RR open chain robot.

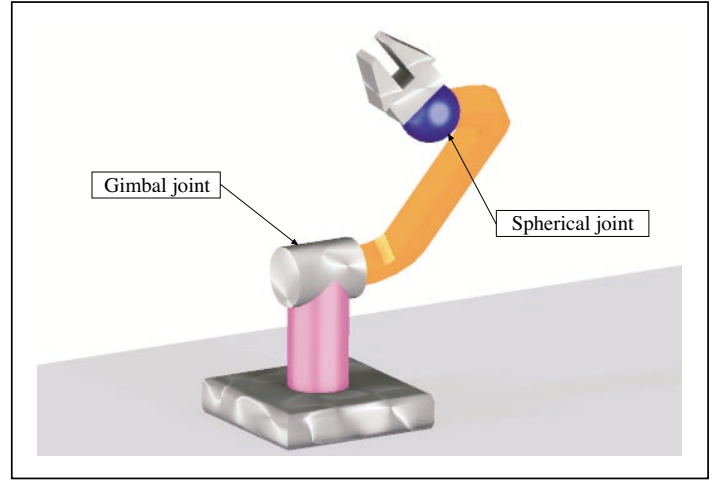


Figure 3. The TS open chain robot.

it graphically. Only certain mechanism types lead to “good” solutions. This post-synthesis evaluation formalism allows the designer to generate a large number of linkages that cannot otherwise be optimized at this time. In particular, it supports platform manipulator topologies where a set of serial chains act together on a common moving link or platform.

RR AND TS LINKAGE PRIMITIVES

Figure 1 shows an RRTS linkage designed to move through three positions. First, a pair of RR chains was synthesized. Next, a set of TS chains was synthesized. Then the chains were combined and evaluated to determine what solutions could move through the three positions smoothly. The details of implementing this design procedure provides insight into the requirements for linkage design software.

The Spatial RR Chain

The RR chain can be viewed as perhaps the simplest robot (Figure 2). Let \mathbf{G} and \mathbf{W}^1 be the directions of the fixed and moving axes, and \mathbf{P} and \mathbf{Q}^1 be points on these axes which locate them in space. The RR chain holds the angle and the distance between these axes constant. This yields the geometric constraints:

$$\begin{aligned} A : \mathbf{G} \cdot \mathbf{W}^i &= |\mathbf{G}| |\mathbf{W}^i| \cos \rho, \\ M : (\mathbf{P} \times \mathbf{G}) \cdot \mathbf{W}^i + \mathbf{G} \cdot (\mathbf{Q}^i \times \mathbf{W}^i) &= -r |\mathbf{W}^i| |\mathbf{G}| \sin \rho \quad (1) \end{aligned}$$

The design parameter vector for this linkage primitive consists of the components of the Plücker coordinate vectors $\mathbf{G} = (\mathbf{G}, \mathbf{P} \times \mathbf{G})$ and $\mathbf{W}^1 = (\mathbf{W}^1, \mathbf{Q}^1 \times \mathbf{W}^1)$.

For a discrete task specified by n homogeneous transforms

$[T_{1i}], i = 1, \dots, n$, we can rewrite these equations in the form:

$$\begin{aligned} A_{1i} : \mathbf{G}^T [A(\phi_{1i}) - I] \mathbf{W}^1 &= 0, \\ M_{1i} : \begin{Bmatrix} \mathbf{G} \\ \mathbf{P} \times \mathbf{G} \end{Bmatrix}^T \begin{bmatrix} D_{1i} A_{1i} & A_{1i} - I \\ A_{1i} - I & 0 \end{bmatrix} \begin{Bmatrix} \mathbf{W}^1 \\ \mathbf{Q}^1 \times \mathbf{W}^1 \end{Bmatrix} &= 0, \\ i &= 2, \dots, n. \end{aligned} \quad (2)$$

The RR chain also enforces normal conditions on the position of the points \mathbf{Q}^i and \mathbf{P} on the common normal between the two axes \mathbf{G} and \mathbf{W}^1 , which take the:

$$\mathbf{G} \cdot ([T_{1j}] \mathbf{Q}^1 - \mathbf{P}) = 0, \quad \mathbf{W}^1 \cdot (\mathbf{Q}^1 - [T_{1j}]^{-1} \mathbf{P}) = 0, \quad i = 1, \dots, n. \quad (3)$$

Mechanism synthesis theory shows that an analytical solution exists that which yields two RR chains that reach an arbitrarily specified set of three spatial positions (Tsai and Roth (1973)).

