Proceedings of DETC'97
1997 ASME Design Engineering Technical Conferences
September 14-17, 1997, Sacramento, California

DETC97/DFM-4353

FIXTURENET II: INTERACTIVE REDESIGN AND FORCE VISUALIZATION ON THE WEB[1]

Charles Anderson
IEOR Department
University of California
Berkeley, CA 94720
cander@ieor.berkeley.edu

Yan Zhuang
Computer Science Department
University of California
Berkeley, CA 94720
yzhuang@cs.berkeley.edu

Ken Goldberg
IEOR Department
University of California
Berkeley, CA 94720
goldberg@ieor.berkeley.edu

ABSTRACT

Digital communication over the Internet offers advantages in terms of speed, efficiency and automation. Fortunately, new geometric algorithms for design, simulation, and manufacturing have been developed and reported in the research literature. Unfortunately, the impact of these advances on the manufacturing community has been limited since implementations are difficult to port from one platform to another.

As an example of how the Web can facilitate interactive design, we focus on one specific application area: modular fixture design. We have substantially extended our previous fixture design service, FixtureNet(Wagner *et al.*, 1996), and added interactive tools to allow the user to build a deeper, more intuitive understanding of the fixtures found by FixtureNet. The first tool allows the user to simulate the effects of forces applied to the part in the fixture. Our second tool enables the user to consider changes to a part and verify in real-time that the fixture will still immobilize the modified part.

Our tools balance the tasks between the Web client and a central server, performing fast user interactions in the client while running compute-bound fixture design jobs on the server.

The implementation of this work can be found online at: `http://riot.ieor.berkeley.edu/riot/Applications/FixtureNet`.

Keywords: modular fixture, design rule, Java applet

[1]THIS WORK WAS SUPPORTED IN PART BY NSF AWARDS IRI-9612491, FORD MOTOR COMPANY AND PRESIDENTIAL FACULTY FELLOW AWARD IRI-9553197 TO PROF. GOLDBERG.

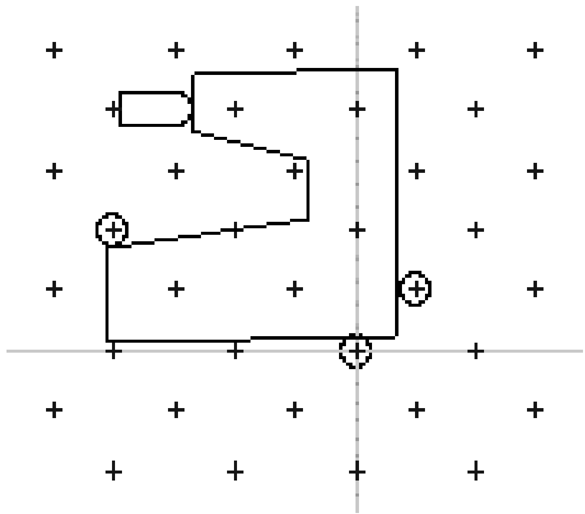

Figure 1. EXAMPLE OF A PART AND A MODULAR FIXTURE.

INTRODUCTION

Modular Fixturing

A fixture is a device that holds a part for machining, assembly, inspection, etc. Fixturing is concerned with "constraining, despite the application of an external wrench, all motions of a rigid body" (Wentink *et al.*, 1996). That is, given a rigid body (a part) we wish to hold it firmly (with a fixture) such that the part cannot translate or rotate. In our case, we are concerned with a subset of this general problem where: the part is two dimensional and polygonal, and the points of the fixture are restricted to a discrete set of possible locations.

This problem is known as *modular fixturing* of planar, polygonal parts. The workspace is a planar surface with holes drilled in some regular lattice, typically in a square-shaped pattern. A fixture is composed of a number of simple fixturing elements (*fixels*) that are placed in the holes, and can only be located where there is a hole - i.e. discrete set of locations as opposed to a continuum.

Two simple fixels are locators and clamps. A locator is simply a peg or pin that is inserted into a hole in the workspace. A clamp has a peg that fits in a hole, but it also has a piece that extends up to one unit out from the hole. This extension only happens along the axes of the workspace: north, south, east or west; it cannot extend in any arbitrary direction.

In general, we only need three locators and one clamp to fixture a sufficiently large, polygonal part (Wentink *et al.*, 1996). If a part is too small, it cannot be fixtured. A more thorough treatment of conditions for fixturing can be found in Zhuang et al. (1996).

For our purposes, a fixture is composed of three locators and one clamp. Our problem can then be stated as: *given a part, find all fixtures to hold the polygonal part in form closure - i.e. immobilize it.* Figure 1 shows an example of a part and a fixture.

