Int. J. Human–Computer Studies (1997) **46,** 773–788

FixtureNet: interactive computer-aided design via the World Wide Web

RICK WAGNER AND GIUSEPPE CASTANOTTO

CS Department, University of Southern California, CA, USA. email: rwagner@usc.edu

KEN GOLDBERG

IEOR Department, University of California at Berkeley, CA, USA. email: goldberg@tinguely.ieor.berkeley.edu

The Internet offers tremendous potential for rapid development of mechanical products to meet global competition. In the past several years, a variety of geometric algorithms have been developed to evaluate computer-aided design (CAD) models with respect to manufacturing properties such as feedability, fixturability, assemblability, etc. Unfortunately, most of these algorithms are tailored to a particular CAD system and format and so have not been widely tested by industry. The World Wide Web may offer a solution: its simple interface language offers a *de facto* standard for the exchange of geometric data with industry and research groups. In this paper, we describe a feasibility study for such an interactive system, which can be tested directly at `http://teamster.usc.edu/fixture/` © 1997 Academic Press Limited

1. Motivation

The Internet has launched a revolution in the means of worldwide data transfer and resource sharing. This has fueled the emergence of many commercial services and products that are marketed and accessed over the Net. World Wide Web (WWW) technology also offers potential for the design and manufacture of new products as suggested in sites such as:

```
http://acorn.eit.com/acorn-info.html
http://www.autodesk.com/products/datapub/mechlibr/mechlib.htm
```

Digital communication over the Internet offers advantages in terms of speed, efficiency and automation. New geometric algorithms for design, simulation and manufacture have been developed and reported in the research literature. Unfortunately, the impact of these advances on the manufacturing community has been limited despite a well-documented need for improved communication during product development. At the same time, researchers rarely have access to each others' algorithms as implementations are difficult to port from one platform to another.

The WWW may offer a solution as it provides a standard that works with different hardware platforms and with different software data standards. However, the design of interactive computer-aided design CAD for the WWW requires systems that have the following characteristics.

1071-5819/97/060773 + 16$25.00/0/hc960107

- Minimize data transfer.
- Can be understood solely from on-line documentation.
- Synchronize service requests from multiple simultaneous users.

In this paper we describe one model for interactive CAD via the WWW that meets these criteria and can benefit both industry and research. As a feasibility study, we draw on our recent study of fixtures. A fixture is a device for holding parts during machining, inspection or assembly. *FixtureNet* is an interactive fixture design service on the WWW that uses the Brost–Goldberg (Brost & Goldberg, 1994) algorithm to consider systematically all possible modular fixtures for a given part: `http://teamster.usc.edu/fixture/`.

2. Related work

Commercially available "modular fixturing" systems typically include a square lattice of tapped and doweled holes with precise spacing and an assortment of precision locating and clamping elements that can be rigidly attached to the lattice using hardened bushings or expanding mandrels. Ordinarily, human expertise is required to synthesize a suitable arrangement of these elements to hold a given part (Hoffman, 1987). Besides being time-consuming, if the set of alternatives is not systematically explored, the designer may fail to find an acceptable fixture or may settle upon a sub-optimal fixture. Our fixture design algorithm often generates counterintuitive solutions that may be overlooked by even an experienced machinist, much as chess machines can play moves that look naive at first glance but lead the experienced chess player to explore new variations.

Work related to ours includes user-interface design, software testing and CAD. The WWW provides an unprecedented opportunity for a large number of researchers to test experimental computer programs. Often the designers of a research algorithm cannot anticipate the kinds of inputs to which a variety of users in related disciplines might subject the program. The automated design of fixtures is a challenging research area. The earliest work in this area is related to the necessary conditions for holding parts (work pieces) securely.

2.1. FORM CLOSURE

Reuleaux (1963) first described form closure which captures the intuitive requirement of a fixture: a part is held in form closure if it can resist arbitrary forces and torques. Lakshminarayana (1978) showed that seven frictionless contacts are necessary to hold a three-dimensional part in form closure; Mishra, Schwartz and Sharir (1987) showed that seven frictionless contacts are also sufficient.

Goldman and Tucker (1956), in a purely mathematical paper on linear algebra, described the necessary and sufficient conditions for positively spanning an n-dimensional Euclidean space, which coincidentally describes the necessary and sufficient conditions for form closure. Wagner, Zhuang and Goldberg (1995) make use of a simplification of that proof in validating an intuitive form-closure test.

2.2. MODULAR FIXTURING

Hoffman's (1987) text provides an overview of conventional practice with modular fixtures. Research on modular fixturing includes basic questions about the existence of solutions (Zhuang, Goldberg & Wong, 1994), practical extensions to three-dimensions (Wagner, Zhuang & Goldberg, 1995) and the problem of fixture loading (Yu & Goldberg, 1995). We are also studying how the model can be extended to curved parts (Wallack & Canny, 1994).

Asada and By (1985) describe an automatic fixture reconfiguration system using a robot manipulator and a CAD system to provide a systematic method for designing fixtures. They also provide an analytic test for form closure and suggest how contact points might be applied, but they did not consider how a restricted set of modular elements could be used to reach those points. They call fixture synthesis "designing a fixture layout", which is in keeping with the mechanical drawing practice of calling a drawing that gives the locations of parts a "layout" drawing. They develop analytic tools for designing fixture layouts using a set of hardware primitives implemented at MIT. They also considered loading and unloading of their fixtures.