The Spatial TS Chain

The chain constructed from a gimbal joint and a spherical wrist, can reach an arbitrary set of seven spatial positions. The design parameter vector is $\mathbf{r} = (x, y, z, \lambda, \mu, \nu)$ is the set of coordinates for both center \mathbf{G} of the gimbal joint and the center of the wrist, \mathbf{W}^1 . The constraint that characterizes this chain is simply:

$$(\mathbf{W}^i - \mathbf{G}) \cdot (\mathbf{W}^i - \mathbf{G}) = R^2. \quad (4)$$

For a given set of task positions defined by the homogeneous transforms $[T_{1i}], i = 1, \dots, n$, we have

$$([T_{1i} \mathbf{W}^1 - \mathbf{G}) \cdot ([T_{1i} \mathbf{W}^1 - \mathbf{G}) = R^2, \quad i = 1, \dots, n. \quad (5)$$

Subtract the first equation from the remaining to obtain:

$$([T_{1i} - I]\mathbf{W}^1) \cdot \mathbf{G} - \frac{1}{2}([T_{1i}]\mathbf{W}^1 \cdot [T_{1i}]\mathbf{W}^1 - \mathbf{W}^1 \cdot \mathbf{W}^1) = 0, \quad i = 2, \dots, n. \quad (6)$$

Clearly, because there are six unknowns in the design parameter vector, six of these equations, which correspond to seven task positions, completely define the device. An analytical solution exists for these equations that yields as many as 20 TS chains to fit an arbitrary set of seven spatial task positions, Innocenti (1994).

Closed Chains Formed from RR and TS Primitives

Any two serial chains designed to pass through the same set of discrete positions can be assembled at each of the design positions. Since the RR and TS chains have different numbers of design parameters they require different sets of design constraints. For example, we can find a pair of RR dyads for the three position problem. Using the same three positions, we can design a set of TS chains, but in order to get a finite set of solutions, we need to specify up to four additional constraints. The RR and TS solutions can then be combined to form RRRR, RRTS, and 5TS closed chain linkages. If more detailed analysis is available for a given linkage topology, we can use it to evaluate the resulting designs.

For example, the movement of the RRTS mechanism can be characterized by the ranges of motion of each R joint. Each R joint can have 1, 2 or 4 distinct ranges of motion. Finding these angular limits requires the determination of the real roots of a quartic polynomial in the link dimensions. Symbolic conditions for the number of real and complex roots of a quartic polynomial can be derived and used to classify the general motion of the RRTS linkage. A two dimensional set of RRTS linkage solutions can be displayed as a color map based on the type classification of each design. We will use this Type Map as a qualitative evaluation tool for sorting RRTS linkage designs. See Su and McCarthy (2001) for details.

It is especially important in spatial mechanism design for the designer to be able to display and animate the potential designs. This requires the forward and inverse kinematics routines. For simple serial chain primitives, these routines are easy to implement. For more complex serial chains, we can use a general root solver to obtain inverse kinematics solutions. For closed chains, the problem becomes more difficult and we typically seek a special purpose algorithm that solves the closed chain constraints.

SOFTWARE FUNCTIONALITY

The functionality of the Synthetica Applet is based on our model of the linkage design process. We confine the initial scope