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.

A complete algorithm to find all modular fixtures was described by Brost and Goldberg (1994). Because it generates all possible solutions, the fixture design algorithm often generates counter-intuitive solutions that may be overlooked by even an experienced machinist, much as chess machines can play moves that look naïve at first glance but lead the experienced chess player to explore new variations.

The job is not necessarily complete after finding an acceptable fixture for a specified part. In the manufacturing industry, we often have to redesign a product. Usually, a new design requires a completely different fixture design, which costs both money and time. This tempts us to consider the following problem: *given a planar polygonal part, and its fixture consisting of three locators and one clamp, is it possible to impose a design rule to specify how much the part shape can be modified such that the redesigned part can still use the same set of locators?*

The original FixtureNet brought fixture design to the Web. We have extended FixtureNet to include interactive tools that allow the user to explore qualities of fixtures that are difficult to quantify.

Related Work

Fixturing is closely related to grasping. The purpose in both cases are to immobilize the part. Modular fixturing is, however, different from grasping because the locators are restricted to the discretized holes on the regular lattice.

Recently there has been a surge of research on modular fixturing. Brost and Goldberg (1996) present the first complete synthesizing algorithm that guarantees to find a fixture consisting of three locators and one clamp for any given polygon. The algorithm also indicates if no such fixture exists. Wallack and Canny (1994) present a complete algorithm for a fixturing model using four locators on a split lattice that can open and close like a vice. Penev and Requicha (1995) present their study on fixture foolproofing: given a fixture consisting of three locators and one clamp, where should the blocking pins be inserted to insure that the part can be loaded in only one desired pose? Overmars *et al.*(1995) propose a new class of planar fixtures that includes flat edge-contact and present a complete algorithm to find such fixture. Brost and Peters (1996) present an algorithm that automatically designs fixtures and assembly pallets to hold three-dimensional parts. Rong (1997) explores an analytical methodology to conduct modular fixture planning based on geometric access analysis. Zhuang and Goldberg (1997) shows a design rule for how to modify a part while still being able to re-use the given fixture.

There are numerous research projects and Web sites related to fixturing and manufacturing via the Web. There is an early fixture verification site at CMU (1996). Cheng (1997) discusses a new integration language environment called C^H. This is available at The Integration Engineering Laboratory (1997). FIXMA(1997) is part of the Machine Tool-Agile Manufacturing Research Institute (MT-AMRI). Chui et al. (1997) discusses a new manufacturing environment for rapid prototyping being developed at The Integrated Manufacturing Laboratory (1997) at U.C. Berkeley.

FixtureNet

FixtureNet made the Brost-Goldberg algorithm available via the Web in the Summer of 1994. To our knowledge, this was the first fixture *design* system on the Web. FixtureNet allows the user to enter a part using the mouse in a browser to click on each vertex of the part. When done, the user clicks on a button to submit the part to a server machine running at the authors' institution. The server computes all possible fixtures and returns images of them to the user's browser.

FixtureNet is composed of two major components: the fixturing programs and a collection of CGI programs (December and Gingburg, 1995) that interact with the fixturing programs. The fixturing programs are composed of a server program that accepts commands over TCP/IP connections and the actual fixturing engine. Both of the fixturing programs are written in Visual Basic from Microsoft and consequently run only under Microsoft Windows (3.x, 95, and NT).

FixtureNet has no real client-side component. It was targeted for version 1.2 of Netscape before support for Java was added. This means all processing of input and rendering of output is done via CGI programs running on the server machines. The CGI programs generate GIF files to represent the part and the work space, and these are interpreted by the browser as image maps to capture the mouse click locations.

Although FixtureNet provides a fairly complete interface for defining parts and displaying solutions, it is slow due to:

- Part input is based on CGI programs such that each mouse click runs a new instance of the CGI program. This is slow and resource intensive.
- In addition to invoking the CGI programs on each mouse click, there are two HTTP connections that must be created and destroyed: one for the HTML file returned by the CGI program and one to retrieve the GIF file generated by the CGI program and reference by the HTML document.
- These GIF files contain a large amount of data compared to the amount of data needed to represent the coordinates of the mouse click. This can be slow, especially on slow network connections like modems.

Objectives

Our project builds on the work of FixtureNet to extend the functionality and improve the performance with the aid of client-side processing. Our first objective is to simply update the FixtureNet user interface using Java to replace all of the CGI programs. This creates a faster user interface by moving the data input and display work to the client machine - i.e. each time the user clicks the mouse to define a vertex we process it on the client instead of communicating with the server. Note that the fixture synthesis algorithm is still run on the server. For our purposes, we treat the FixtureNet core like a legacy system - i.e. we did not modify any of it, rather we used the existing interfaces (ASCII data transmitted via TCP) developed for the CGI programs.