Wolter and Trinkle (1994) describe a non-modular fixture synthesis that uses analysis of frictionless stability in "Automatic Selection of Fixture Points for Frictionless Assemblies". This is an impressive paper because it applies to both two- and three-dimensional fixtures, but it is "non-modular" because the fixture points selected are from a continuum in space and not from a discrete set of locations. In their problem, frictionless elements of assemblies need to be held together by a fixture. They analyse fixtures for "stability" in terms of virtual work. Their fixture synthesis algorithm uses a "shotgun" approach: they scatter fixels about the assembly and solve a linear program to minimize contact forces at the fixels by having fixel location on the part boundary be a system variable. Fixels that have reaction forces of zero get discarded. This is an effective approach, but it is not guaranteed to find an optimal solution. Also, it is not applicable to modular fixturing hardware sets as currently available.

Brost and Goldberg (1994) have demonstrated a complete algorithm for synthesizing two-dimensional fixtures that forms the basis of *FixtureNet*. Since this paper, other papers regarding planar modular fixturing have appeared, including "Planning for Modular and Hybrid Fixtures" by Wallack and Canny (1994), which describes a vise-like fixture with four cylindrical locators. Clamping motion is provided by a translating lattice, and they give a complete algorithm to evaluate all possible configurations.

Until recently there have been few papers describing modular fixture synthesis algorithms in three-dimensions. However, Wagner *et al.* (1995) have described a new modular strut hardware set and a complete algorithm for automated fixture synthesis with these primitives, which is the subject of an extension to *FixtureNet* in Section 6.

Recently, Ponce, Burdick and Rimon (1995) have described "immobilizing" grasps, and have proposed their possible application in fixturing in both two and three dimensions. Immobilizing fixtures require only three contact points in the plane and four contacts in three dimensions. The practical application of immobilizing fixtures is somewhat limited, however, in that when they are evaluated in terms of the quality metrics generally applied to form closure fixtures, they will be ranked below fixtures with form closure. This is because an immobilizing fixture generates very large reactions (assuming no friction and rigid parts and fixture elements) with the application of a moment load. Immobilizing fixtures may be a very attractive alternative for light-duty applications with friction.

2.3. RELATED WEB SITES

Related web sites include the following. All of these can be reached from our *FixtureNet* "Related Links" page.

- University of Minnesota's *Geometry Center* has a great collection of interactive geometric algorithms:

  `http://www.geom.umn.edu:80/apps/`

- David Eppstein's *Discrete and Computational Geometry* page:

  `http://www.ics.uci.edu/ ~ eppstein/geom.html`

- Jeff Erickson has been maintaining a small collection of computational geometry World Wide Web pages:

  `http://www.cs.duke.edu/ ~ jeffe/compgeom/`

- *Prof. Antonio Bicchi's Non-Holonomic Motion Planning Site* at University of Pisa allows users to define obstacles for path planning and even sets up a standard for others to submit algorithms for comparison:

  `http://www.piaggio.ccii.unipi.it/prova/motion.html`

- *The AutomationNET!* In December 1996, *PC Computing* as one of the best engineering web sites:

  `http://www.AutomationNET.com/`

3. The modular fixturing algorithm

Brost and Goldberg (1995) considered a class of modular fixtures that prevent a part from translating and rotating in the plane, based on three round locators, each centered on a lattice point, and one translating clamp that must be attached to the lattice via a pair of unit-spaced holes, thus allowing contact at a variable distance along the principal axes of the lattice (see Figure 1). Brost and Goldberg gave an algorithm that accepts part geometry as input and synthesizes the possibly empty set of all fixture designs in this class that achieve form closure for the given part. If the part has n edges and its maximal diameter (in units of lattice spacing) is d, the algorithm runs in time $\mathbf{O}(n^5d^5)$. The paper describes extensive experiments run on a Lisp machine. This is the first fixture synthesis algorithm that is complete in the sense that it is guaranteed to find an admissible fixture if one exists.

3.1. QUALITY METRIC

The *FixtureNet* algorithm is *complete* in the sense that it will find all solutions to a fixturing problem if any exist and report failure if no solutions exist. *FixtureNet* often

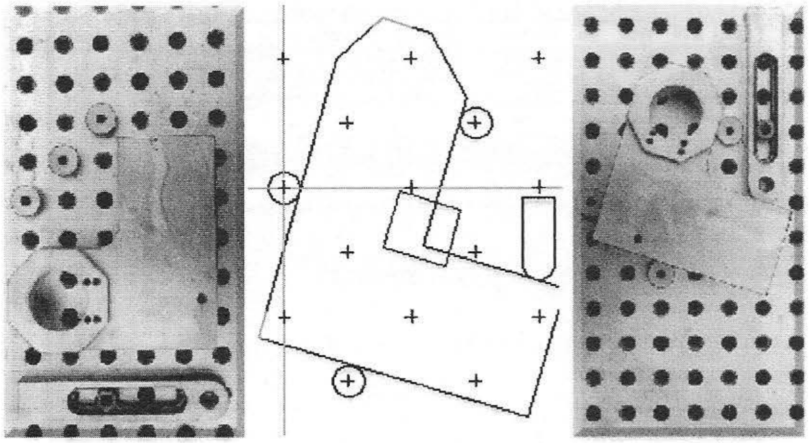


FIGURE 1. Modular fixturing system. Left: three circular fixture elements (fixels) and a sliding clamp are shown on a regular lattice of mounting holes. The L-shaped bracket is a typical industrial part. Center: a geometric fixture found by *FixtureNet*. Right: the corresponding fixture and L-shaped bracket.

finds a large number of solutions. The user generally cares only about a subset of the *best* solutions. The issue of measuring the quality of a fixture is a research topic in its own right. *FixtureNet* provides two very different quality metrics, a default metric that minimizes part interface reactions under a "general" load and an optional metric that minimizes locator interface reactions for a fixed-clamp load. The "best" fixtures selected under these two metrics are usually quite different sets.