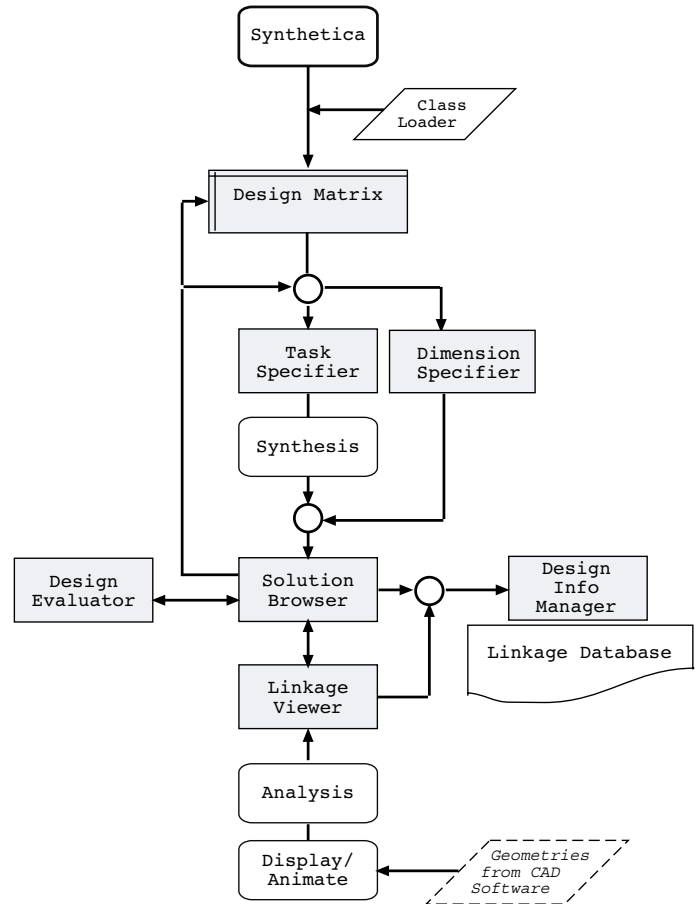


Figure 4. Software Architecture

of the software to the specification and synthesis of serial chain mechanisms (planar, spherical and spatial) and mechanisms with a platform topology (that is mechanisms that have a single base and single moving platform connected by serial subchains). *The key feature of the software architecture is that it can support the integration of multiple mechanism topologies and multiple synthesis routines.*

The key components and flow of information for Synthetica are shown in Figure 4. In order to provide flexibility and extensibility, we have made each key stage in the process its own software module. The first version of the software is in the form of a Java applet which is designed to run under a web browser or applet viewer on Windows, Mac OS, and Linux operating systems. The GUI elements, 3D graphics capabilities, dynamic class loading, and data file management are all implemented using standard Java packages, third-party packages, and custom designed packages.

The first key GUI element is called the Design Matrix. This is a tabular representation of the mechanism topologies and synthesis routines available for specifying and creating new mecha-

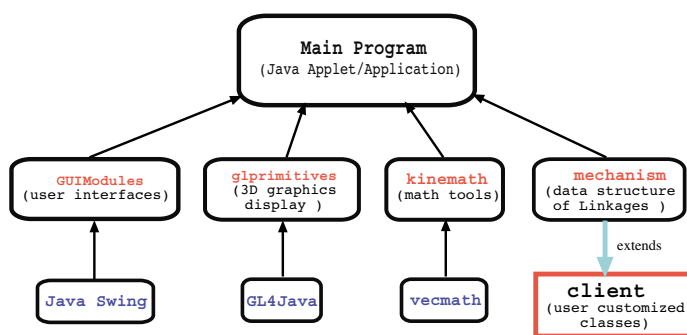


Figure 5. Synthetica Package Organization

nism designs. This table is dynamically constructed from information about the available classes. At this point the user can also load customized classes and include them in the selection matrix. The purpose of this module is to allow the user to direct the design from either a task oriented perspective, or a linkage topology perspective. From this point, the designer can either manually input the dimensions of the linkage under consideration, or specify the parameters of the task required by the selected synthesis method.

Once a linkage topology and synthesis method is selected, a Task Specifier module is dynamically generated to allow the user to input the required position information, and any additional constraints required by the synthesis method. Once this information is provided, the synthesis method is called, and a list of solutions is generated and made available in the Solution Browser. At this point the user can change the selected linkage/synthesis routine, modify the task and constraints, or directly modify the linkage dimensions. In addition, if an evaluation routine is available for the linkage, the designer can select to use it to help rank and classify the solutions.

Once a particular solution is selected, a 3D model of the linkage can be generated, displayed, and manipulated interactively. In the first version of the software, the designer can change the material, color, and texture of the linkage components. From here, the user can select a different linkage to view, or continue to refine the task.

SYNTHETICA PACKAGES

Synthetica has four major components organized into Java packages as shown in Figure 5. The main program integrates and coordinates the flow of information using the four underlying packages described briefly as follows.

GUIModules The package `GUIModules` provides classes that generate the interfaces for user input. The Swing library is used extensively in this package.

glprimitives The package `glprimitives` provides the 3D graphics engine for the main program. It uses `GL4Java`, a

JNI encapsulation of the OpenGL graphics library, to generate and display mechanism models in real-time.

kinemath The package `kinemath` 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.

mechanism The package `mechanism` provides the linkage data kernel. It defines necessary data classes for extending a new mechanism. Users who want to implement a new mechanism can use this package as a basis to build a new linkage. See Figure 6 for the components of the package `mechanism`.