To this core we have added two new features: interactive reaction force display and interactive part design evaluation. The first tool allows the user to simulate interactively forces on the part to view the reaction forces at the fixels. The second tool allows the user to move vertices of the part around on the display to experiment interactively with the part and its fixture to see how the part can change and still keep the same fixture. Both of these interactive exploration tools serve to build an intuitive understanding beyond what can be gleaned from the otherwise static, two-dimensional display of the part and its fixture.

FIXTURENET II

The new version of FixtureNet is implemented as a Java applet that can be downloaded from a Web site and run in the user's browser. The browser can be any Java-compatible browser such as Netscape 2.0, Netscape 3.0 or Internet Explorer 3.0. In contrast to many stand-alone Java applets where all processing is performed within the Java client, FixtureNet II uses the original FixtureNet server process running on the HTTP server machine to perform the actual fixturing calculations.

The choice to leave the fixturing calculations on the server was a deliberate one motivated by a number of considerations. First of all, we did not wish to spend time re-inventing an existing "wheel". We wanted to proceed to new work such as interactive analysis rather than reimplementing the existing algorithm. Secondly, we wanted to explore the possibilities of using Java in a client-server environment rather than a client-only model, which is more typical of early Java applets. There is work under way at USC, the birthplace of FixtureNet I, to convert the fixturing engine to Java. When that becomes available, we will be able to compare the client-server and client-only implementations.

Figure 2 shows the process architecture of the old and new versions of FixtureNet. The top terminal (labeled FN 1) shows the CGI-based architecture. All user inputs (mouse and button clicks) result in a call to the HTTP server that creates and executes the CGI processes. To compute fixtures, a CGI program calls the Fixture Server via TCP/IP, which in turn runs the Fixture Synthesizer via Microsoft's DDE. For details of FixtureNet I, see the original FixtureNet paper (Wagner *et al.*, 1996).

Ideally, the new FixtureNet client would invoke the Fixture Server directly, as shown in the middle terminal (FN 1.5). However, the Fixture Server is not capable of handling multiple users simultaneously. Therefore, we created a Proxy Server to handle the coordination of multiple users, as shown in the bottom terminal (FN 2). This is a significant improvement over the original version because it does not require creating a new CGI process for every user input.

The process of fixturing in FixtureNet can be broken into the following steps: entering the part, computing the solution, and displaying the solution.

Part input with the new version appears much the same

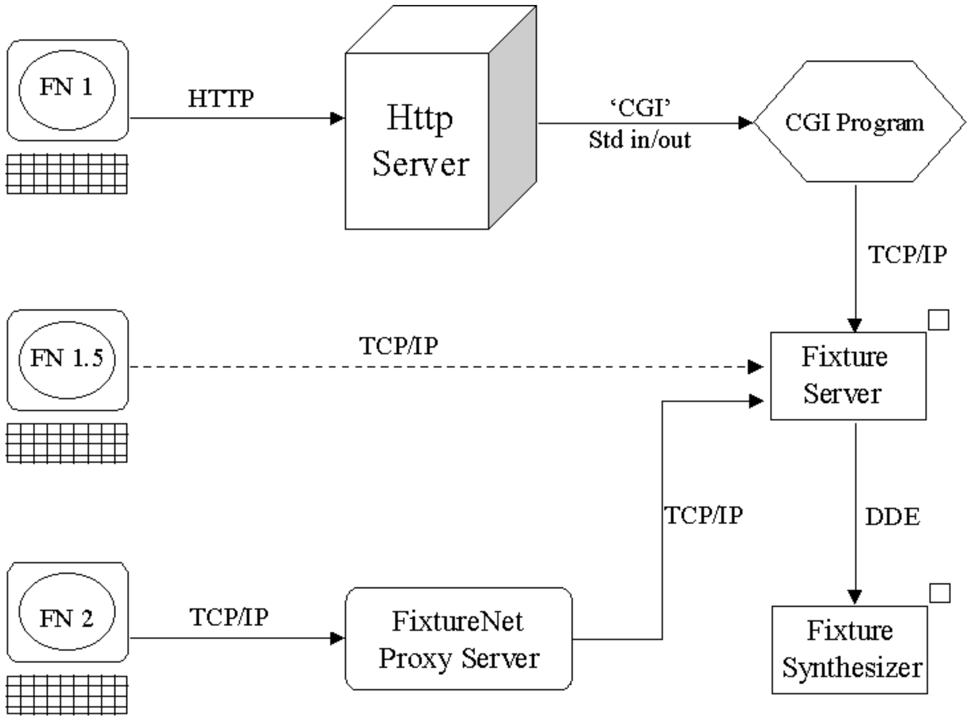

Figure 2. FIXTURENET PROCESS BLOCK DIAGRAM.

as part input with FixtureNet. The user clicks on the workspace canvas to trace out the vertices of the part. The difference is that all display processing is performed locally within the Java client program. Each mouse click is interpreted locally and the display is updated without intervention from the server.