4. *FixtureNet* interface

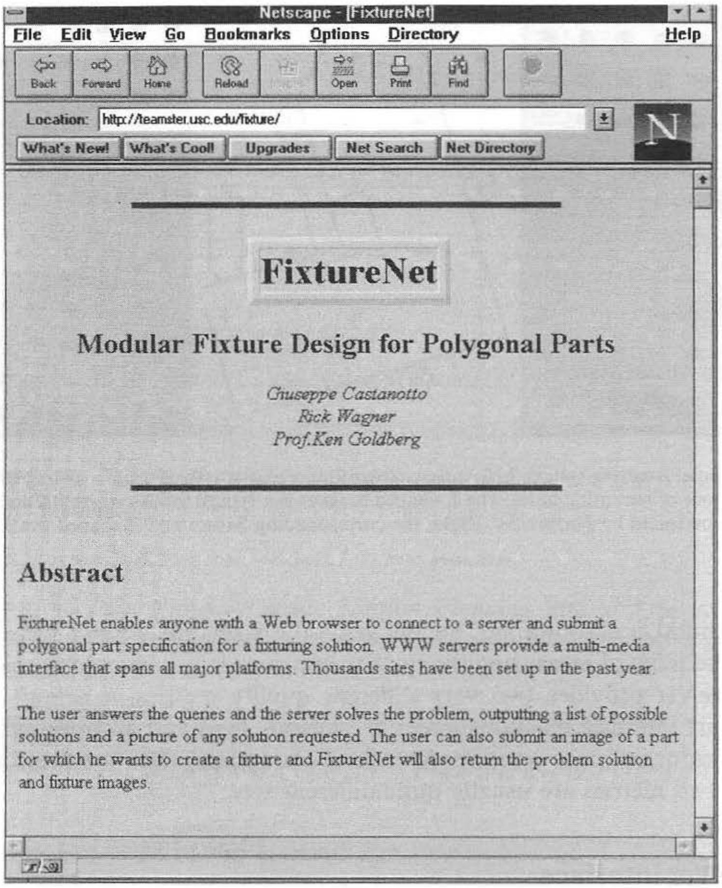
Recently, we developed a WWW server to provide modular fixture design alternatives to engineering users around the world. We refer to this service as *FixtureNet*. The first version is based on the Brost–Goldberg algorithm as described above; subsequent versions will incorporate the three-dimensional extensions described in Wagner *et al.* (1995).

We invite readers to test the software by going to the *FixtureNet* home page:

```
http://teamster.usc.edu/fixture/
```

The home page is shown in Figure 2 and includes a graphical introduction to the algorithm, many examples, and links to related work and papers. It also includes an on-line manual detailing how to use the *FixtureNet* service (and documentation on the *FixtureNet* implementation itself, including system architecture, etc.).

When the user clicks on the FIXTURE SERVICE link, he or she is given an option for the input data style and then encounters the appropriate form to describe the user's polygonal part, including a graphical interface to permit users to point and click to define parts (using an "ISMAP" that is a feature of the HTML language) (Figure 3). All processing is done on the servers at the USC campus, not on the client machine as with


Netscape - [FixtureNet]
File Edit View Go Bookmarks Options Directory Help
Back Forward Home Reload Open Print Find
Location: http://teamster.usc.edu/fixture/
What's New! What's Cool! Upgrades Net Search Net Directory

FixtureNet

Modular Fixture Design for Polygonal Parts

Giuseppe Castanotto
Rick Wagner
Prof.Ken Goldberg

Abstract

FixtureNet enables anyone with a Web browser to connect to a server and submit a polygonal part specification for a fixturing solution. WWW servers provide a multi-media interface that spans all major platforms. Thousands sites have been set up in the past year.

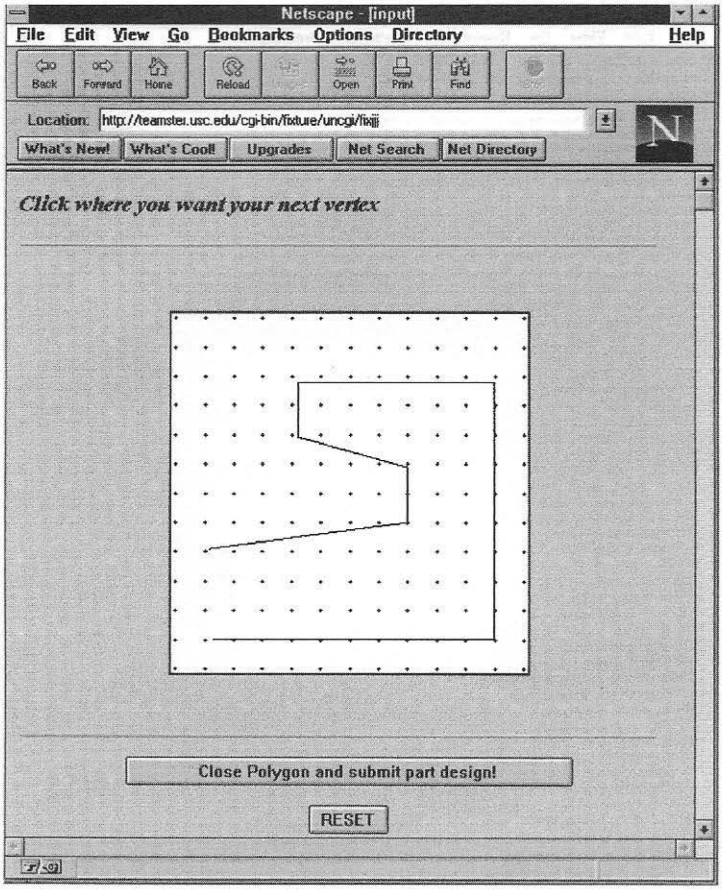
The user answers the queries and the server solves the problem, outputting a list of possible solutions and a picture of any solution requested. The user can also submit an image of a part for which he wants to create a fixture and FixtureNet will also return the problem solution and fixture images.

