Planning for Modular and Hybrid Fixtures

Aaron S. Wallack * John F. Canny †
Computer Science Division
University of California
Berkeley, CA 94720
http://robotics.eecs.berkeley.edu/ (wallack,jfc)/(wallack,jfc).html

Abstract

Fixturing encompasses the design and assembly of fixtures to locate and hold a workpiece during a manufacturing operation such as machining or assembly. We have implemented an automated design algorithm for a fixturing toolkit called the fixture vise; the fixture vise involves two fixture plates mounted on jaws of a vise and modular fixturing elements (pegs or flatted pegs). Generally, a fixture vise can handle appropriately sized prismatic workpieces in many different ways. The design algorithm runs in $O(A)$ time, where A is the number of configurations achieving simultaneous contact between the modeled object and four fixture elements; since simultaneous contact is a necessary but not sufficient precondition for force closure, A provides an upper bound on the number of force closure fixture configurations.

1 Introduction

The task of immobilizing and locating a workpiece via mechanical devices is commonly called fixturing or workholding. Fixtures can be fabricated from stock, or assembled from a toolkit of modular fixture elements; the latter approach is termed modular fixturing. In this paper, we describe an output sensitive algorithm which enumerates all configurations of a modular fixturing toolkit capable of immobilizing a given prismatic object.

The fixture vise system consists of two fixture plates mounted on jaws of a vise and reconfigurable modular fixturing elements (Figure 1). Researchers have implemented automated fixture design algorithms for other modular fixture toolkits [1, 3, 7, 11], but such attempts have been hampered by the sheer complexity of the task: most fixture toolkits have more degrees of freedom than degrees of constraint, resulting in an infinite number of valid fixtures for most workpieces. The fixture vise, on the other hand, has exactly the same number of degrees of freedom as degrees of constraint, yielding a finite number of valid fixture configurations. We focus on the fixture vise because a complete enumeration strategy would be impossible without this finite bound.

*Supported by Fannie and John Hertz Fellowship and NSF IRI-9114446

†Supported in part by David and Lucile Packard Fellowship and National Science Foundation Presidential Young Investigator Award (# IRI-8958577).

The fixture vise is akin to the Black & Decker Workmate system, a fixturing system wherein pegs are inserted into holes inlaid in two movable plates. A fixture vise system consists of modular fixture elements (pegs or flatted pegs) inserted into fixture tables which are mounted on both jaws of a vise (Figure 1(a)). Assuming additional devices are used to counter out of plane forces and torques, the fixture vise can immobilize arbitrary prismatic workpieces; since we are dealing with prismatic objects and prismatic pegs, we only need to consider the two-dimensional projection (Figure 1(b)). The term configuration characterizes the state of the fixture vise system; *i.e.,* the peg positions and the separation between the vise jaws. For the remainder of this paper, we will use the term pegs to denote modular fixturing elements.

The fixture vise system has many advantages: first, the fixture vise's simple design enables high precision tooling and reconfigurability due to the peg/hole mechanism, second, closing the vise has the auxilliary property of squeezing the workpiece into the desired position (Figure 2), third, turned upside down, the fixture vise can also be viewed as a *general-purpose reconfigurable* gripper, and, fourth, our fixture design algorithm enables the fixture vise to rapidly handle different workpieces.

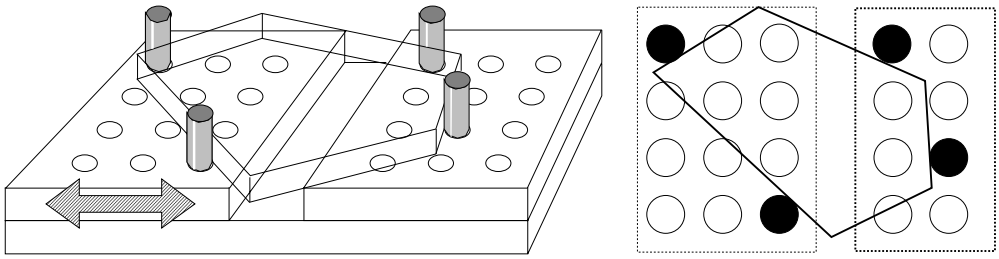


Figure 1: (a) A fixture vise consists of two fixture table jaws which translate along one direction. (b) The two-dimensional projection of the object and fixture vise.