When the user is ready to submit the part, he clicks on the submit button. This draws in the last edge of the part and submits it to the FixtureNet server. The applet then polls the FixtureNet server waiting for the server to complete the fixturing calculations.

When the server is done, the applet requests the first fixturing solution. A solution is composed of the positions of the three locators, the position and orientation of the clamp, and a translation and rotation transformation to apply to the part to position it in the fixtured pose. At this point the solution can be displayed to the user. The user can scan forward and backwards through the fixture solutions that the server found.

INTERACTIVE FORCE DISPLAY

Once we have a fixture from FixtureNet, we may wish to study the effects of forces applied to the part such as force that will result when the part is machined while in the fixture. When a force is applied, the fixels will resist the force. These are the reaction forces we wish to study.

In preparation for displaying the reaction forces, the applet must compute the contact point of the fixels with the edges. Although the FixtureNet server obviously has this information, it is not communicated back to the client - a common problem when dealing with legacy systems. Therefore, we must calculate it from the information provided. This is difficult because the FixtureNet server supports fixels with non-zero diameters. This means that fixels do not lie directly on an edge, but rather a small distance from it.

We reverse-engineer this information by trying every pair of edge and fixel. We compute the distance from the fixel to the edge. For each fixel, we keep track of the closest edge we have found. When we are done with the enumeration, we have four edges paired with the four fixels. Note, that it is possible for two fixels to be located on the same edge.

For a given fixel, once we know which edge it contacts, we can easily compute the point on the edge were the fixel makes contact. This is the point where reaction forces will originate. The force is always perpendicular to the edge in contact with the fixel, in the direction of the part interior.

Once the solution is displayed, the user can click anywhere within the part and drag the mouse to a new location to specify a force. The force acts at the point where the user first clicked. The magnitude and direction of the force are

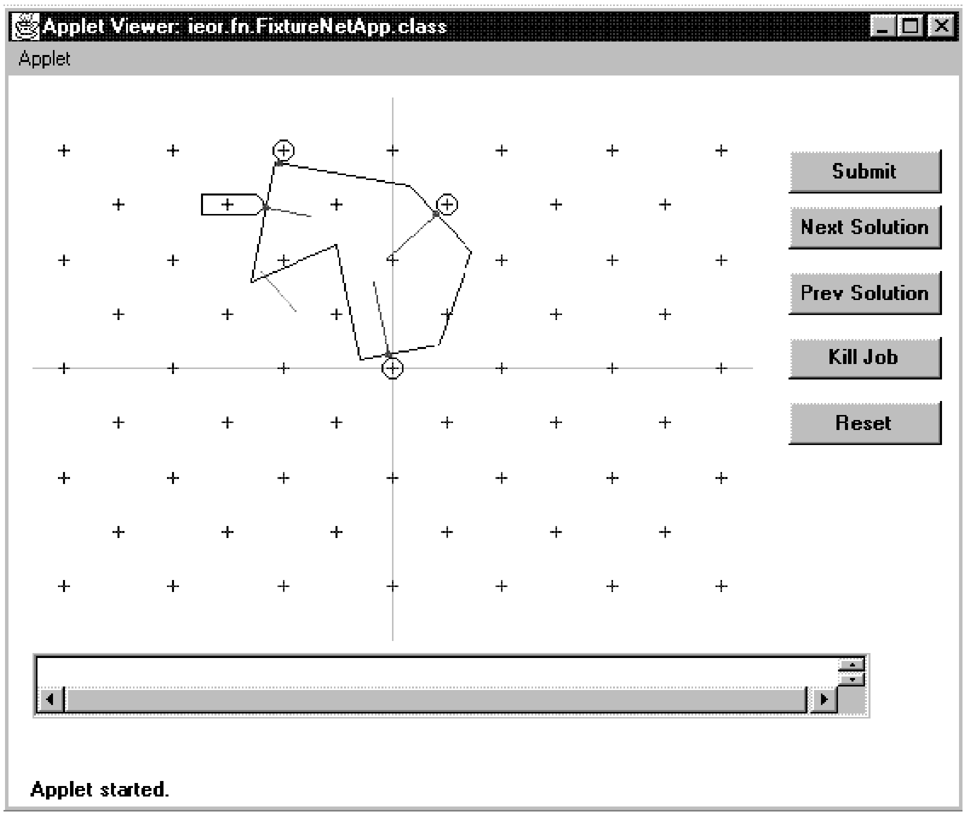

Figure 3. SCREEN SHOT SHOWING A PART, THE FIXTURE, THE USER-SPECIFIED FORCE, AND THE REACTION FORCES.