FIGURE 2. The *FixtureNet* home page.

Java applications. The data is then submitted by clicking on a SUBMIT button and the server responds with a line drawing of the user's part (Figure 4).

After pressing CONTINUE, the system returns with an estimate of the approximate time required to run the algorithm for the user's part (Figure 5). The time estimate is computed by sampling the server's current CPU processing power (a function of the current load) and evaluating the complexity of the computation by analysing the part size and number of edges. The algorithm currently runs on a 25 MHz 486 machine with limited (4 Mbyte) memory and can require several minutes to compute a set of solutions.† One issue is that the system currently permits only one user at a time; hence users must queue and long run times can become a problem (see Section 7). The HTML server needs to be responsive to the client browser requests to avoid browser time-out, so the server sends back a place holder (teaser), an important innovation in this project.

† If the estimated time is over 7 mins, the user is encouraged to select another part.


Netscape - [input]
File Edit View Go Bookmarks Options Directory Help
Back Forward Home Reload Images Open Print Find Stop
Location: http://teamster.usc.edu/cgi-bin/fixture/uncgi/fixjjj
What's New! What's Cool! Upgrades Net Search Net Directory
Close Polygon and submit part design!
RESET

Click where you want your next vertex

FIGURE 3. The user may use the mouse to click vertex points to draw a new part. Here the hook-shaped part is nearly completed.

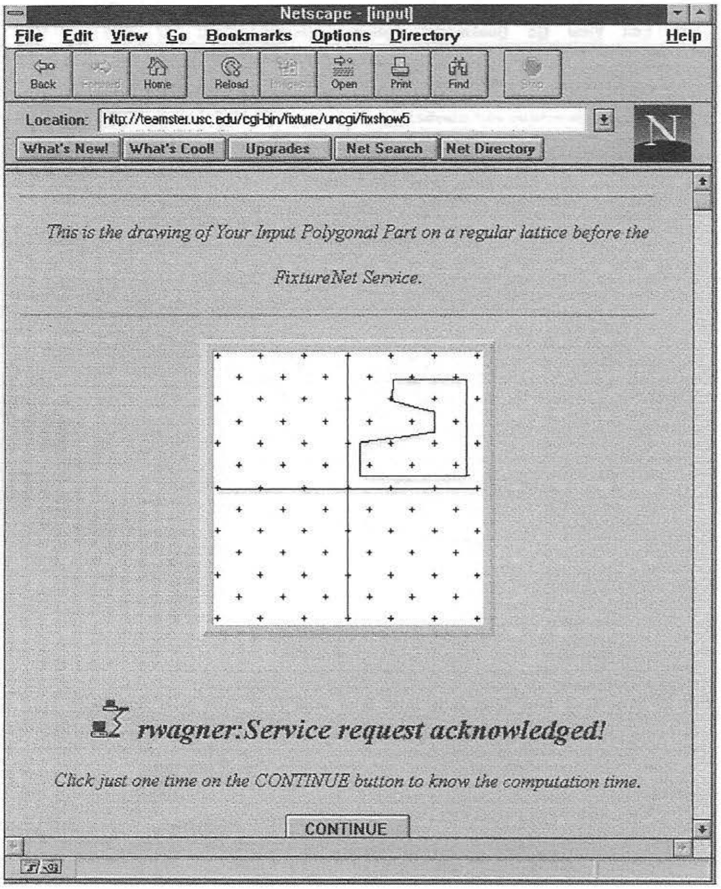
After the estimated time, the user presses the CONTINUE button and the server either asks for more time (with an appropriate estimate), or returns images of the user's part in each of the "best" four solutions (Figure 6) (modular fixture configurations) ranked on the basis of a quality metric related to the fixture's ability to resist a combination of forces in the plane of the fixture and moments about a normal to the lattice plane.

The user can also request other solutions and is given an ID number so that he or she can view the solutions any time within 24 h (after which the solutions are deleted and must be regenerated).

5. Implementation

5.1. ARCHITECTURE

The *FixtureNet* system architecture is shown in Figure 7 and is also illustrated on-line. *FixtureNet* involves two machines connected via an Ethernet local area network (LAN).


Netscape - [input]
File Edit View Go Bookmarks Options Directory Help
Back Forward Home Reload Images Open Print Find Stop
Location: http://teamster.usc.edu/cgi-bin/fixture/uncgi/fixshow5
What's New! What's Cool! Upgrades Net Search Net Directory
CONTINUE

This is the drawing of Your Input Polygonal Part on a regular lattice before the FixtureNet Service.

rwagner:Service request acknowledged!

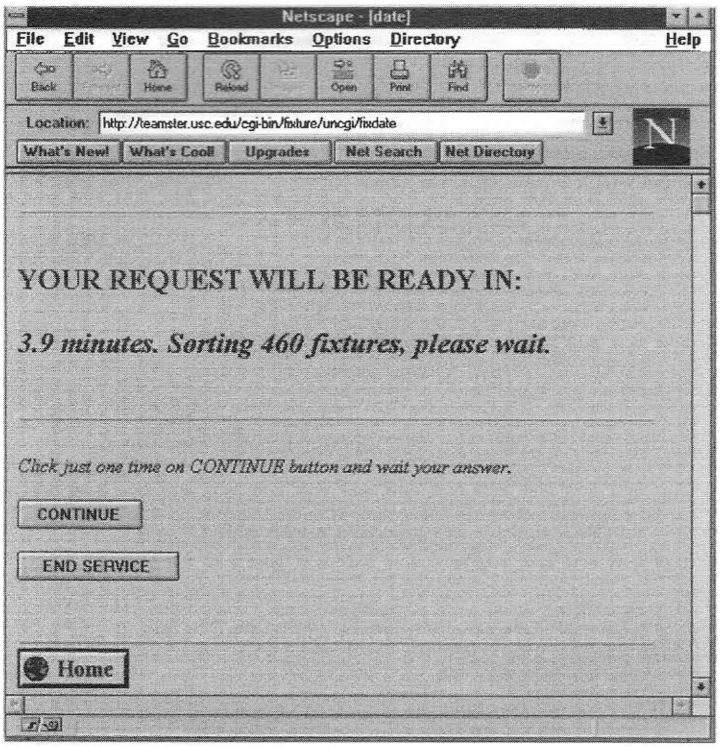
Click just one time on the CONTINUE button to know the computation time.