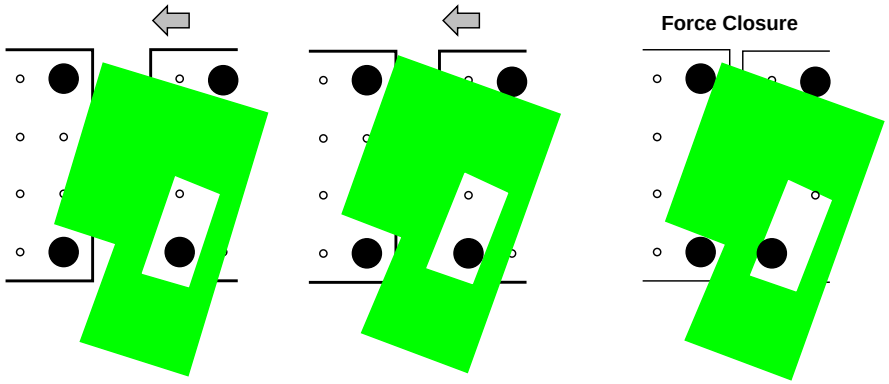


Figure 2: Closing the vise squeezes the workpiece into its appropriate pose.

The main result of this work is a complete algorithm which enumerates all of the fixture configurations capable of immobilizing a prismatic object. The fixture design algorithm is basically an enumeration algorithm which exploits problem-specific pruning heuristics. By enumerating

all possible fixture configurations for a series of workpieces, we hope to find a single fixture configuration which can immobilize all the workpieces, and thereby reduce changeover time. Furthermore, the fixture configurations can be ranked according to a variety of criteria, such as: minimizing the maximal forces required to counter any unit force through the object's center of mass, minimizing the maximal forces required to counter torques along a specific cutting path, etc.

1.1 Related Work

Since fixturing is a fundamental task in assembly, its automation is greatly desired by industry. Fixtures and automated fixture design are discussed at length in the mechanical engineering literature and the manufacturing literature, and they are also related to work done in the field of grasping by the robotics community. Hazen and Wright reviewed the mechanical engineering literature of automated fixturing techniques, components, planning, and execution [4]. Their review implied that most of the mechanical engineering research in automated fixturing systems are based on expert systems; expert systems have the drawback that without geometric analysis, they can only describe "types" of fixturing components, but they cannot propose specific fixture designs. The robotics community has also made significant contributions in terms of grasping, efficiently enumerating useful grasps, and grasp quality metrics; the reader is referred to [2, 8, 12, 14] for an overview of these results.

This research is similar to an algorithm developed by Brost and Goldberg [2]. Their algorithm designed fixtures for prismatic polyhedral objects using a modular fixturing system consisting of fixture elements and a translating clamp, all mounted on a fixture plate. Zhuang *et al.* proved that their fixturing system is incomplete by presenting a class of arbitrarily sized polygons which cannot be fixtured [16]. There have yet to be found any polygons or classes of polygons which cannot be fixtured using an appropriately sized fixture vise toolkit. Furthermore, all of translating clamp fixtures produced by Brost and Goldberg's algorithm correspond to fixture vise configurations where one of the fixture plates contains a single peg.

Recently, Overmars *et al.* introduced a point/edge fixturing paradigm wherein objects are immobilized by a combination of one edge and two point contacts [10]. They prove that their fixturing paradigm is complete in the sense that their system can fixture all appropriately sized polygons whose convex hulls do not have parallel edges. They also present an algorithm for designing modular fixtures for given polygonal objects. Notice that this fixture toolkit poses a simpler design problem because there exist only $O(n)$ orientations for which an edge of the convex hull contacts the fixture edge.