indicated by the distance and direction from the click-point to the current mouse location.

This force exerts translation forces on the part in both the X and Y directions, as well as inducing a torque. The fixels will react against this force in the direction normal to the edge. The magnitude of the reaction force at each fixel is dependent on the magnitude and direction of the user-supplied force. In fact, one fixel will have no reaction force at all - i.e. the user force is borne by only three of the fixels.

To compute the magnitude of the reaction forces at each fixel, we solve the following system of equations:

$$\begin{pmatrix} r_{1x} & r_{2x} & r_{3x} & r_{4x} \\ r_{1y} & r_{2y} & r_{3y} & r_{4y} \\ M_1 & M_2 & M_3 & M_4 \end{pmatrix} \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \end{bmatrix} = \begin{bmatrix} -F_x \\ -F_y \\ F_x(-y_t) + F_y(x_t) \end{bmatrix} \qquad (1)$$

where,

$M_i = r_{ix}p_{iy} + r_{iy}(-p_{ix})$;
r_i is the unit reaction force at fixel i with components r_{ix} and r_{iy};
(p_{ix}, p_{iy}) is the contact point of fixel i with the part edge;
F_x and F_y are the X and Y components of the user-specified force;
(x_t, y_t) is the location of the user force;
R_i is the magnitude of the reaction force at fixel i.

The first two equations say that the translation forces from the fixels and the user force are balanced - i.e. cancel each other out. The last equation says that the torque induced by the user force is canceled out.

Because this system is over constrained (three equations and four unknowns), and because we are searching for a solution where one fixel has zero reaction force, we solve the system four times, each time setting one of the R_i terms to zero. Our objective is a solution with non-negative values for the remaining R_i terms; a solution with a negative magnitude for a reaction force is nonsensical.

Once we have a non-negative solution with at least one term equal to zero, we can use these to compute the lengths of the reaction vectors. We can then draw them normal to the edges, pointing towards the interior of the part.

A sample screen shot showing a part, the fixture and the reaction forces is shown in figure 3. The user-applied force can be see as a line originating at the leftmost vertex of the part and moving down and to the right - i.e. towards the origin of the workspace.

All of these computations and the display of the force vectors take place interactively while the user is dragging the mouse. The computations are performed on the client machine without contacting the FixtureNet server. In fact, it would be very difficult to provide real-time updates if round-trip communication with the server were required. The resulting user interface would probably be very jerky and uneven.

DESIGN RULES FOR TOLERANCE INSENSITIVE FIXTURES

In the manufacturing industry, we often have to redesign a product. Usually a new design requires a completely different fixture design, which costs both money and time. This tempts us to consider the following problem: given a planar polygonal part, and its fixture which consists of three locators and one clamp, is it possible to specify how much the part shape can be modified such that the redesigned part can still use the same set of locators? Moreover, we want to have an interactive user interface so that the designer can be immediately notified if the modification is too great to re-use the same fixture.[2]

We assume that the designer will redesign the part by modifying one edge at a time. Clearly, the designer can modify the non-contacting edges at will, without affecting the fixture. The real question is: *given three locators and their corresponding contacting edges, if we keep two of the edges, how much we can modify the third edge while still being able to achieve three point contact? (Figure 4)* Note that the dashed curves in Figure 4 represent all the edges connecting the three edges in consideration. Without loss of generality, we expand the part by the radius of locators using a Minkowski sum operation. This allows us to consider each locator as an ideal point. The original part contacts a locator if and only if the corresponding edge of the expanded part passes through the point. In the rest of this section, we simply refer to the expanded part as the part.

The Contact Locus

Given two locators and their corresponding edges, the part can move while maintaining the two point contact.

[2]Very likely the clamp position has to change. Therefore we focus on the re-usability of the three locators.