FIGURE 4. The input part is shown in relation to the fixel grid (lattice).

The LAN is in turn connected via the USC gateway to the Internet. Machine A (host *Teaser* in Figure 8) runs the Brost–Goldberg algorithm. It is a dedicated 486DX 25 MHz PC [industry standard architecture (ISA)] running the Microsoft Windows 3.1 environment on MS-DOS. Machine B (host *Teamster* in Figure 9) is a WWW server and uses custom cgi-bin routines to send requests to the fixture server running on Machine A. Machine B is a 90 MHz Pentium^{TM} PC (ISA, industry standard architecture) running the Linux operating system. Machine A communicates with B via Ethernet^{TM} using the TCP/IP protocol.

The user accessses the *Teamster* HTTP server (a Linux client with respect to the fixture server on *Teaser*) via the Internet (with Netscape or some other WWW browser application). The ability of the user to describe the part to be fixtured by drawing with the mouse is a convenient feature.† The mechanism behind image maps is straightforward: the image that we wish to use (in our case a square where the user can draw his

† But for serious use with real parts we emphasize that the user can submit a descriptive file via FTP.


Netscape - [date]
File Edit View Go Bookmarks Options Directory Help
Back Forward Home Reload Images Open Print Find Stop
What's New! What's Cool! Upgrades Net Search Net Directory
CONTINUE
END SERVICE
Home

Location: http://teamster.usc.edu/cgi-bin/fixture/uncgi/fixdate

YOUR REQUEST WILL BE READY IN:

3.9 minutes. Sorting 460 fixtures, please wait.

Click just one time on CONTINUE button and wait your answer.

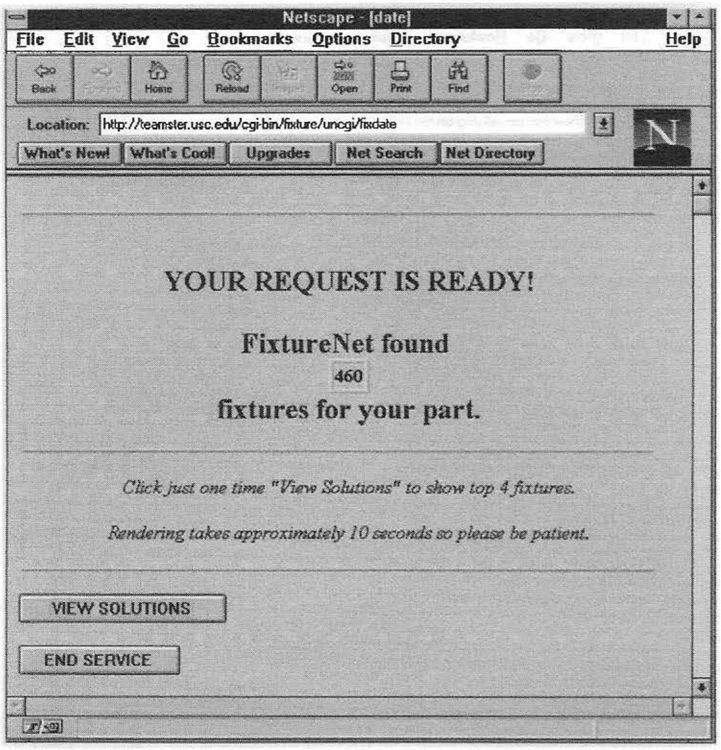
FIGURE 5. Time estimate generation considers current CPU load on the host machine.

part) consists of an array of picture elements (pixels) and the coordinates that define these points are determined by the local operating system and recorded by the browser application when the user clicks his mouse. When the user selects a point, its coordinates are passed to a drawing program (written in the C language). The coordinates are stored for use when the part will be submitted to *FixtureNet*. The user can also enter the part vertex coordinates manually through the keyboard, prepare and send (by FTP) a file listing of the coordinates which the server can read or can change and submit default input† provided in input text boxes.

We used the Unix$^{\text{TM}}$ Bourne shell script language to build our gateway scripts. These scripts embody a powerful feature of web browser and server interaction: they enable the users to interact with the web document.

A gateway script is a program that is run on a web server activated by input from a browser. It is usually a link between the server and some other program running on the system: they are also called CGI (common gateway interface) scripts. The gateway scripts are called by the server based on information from the browser. The URL (uniform resource locator) points to a gateway script in the same way that it points to any other

† Several simple sample parts, including a square, a pentagon and a star, are provided for those wishing for a quick-trial of *FixtureNet*.


Netscape - [date]
File Edit View Go Bookmarks Options Directory Help
Back Home Reload Open Print Find
Location: http://teamster.usc.edu/cgi-bin/fixture/uncgi/fixdate
What's New! What's Cool! Upgrades Net Search Net Directory
VIEW SOLUTIONS
END SERVICE

YOUR REQUEST IS READY!

FixtureNet found

460

fixtures for your part.