To accommodate the variety of mechanisms and synthesis routines, we taken advantage of Java's built-in capabilities to dynamically inspect class contents at runtime. This feature is represented in Figure 4 by the Class Loader. When the Synthetica applet loads, the `mechanism` and synthesis classes are inspected to determine the mechanism topology, and the 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. 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.

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 package `mechanism` 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 `mechanism` package hierarchy shown in Figure 6. A mechanism is defined by serial chains composed of links and joints. A design tasks is defined by a set of discrete positions and a list of additional constraints. Associated with each mechanism is a set of geometric objects that define the base, the gripper, and the link and joint geometries. The `SerialMechanism` and a `ParallelMechanism` classes are designed to be extended and contain useful methods for defining and implementing new linkage topologies.

We have also defined four Java interfaces. The `ForwardKinematics`, and `InverseKinematics` interfaces specify the function prototype required for mechanism position analysis. The `Synthesizable` interface specifies a set of functions 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.

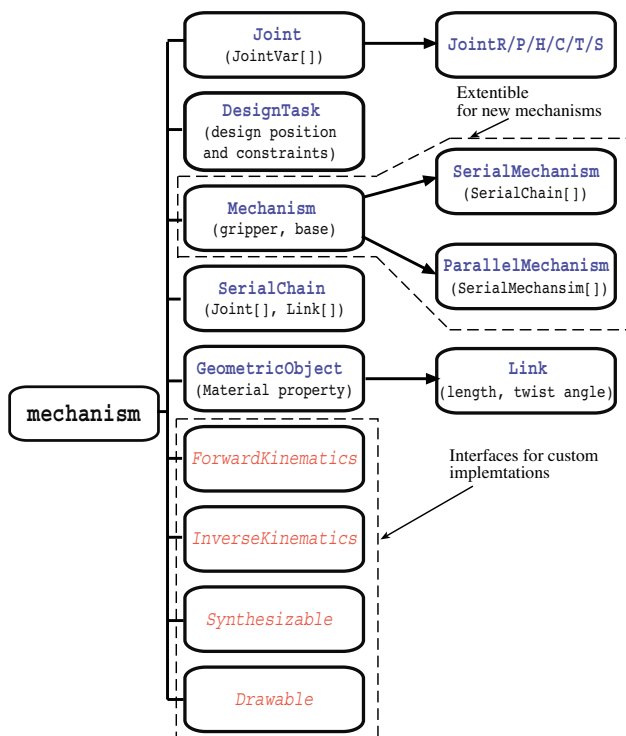


Figure 6. Mechanism Package Organization

Implementing Mechanism and Synthesis Classes

Basically, Synthetica serves two roles. It can run as a standalone application/applet which causal users can use it 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 runtime, the program examines the contents of user defined 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, for example, `SerialMechanism`.
2. Implement any interface methods you require, such as `Synthesizable`, `ForwardKinematics`, 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 ClassLoader.
5. The information found in the new class will be mapped into the Design Matrix and made available for use.

EXAMPLE DESIGN SESSION

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 Figure 7. 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 how many synthesis methods are available for that particular combinations of positions and linkage primitive. In this case, we 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 allow 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 Synthesis button leads to the generation of the Task Specifier. This window, shown in Figure 8. The Task Specifier window allows us to enter the design positions along with any additional constraints required by the synthesis method. It is customized for the particular linkage/synthesis routine. For the RR chain, it will only allow the user to specify three positions. When we design an 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 generates a list of solutions and displays them in the Solution Browser shown in Figure 9. At this point we have a number of options. We can change the entire design selection, we can revise the task, we can 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 an 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 each solution individually, or combine the serial chains into a parallel mechanism. 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 Figure 10.