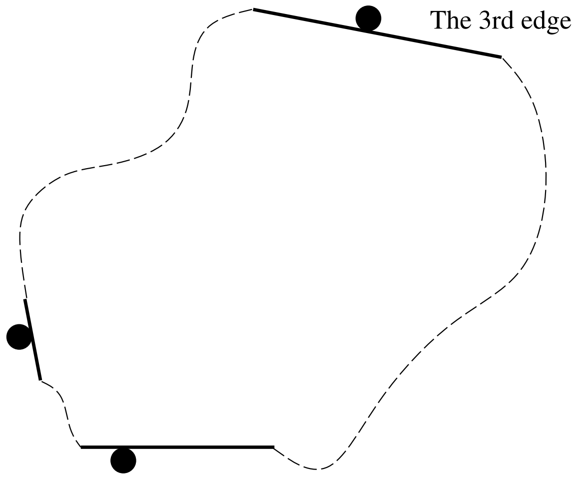

Figure 4. HOW MUCH CAN WE MODIFY THE THIRD EDGE SUCH THAT WE CAN STILL ACHIEVE THREE POINT CONTACT?

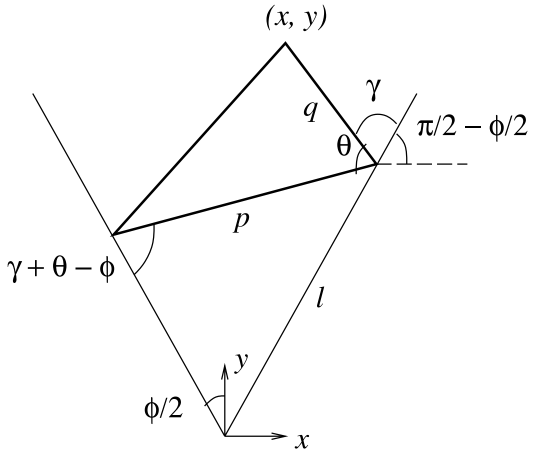

Figure 5. IT IS AN *ELLIPTIC* CURVE.

During the motion, the third edge in consideration sweeps out a region within which it can achieve the third point contact. This motion is difficult to characterize and the region swept out by the third edge has a complicated shape. To simplify the analysis, we can instead consider the motion of the triangle, formed by the three point locators, in the moving frame attached to the moving part. The triangle's two vertices can slide on two edges of the part, while the third vertex sweeps out a curve in the moving frame. If this curve in the moving frame intersects the third edge, there exists a two-dimensional rigid body motion to achieve three point contact. Note that all three edges are now fixed edges in the moving frame. Now the question becomes: what is that curve in the moving frame?

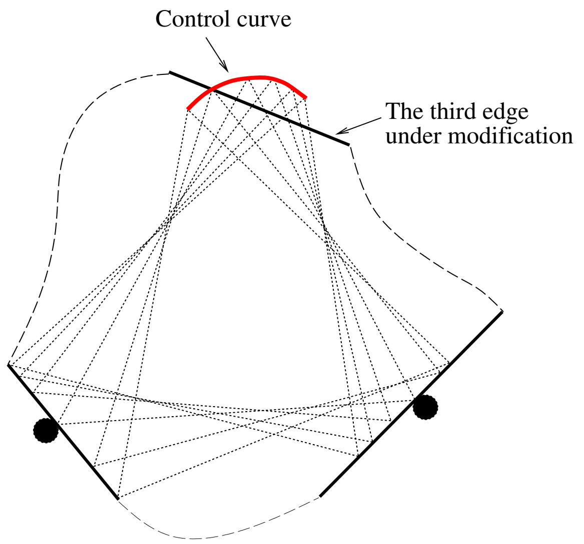

Figure 6. THE THIRD POINT CONTACT CAN ALWAYS BE ACHIEVED AS LONG AS THE THIRD EDGE INTERSECTS THE CONTROL CURVE.

To characterize the curve swept out by the third locator, we assume, without loss of generality, that the triangle is sliding on a cone, which has its apex at the origin and is symmetric about y-axis (Figure 5). Furthermore, given the cone and the triangle, the following values are constants: ϕ, p, θ and q. The position (x, y) of the third vertex of the triangle is uniquely determined by a single parameter γ. Zhuang and Goldberg (1997) shows in detail that this curve is an elliptic curve:

$$ax^2 + 2bxy + cy^2 + d = 0 \tag{2}$$

where

$$\begin{cases} a = q^2 - pq\frac{\sin(\frac{\phi}{2}-\theta)}{\sin\frac{\phi}{2}} + \frac{p^2}{2-2\cos\phi} \\ b = \frac{2pq\cos\theta - p^2}{2\sin\phi} \\ c = q^2 - pq\frac{\cos(\frac{\phi}{2}-\theta)}{\cos(\frac{\phi}{2})} + \frac{p^2}{2+2\cos\phi} \\ d = -q^2(q - p\frac{\sin(\phi-\theta)}{\sin\phi})^2 \end{cases} \tag{3}$$

We call this curve the *contact locus* for design. The detail of a similar derivation of the elliptic curve can be found in (Jia and Erdmann, 1996).