Click just one time "View Solutions" to show top 4 fixtures.

Rendering takes approximately 10 seconds so please be patient.

FIGURE 6. If more than zero fixtures are found for the part, *FixtureNet* offers to display the best four.

HTML page on a server; when the server receives the request, it notes that the URL points to a script (based on the file location, usually the cgi-bin directory) and executes that script.

The script performs some action based on the input, in our case, simply to call a correspondent C program. Then the script formats its result in a manner that the web server can understand. The web server receives the result from the script and passes it back to the browser, which formats and displays it for the user. *FixtureNet* uses 13 different gateway scripts and 13 correspondent C programs. Each script is a web server interface to call a C program binary.

When the part is submitted, *FixtureNet* parses the input in accordance with the input format specification (described for the user on the "explanation") page. If an error is found, a message is returned to the user, otherwise *FixtureNet* opens network communication with the fixture server on Machine A (*Teaser*) and sends the formatted input part.

After the Linux client sends the fixture server the data describing a polygonal part, the fixture server initiates the fixture design algorithm by spawning a fixture synthesis process which estimates run time based on part size and grid pitch.† The estimate is

† The grid pitch is the inverse of the distance between holes in the fixture plate. For example, a plate with holes spaced 2 in apart has a grid pitch of *one-half* (holes per inch), while if there were $\frac{1}{2}$ in between holes, the pitch would be *two*. The hardware has a maximum grid pitch, but by ignoring every other hole, we can reduce the virtual grid pitch by half, reducing computational complexity.

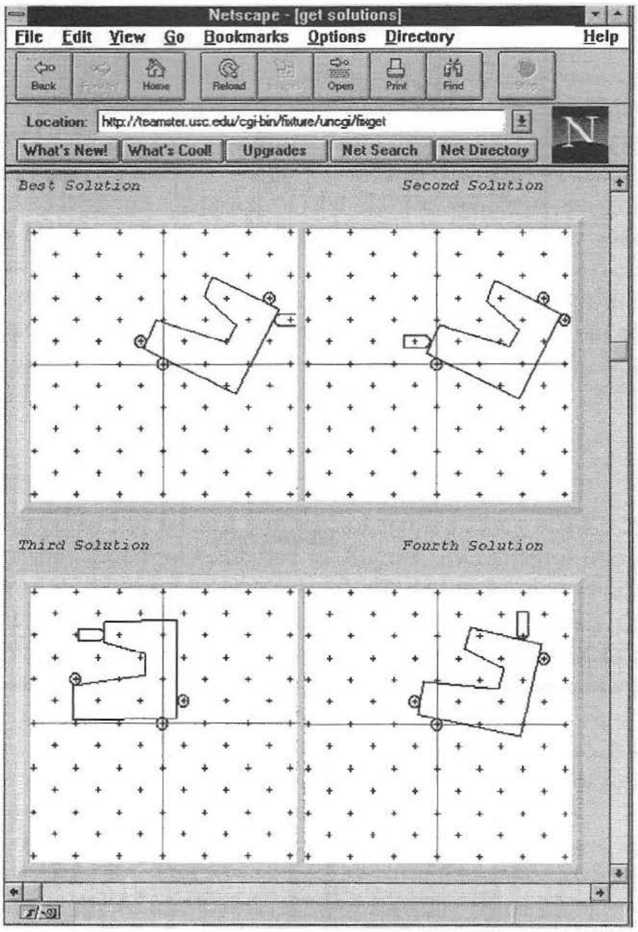

Netscape - [get solutions]
File Edit View Go Bookmarks Options Directory Help
Back Forward Home Reload Images Open Print Find Stop
Location: http://teamster.usc.edu/cgi-bin/fixture/uncgi/fixget
What's New! What's Cool! Upgrades Net Search Net Directory
Best Solution
Second Solution
Third Solution
Fourth Solution

FIGURE 7. The four best solutions (fixture configurations) for the hook-shaped part. The default quality metric is formulated to resist several generic combinations of forces and moments.

returned to the Linux client, which formats it into an HTML page and returns it to the user.

When the fixture synthesis program completes the design algorithm, it communicates the data via Windows dynamic data exchange (DDE) to the fixture server which relays it to the Linux client in the form of textual descriptions of solutions. Each solution includes the pose (position and orientation in the plane) of the user's part, the position of three locators on the lattice and the position and offset for a clamp such that the part is held in form closure.

The Linux client then runs a custom graphics routine to generate (CompuServe's) graphic interchange format (.gif) images of the part in each of the best four solutions.

Communication via Berkeley sockets is the key to building a cross-platform WWW service of this kind. We used the Windows Socket application programming interface

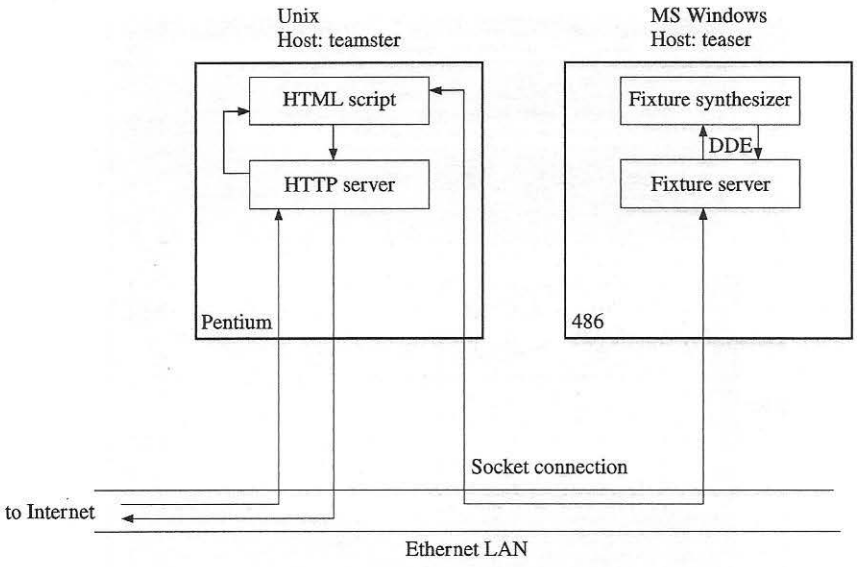


FIGURE 8. The *FixtureNet* architecture.