1.2 Overview

In section two we present theoretical background and in section three we present an outline of the algorithm. We detail the two main subroutines of the algorithm in sections four and five. In section six, we present a few automatically designed fixture vise configurations for various workpieces, and conclude by highlighting the results and advantages of this technique. This article elaborates on the fixture vise system introduced by Wallack and Canny [15] and described in more detail in [14].

2 Theoretical Background

In this section, we present five ideas which lay the foundation for the enumeration algorithm: two observations, an argument which shows we can ignore a degenerate case, a computational model of force closure, an argument showing that achieving four simultaneous contacts necessarily requires a total of four degrees of freedom, and a conservative test for checking whether a quartet of edges can admit a force closure configuration.

2.1 Observations

To simplify the analysis, we reduce the problem of achieving simultaneous contact with pegs to the problem of achieving simultaneous contact with points by appropriately transforming the object. Cylindrical or flatted pegs can conservatively be considered point contacts (pegs of zero radius) by appropriately transforming the edge segments (Figure 3). For circular pegs, this reduction is equivalent, but for flatted pegs, this reduction is conservative. This point contact approximation provides a consistent framework for treating both types of contacts. The drawback of this assumption is that we require four pegs per configuration, even though three flatted pegs may provide force closure.

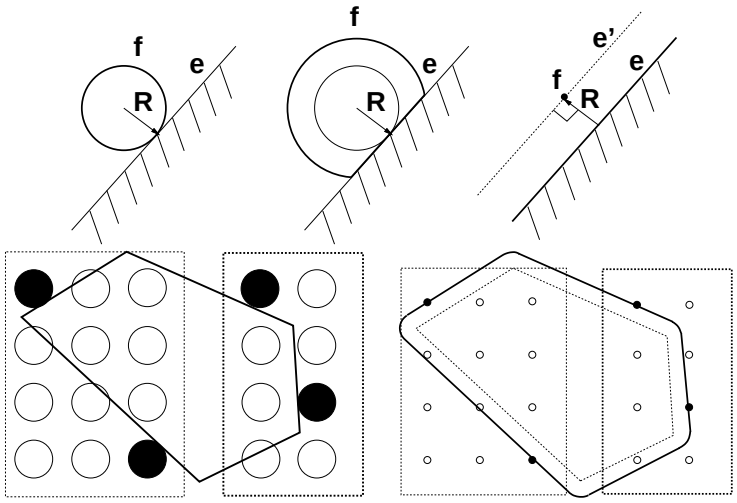


Figure 3: Cylindrical or flatted pegs can be considered point contacts by appropriately transforming the corresponding edge.

The second observation is that there are only two generic modes of simultaneous contact: Type I- where two edge segments of the object contact pegs on each each jaw, and Type II- where three of the object's edge segments contact pegs on a single jaw (Figure 4). We restrict our attention to peg/edge contacts since peg/vertex contacts would more likely cause part deformation, thereby reducing the predictability of part position.

2.2 Degnerate Parallel Edge Pair Case is Irrelevant

In this section, we show that a certain class of fixture configurations will never provide repeatable localization. We focus on this class in particular because it is a degenerate case for our pose parameterization. Since our pose paramaterization involves the extended intersection

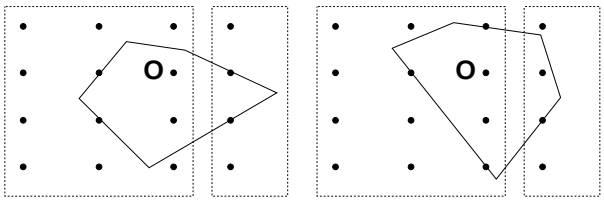


Figure 4: Type I (left): two edge segments of the object contact pegs on each jaw. Type II (right): three edge segments of the object contact pegs on a single jaw.

of two features on the same jaw, therefore, the case of two pairs of parallel edges is degenerate. Since we prove that these cases cannot correspond to reliable fixtures, we do not need to provide a separate parameterization. Figure 5 shows that Type I configurations where the two edges contacting pegs on the left jaw are parallel and the two edges contacting pegs on the right jaw are parallel does not provide repeatable positioning.