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

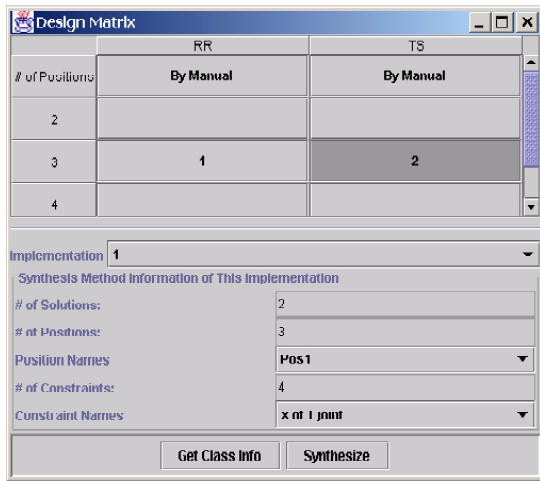


Figure 7. Sample Design Matrix

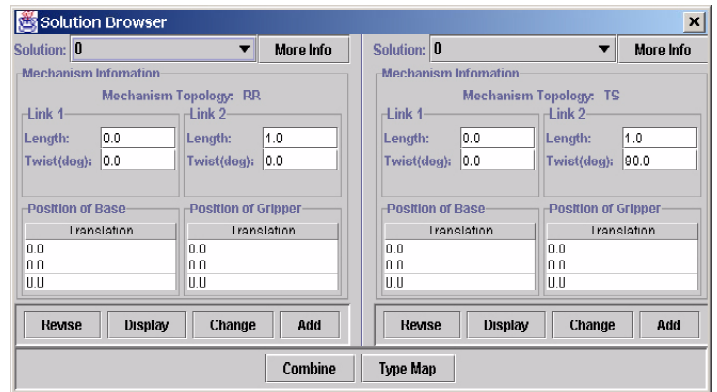


Figure 9. Sample Solution Browser

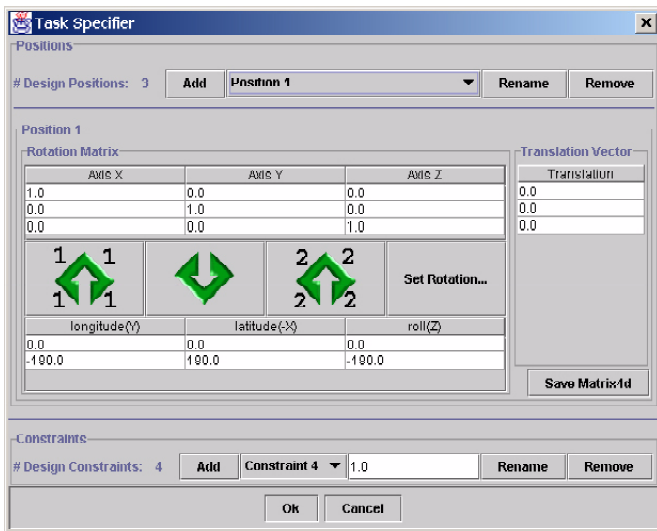


Figure 8. Sample Task Specifier

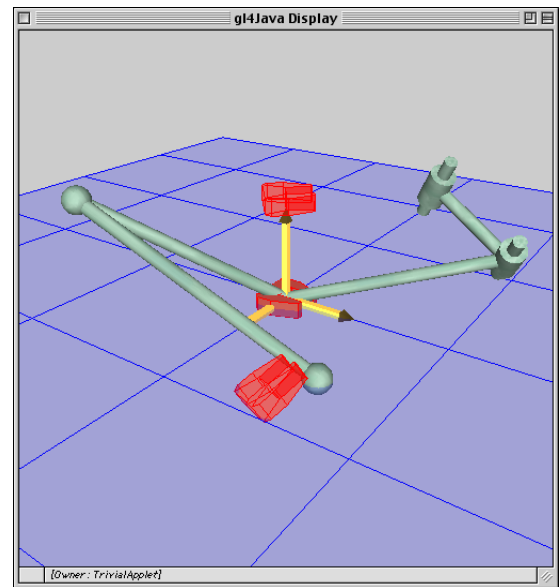


Figure 10. Sample Mechanism Viewer

not guarantee that the mechanism can move in a desirable way, thus motivating the need for post-synthesis evaluation.

DISCUSSION

The current version of the design software is an applet that runs in a web browser or an applet viewer application. The applet has been tested under Windows 95/98/NT, Mac OS 9, and Mac OS X. True cross platform compatibility is complicated by the fact that different web browsers accommodate different Java Virtual Machines, and the OpenGL links to Java are platform dependent. However, with once the proper libraries are installed, the same java code can be compiled to run on any of the supported operating systems.

The current applet prototype has a simple display of mechanism solutions. We are working on adding sorting and searching capabilities to this environment. In addition, we are planning an evaluator interface specification for automating the search procedure when quantitative performance metrics are available. Once a mechanism is identified and selected, the kinematic data can easily be transferred to other analysis tools, such as a static analysis module or interference analysis module. The the purpose of the evaluator interface is to provide analysis tools that are either not available in commercial simulation packages, or could be particularly useful for initial design evaluations.