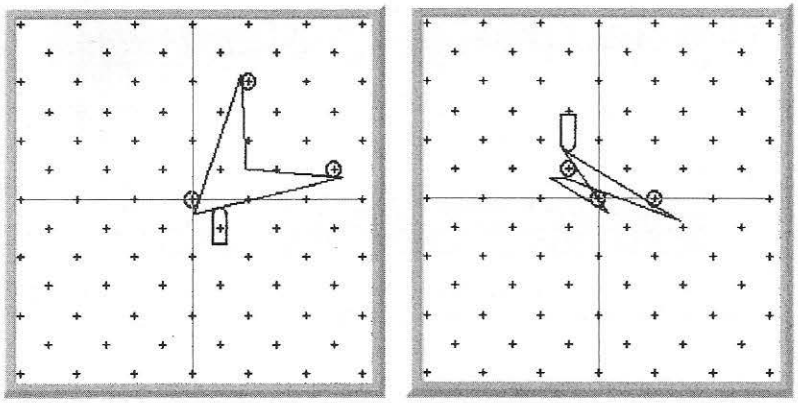


FIGURE 9. A typical part (left) submitted by a user with the drawing input feature, shown in its optimal fixture and an atypical part (right). The part on the right is not a valid part because it has a self-intersecting edge. Submitted parts like this are rare because users intuitively know that parts cannot have this configuration. *FixtureNet*, however, computes fixtures for invalid parts.

(API), a subset of Berkeley sockets. For rapid development, we implemented the algorithm in Visual Basic using a Windows Socket custom control [a precompiled MS Windows dynamic link library (DLL)] from Distinct Corporation.

There are a number of architectures that can be used for communication with sockets, and we experimented with several of them before finally settling on using a single client socket in the Linux client and a server socket in the fixture server. We used a 7-bit ASCII string to pass information through the socket connection formatted with an 8-digit service request identifier and a 3-digit-type code.

The Linux client initiates a fixture service request with a socket connect request and an initial fixture request message (code 001) that includes the part data (number of vertices and x–y coordinates of each vertex) and the desired grid pitch (coarse, medium or fine). When the fixture server receives the request, it responds with a "request acknowledged" string, and then spawns an instance of the fixture synthesis program which then generates a time estimate for the job based on the current CPU load and the part parameters. When the Linux client requests job status, the fixture server then relays the fixture synthesis program time estimate and state to the client. While multiple fixture synthesis program instances can be run simultaneously, the throughput of the system will not be increased by doing so. *FixtureNet* can be easily extended by adding additional server machines.

5.2. USER STATISTICS AND FEEDBACK

FixtureNet was first operational and publicly available in July of 1995. Some problems were identified at that time and remedied in August. Incremental improvements were incorporated through November, and usage statistics were compiled starting 16 December 1995. *FixtureNet* has been publicly available continuously since then. From 16 December 1995 to 31 March 1996, the *FixtureNet* usage statistics are as given in Table 1.

In many cases (35, 25%), users jump to another page and do not return to request to view the solutions. These solutions are computed, but are not delivered and not captured into the "sets delivered" statistics. A typical run time for a fixture computation is about a minute. The "square" example part takes 4 s.

The "user friendliness" of the input interface seems to have a strong influence on the types of requests submitted. The easiest way to submit a part is to use the drawing interface in which the user clicks the vertices of the part using his mouse on an image representation of the part. Most requests were submitted using the part drawing feature. Keyboard input is significantly more difficult. The user must compute the vertices of a part himself and submit them via keyboard entry. The "canned" parts for demonstration are counted as "keyboard" input and, if these were not available, it is not likely that the keyboard input count would be as high as it is. FTP file submission is by far the most

TABLE 1
FixtureNet statistics

FixtureNet statistics	Number
Requests of *FixtureNet* service input (keyboard)	149
Requests of *FixtureNet* service input (FTP file)	0
Requests of *FixtureNet* service input (drawing)	281
Solution sets delivered	287
Total fixture configurations computed	19731

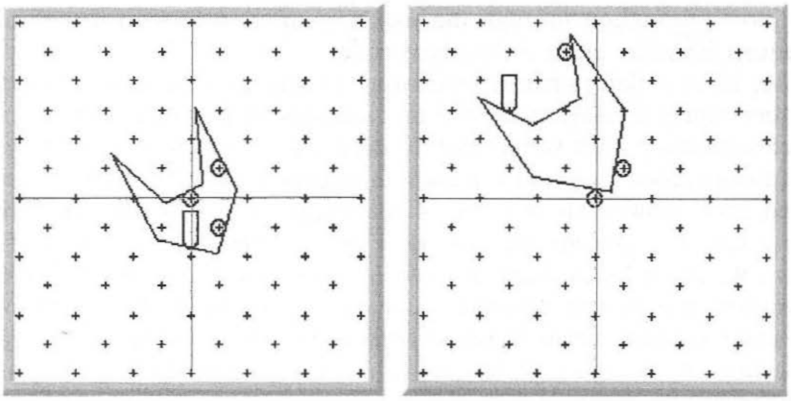


FIGURE 10. This user-submitted part (left) was drawn in reverse order (clockwise). As a result, *FixtureNet* interpreted the part as a hole in a plate and located the fixturing elements accordingly (on the interior of the polygon). The user learned rapidly, as shown by his subsequent submittal (right), drawn in the correct order.

difficult option, and exactly zero people chose to submit parts in this manner. We can hardly blame them. This option requires the user to compute the vertices of the part, store them in a prescribed file format, connect to the site via FTP and send the file. The user must then record an interaction identification number, and request the solution at a later time by submitting the ID number.