Interactive Fixture Design Using the Contact Locus

This contact locus enables us to design an interactive user interface to guide the process of part modification because the contact locus provides us a simple design rule: the third edge has to intersect the elliptic contact locus in the moving frame (Figure 6). Since the contact locus is quadratic, the intersection can be instantly verified.

Our interface allows the user to select an edge to be manipulated. The user can move the entire edge or drag a vertex to change the part's shape. So long as the modified shape is compatible with the fixture, the display is updated in real-time. As soon as the edge no longer intersects the contact locus, the indicator on the upper right corner (figure 7) changes color to notify the user the violation of the design rule. Again, all of this processing is performed on the client without server interaction.

Figure 7 shows an example of design process. On the left, we show an initial part and its fixture. On the right, the part has been modified but still uses the same locators. Of course, the clamp position is different.

CONCLUSIONS

Geometric algorithms are often proprietary, closed, or otherwise unavailable to large numbers of people. FixtureNet was an early effort to make modular fixturing available via the Web. In this paper, we have enhanced and extended FixtureNet through the balanced use of client-side processing. The new version provides the same basic part input and fixture display facilities as the original, but the user interface is much more responsive because the user interface processing is performed locally on the client machine using Java.

After FixtureNet II submits a part of the FixtureNet server and receives a solution, we provide two new tools to perform interactive analysis of the fixture. First the user can simulate the effect of forces on the part through a real-time display of the reaction forces. Secondly, the user can consider modifications to the part to see how much the part can change and still be compatible with the original fixture. Both of these interactive tools build a much deeper intuition about the fixture and how it interacts with the part.

ACKNOWLEDGMENTS

We owe great thanks to to Rick Wagner and Giuseppe Castanotto for the original FixtureNet server as well as providing documentation and support for it. Also, thanks to Jaramporn "Jack" Hassamontr who provided an implementation of Gaussian elimination that was adapted to Java to solve the equations for the reaction forces.

REFERENCES

Randy Brost and Ken Goldberg. A complete algorithm for synthesizing modular fixtures for polygonal parts. In *In-*