CONCLUSION

This paper presents a software architecture for the functional design of linkages consisting of four primary modules: a Design Matrix, a Task specifier, a Solution browser, and a Mechanism viewer. These modules are supported by four packages. Mechanism definitions and synthesis routines are implemented using the specifications found in the `mechanism` package. After these routines are written and compiled, their properties are automatically identified and integrated into the Design Matrix. The system supports open chain primitives and platform topologies and allows the designer to synthesize any number of different open chains for the same fundamental positions. This approach allows the user to investigate a large number of spatial mechanism topologies and extend the system with new mechanism definitions and synthesis routines. The first prototype system defines spatial RR and TS open chains and closed chain topologies that can be constructed from them. We show that OpenGL, GL4Java, and Java2 combine to provide 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.

ACKNOWLEDGEMENT

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

REFERENCES

Campione, M., Walrath, K., and Huml, A., *The Java(TM) Tutorial: A Short Course on the Basics, 3rd Ed.*, Addison-Wesley, Pub Co., San Francisco, 2000. (See also <http://java.sun.com/docs/books/tutorial/index.html>)

Erdman, A., and Gustafson, J., "LINCAGES: Linkage INTERactive Computer Analysis and Graphically Enhanced Synthesis Packages," Technical Report 77-DET-5, American Society of Mechanical Engineers, 1977. (See also: <http://www.me.umn.edu/divisions/design/lincages/>)

Furlong, T.J., Vance, J.M., and Larochelle, P.M., "Spherical Mechanism Synthesis in Virtual Reality," *CD-ROM Proc. of the ASME DETC'98*, paper no. DETC98/DAC-5584, Sept. 13-16, Atlanta, GA, 1998.

GL4Java Home Page, <http://www.jausoft.com/gl4java/>

Innocenti, C., "Polynomial Solution of the Spatial Burmester Problem," *Mechanism Synthesis and Analysis*, DE-Vol. 70, ASME DETC94, Minneapolis, MN, 1994.

Kihonge, J., Vance, J., and Larochelle, P., "Spatial mechanism design in virtual reality with networking," *Proc. ASME 2001 Design Engineering Technical Conferences*, Pittsburgh PA, Sept 9-12, 2001.

Larochelle, P.M., "Spades: Software for Synthesizing Spatial 4C Linkages," *CD-ROM Proc. of the ASME DETC'98*, paper no. DETC98/Mech-5889, Sept. 13-16, Atlanta, GA, 1998.

McCarthy, J. M., *Geometric Design of Linkages*. Springer-Verlag, New York, 2000.

Perez, A, and McCarthy, J.M., "Dimensional synthesis of spatial RR robots," *Advances in Robot Kinematics*, Piran-Portoroz, Slovenia, 2000.

Perez, A, and McCarthy, J.M., "Dimensional synthesis of Bennett linkages," *Proc. 2000 ASME Design Engineering Technical Conferences*, Baltimore, MD, Sept. 10-13, 2000.

Ruth, D. A., and McCarthy, J. M., "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, 1997.

Sandor, G. N., and Erdman, A. G., *Advanced Mechanism Design: Analysis and Synthesis, Vol. 2*. Prentice-Hall, Englewood Cliffs, NJ, 1997.

Su, H., and McCarthy, J. M., "Classification of Designs for RRSS Linkages," *Proc. 2001 ASME Design Engineering Technical Conferences*, Sept. 9-12, Pittsburgh, PA, 2001.

Suh, C.H., and Radcliffe, C.W., *Kinematics and Mechanisms Design*. John Wiley and Sons, New York, 1978.

SyMech, software, <http://www.symech.com/>

Tsai, L.W., and Roth, B., "A Note on the Design of Revolute-Revolute Cranks," *Mechanism and Machine Theory*, Vol. 8, pp. 23-31, 1973.

Tsai, L. W., *Mechanism Design: Enumeration of Kinematic Structures According to Function*, CRC Press, New York, 2001.

Walrath, K., and Campione, M., *The JFC Swint Tutorial: A Guide to Constructing GUI's*, Addison-Wesley Pub Co., San Francisco, 1999. (See also <http://java.sun.com/docs/books/tutorial/index.html>)

WATT, software, <http://www.heron-technologies.com>

Woo, M., Neider, J., Davis, T., and Shreiner, D., *OpenGL Programming Guide, Third Edition: The official Guide to Learning OpenGL, Version 1.2*, Addison-Wesley, Menlo Park, CA, 1999.