User feedback was generally enthusiastic and positive. Some samples are as follows.

- "This is really cool. I wish more researchers would dare to have their work mass-tested."
- "It is very nice drawing the polygon with the mouse!"
- "Nice with interactive WWW-services. You can do a very good CAL (computer-aided learning) package this way."

As is frequently the case with software applications, users did not always read all the instructions before jumping in. One feature of our fixturing algorithm is that if the part shape is drawn in a clockwise ordering of vertices, the shape is treated as negative (a hole in a plate, e.g.) and fixtured as such. There are two ordering possibilities for drawing parts, and several users, choosing the wrong one for their purposes, reported our hole-fixturing feature as a bug, indicating that they had not read all the instructions. Some of the parts submitted by users are shown in Figures 9 and 10.

6. Discussion

We do not claim that *FixtureNet* is ready for practical use by industrial designers. *FixtureNet* is a feasibility study for how the Internet can be used as an interactive resource for design and manufacturing analysis. We have demonstrated how geometric part descriptions can be sent over the Internet and how the WWW can provide remote execution of geometric algorithms and graphical display of results. Much remains to be done.

FixtureNet also suggests a model for sharing software that is not used frequently. Rather than owning a user license and copy of the software, users can send data and receive results, with payment based on usage. For designers of algorithms, the Internet provides access to an enormous community of tinkerers who will be more than happy to discover flaws in an algorithm. The Internet is also a great way to disseminate research results to academic colleagues, industrial users and ultimately to the taxpayers who support the work directly or indirectly.

7. Future work

Since *FixtureNet* was developed in the summer of 1995, WWW software has improved dramatically. The introduction of client-side processing languages such as Java will allow us to replace the current graphical input method which requires a new image to be generated at the server and transmitted to the client after each vertex is clicked in. The new input method uses a Java applet on the client side that is by far faster and gives more flexibility, such as the ability to edit the part locally. The resulting geometry will be transmitted to the server, which will compute the set of fixtures and return them to the Java client for display. The Java client could also allow users to apply forces interactively (graphically with the mouse) and view reaction forces. We expect this new version, *FixtureNet II*, to be available in Spring 1997.

We thank Steven Gentner and Jeff Wiegley (graduate students at USC) for their generous assistance with the details of the WWW, Linux, sockets and .gif images. Steven was instrumental in getting over some initial rough spots and it was with Jeff's continued prodding that we arrived at a satisfactory socket communication architecture. We also thank George Bekey and Ari Requicha (Professors at USC) for their advice and discussions on modular fixturing.

This work was supported in part by the National Science Foundation under Award DDM-9215362 (Strategic Manufacturing Initiative) and by NSF Young Investigator Award IRI-9457523, and by a grant from Cu-Co, Inc., a vendor of modular fixture systems. Part of this research was performed while Goldberg was at the University of Southern California.

References

ASADA, H. & BY, A. B. (1985). Kinematic analysis of workpart fixturing for flexible assembly with automatically reconfigurable fixtures. *IEEE Journal of Robotics and Automation*, RA-1.

BROST, R. & GOLDBERG, K. (1995). A complete algorithm for synthesizing modular fixtures for polygonal parts. *International Conference on Robotics and Automation*, IEEE, May, 1994.

ENGEL, F. L. & HAAKMA, R. (1993). Expectations and feedback in user-system communication. *International Journal of Man–Machine Studies*, **39**, 427–452.

GOLDMAN, A. J. & TUCKER, A. W. (1956). *Polyhedral Convex Cones*. Princeton, NJ: Princeton University Press.

HOFFMAN, E. G. (1987). *Modular Fixturing*. Lake Geneva, WI: Manufacturing Technology Press.

LAKSHMINARAYANA, K. (1978). *The mechanics of form closure*. Technical Report 78-DET-32, ASME.

MISHRA, B., SCHWARTZ, J. J. & SHARIR, M. (1987). On the existence and synthesis of multifinger positive grips. *Algorithmica*, **2**, 641–658

PONCE, J., BURDICK, J. & RIMON, E. (1995). Computing the immobilizing three-finger grasps of planar objects. Draft.

REULEAUX, F. (1963). *The Kinematics of Machinery*. New York: Macmillan, 1876 (Republished by New York: Dover, 1963).

WAGNER, R., ZHUANG, Y. & GOLDBERG, K. (1995). Fixturing faceted parts with seven modular struts. *International Symposium on Automation and Task Planning*, IEEE.

WALLACK, A. S. & CANNY, J. F. (1994). Planning for modular and hybrid fixtures. *International Conference on Robotics and Automation*, IEEE.

WALLACK, A. S. (1995). *Algorithms and techniques for manufacturing*. Ph.D. Dissertation.

WOLTER, J. D. & TRINKLE, J. C. (1994). Automatic selection of fixture points for frictionless assemblies. *IEEE Transactions on Robotics and Automation*.

YU, K. & GOLDBERG, K. (1995). Loading planar fixtures in the presence of uncertainty. *International Symposium on Assembly and Task Planning, ISATP Proceedings*, August.

ZHUANG, Y., GOLDBERG, K. & WONG, Y. C. (1994) On the existence of modular fixtures. *IEEE International Conference on Robotics and Automation*, pp. 543–549. San Diego, CA.