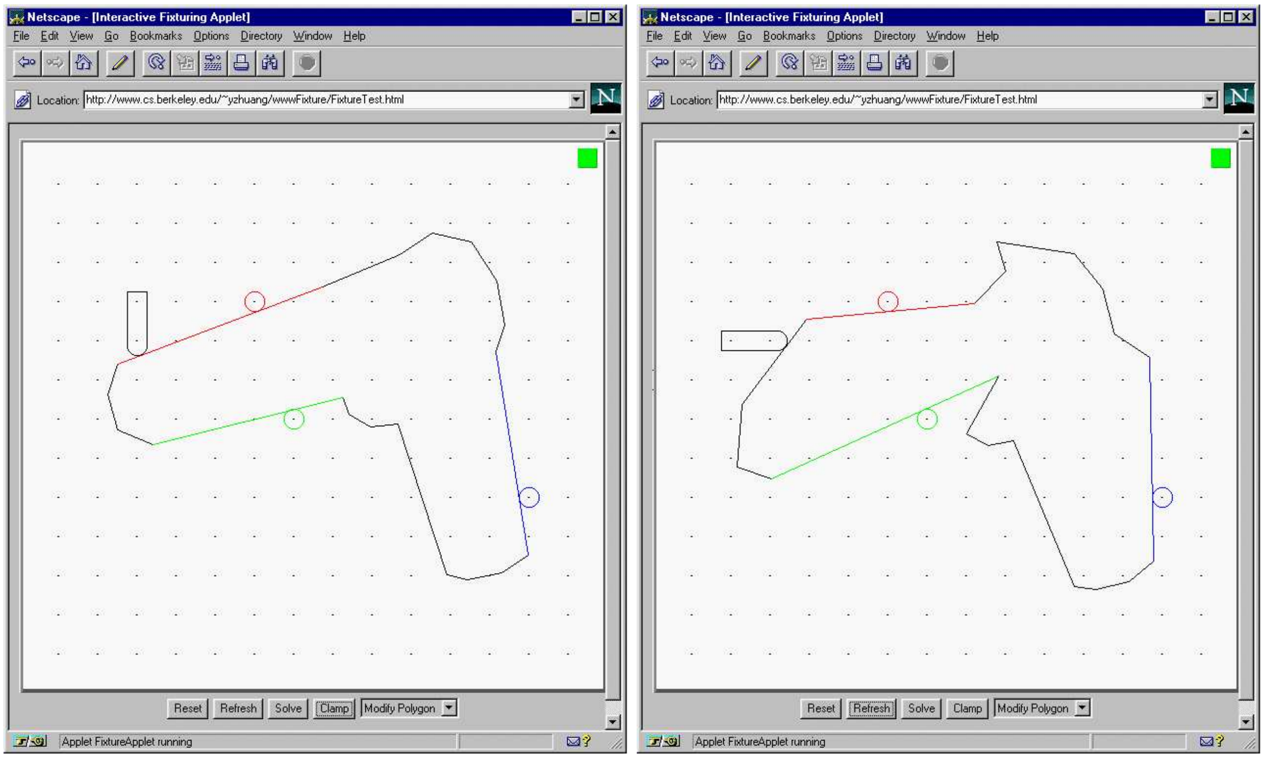

Figure 7. AN INITIAL PART (LEFT) AND A MODIFIED PART (RIGHT) THAT SHARE THE SAME LOCATOR POSITIONS.

ternational Conference on Robotics and Automation. IEEE, May 1994. to appear in the IEEE Transactions on Robotics and Automation.

Randy C. Brost and Kenneth Y. Goldberg. A complete algorithm for designing planar fixtures using modular components. *IEEE Transactions on Robotics and Automation*, 12(1):31–46, February 1996.

Randy C. Brost and Ralph R. Peters. Automatic design of 3-d fixtures and assembly pallets. In *IEEE International Conference on Robotics and Automation*, April 1996.

Harry H. Cheng. Network computing and its applications in design and manufacturing. *Proceedings of the 1997 NSF Design and Manufacturing Grantees Conference*, January 1997.

William H Chui, Paul K Wright, and Paul Sheng. Network manufacturing environment for rapid prototyping and education. *Proceedings of the 1997 NSF Design and Manufacturing Grantees Conference*, January 1997.

John December and Mark Ginsburg. *HTML and CGI Unleashed*. Sams.net Publishing, 1995.

FIXMA - A Testbed for Fixture Modeling and Analysis Software. `http://tool.ie.psu.edu/~fixture`.

E. G. Hoffman. *Modular Fixturing*. Manufacturing Technology Press, Lake Geneva, Wisconsin, 1987.

Integration Engineering Laboratory. `http://iel.ucdavis.edu`.

Integrated Manufacturing Laboratory. `http://kingkong.me.berkeley.edu`.

Yan-Bin Jia and Michael Erdmann. Geometric sensing of known planar shapes. *The International Journal of Robotics Research*, 15(4):365–392, August 1996.

Raju Mattikalli. The CMU Fixture Design Tool on Netscape and Mosaic. `http://www.cs.cmu.edu/afs/cs.cmu.edu/user/rajum/www/fix4.html`.

Mark Overmars, Anil Rao, Otfried Schwarzkopf, and Chantal Wentink. Immobolizing polygons against a wall. In *ACM Symposium on Computational Geometry*, 1995. Vancouver, BC.

Kamen Penev and Aristides A. G. Requicha. Fixture foolproofing for polygonal parts. In *International Symposium on Assembly and Task Planning*. IEEE, August 1995. Pittsburgh, PA.

Yiming (Kevin) Rong. Scientific basis for automated modular fixture planning. *Proceedings of the 1997 NSF Design and Manufacturing Grantees Conference*, January 1997.

R. Wagner, G. Castanotto, and K. Goldberg. Fixturenet: Interactive computer aided design via the www. *In-*

ternational Journal of Human Computer Studies, accepted August, 1996.

Aaron Wallack and John Canny. Planning for modular and hybrid fixtures. In *International Conference on Robotics and Automation*. IEEE, May 1994.

Chantal Wentink, A. Frank van der Stappen, and Mark Overmars. Fixture planning. In Jean-Paul Laumond and Mark Overmars, editors, *Workshop on Algorithmic Foundations of Robotics*, July 1996.

Yan Zhuang and Ken Goldberg. On the existence of solutions in modular fixturing. *The International Journal of Robotics Research*, 15(6):646–656, December 1996.

Yan Zhuang and Ken Goldberg. Design rules for tolerance-insensitive and multi-purpose fixtures. In *8th International Conference on Advanced Robotics*, July 1997. Submitted.