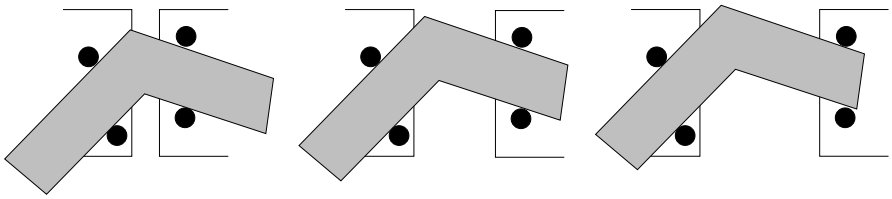


Figure 5: We can ignore the degenerate case where each jaw contacts two parallel edges because the jaw separation is not uniquely determined.

2.3 Verifying Force Closure Mathematically

In this section, we describe a novel mathematical method to check for force closure assuming frictionless point contacts. Consider an object constrained by four point contacts (Figure 6). Forces at the four contact points induce four torques relative to an arbitrary reference point on the object. The torques correspond to the cross-products of the forces f with the lever arms r (equation (1)).

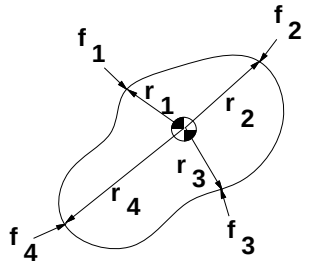


Figure 6: Forces at the four contact points induce four torques relative to a reference point on the object.

$$\tau = f \times r = f_x r_y \Leftrightarrow f_y r_x \tag{1}$$

Force closure in two dimensions, given four point contacts, can be verified using matrices. Consider the four x, y, τ wrenches: $(f_{1,x}, f_{1,y}, \tau_1)^\mathsf{T}$, $(f_{2,x}, f_{2,y}, \tau_2)^\mathsf{T}$, $(f_{3,x}, f_{3,y}, \tau_3)^\mathsf{T}$, $(f_{4,x}, f_{4,y}, \tau_4)^\mathsf{T}$. Force closure corresponds to finding four multipliers $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ such that $\lambda_i > 0$ which satisfy equation (2).

$$\lambda_1 \begin{pmatrix} f_{1,x} \\ f_{1,y} \\ \tau_1 \end{pmatrix} + \lambda_2 \begin{pmatrix} f_{2,x} \\ f_{2,y} \\ \tau_2 \end{pmatrix} + \lambda_3 \begin{pmatrix} f_{3,x} \\ f_{3,y} \\ \tau_3 \end{pmatrix} + \lambda_4 \begin{pmatrix} f_{4,x} \\ f_{4,y} \\ \tau_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{2}$$

We can construct a three by four matrix M consisting of the four contact vectors, and compute the three by three minors M_i obtained by removing the i^{th} column from M (equation (3)). Since the multipliers $(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$ must lie within the nullspace of M, they must be scaled multiples (α) of the minors (M_1, M_2, M_3, M_4) (equation (4)).

$$M = \begin{bmatrix} f_{1,x} & f_{2,x} & f_{3,x} & f_{4,x} \\ f_{1,y} & f_{2,y} & f_{3,y} & f_{4,y} \\ \tau_1 & \tau_2 & \tau_3 & \tau_4 \end{bmatrix} \tag{3}$$

$$(\lambda_1, \lambda_2, \lambda_3, \lambda_4) = \alpha(M_1, M_2, M_3, M_4) \tag{4}$$

Therefore, the four vectors positively span the three-dimensional x, y, θ force torque space (*i.e.*, provide force closure) **if and only if** the minors of M all have the same sign and none of the minors is zero.

2.4 Force Closure Requires At Least Four Degrees of Freedom

Fixturing prismatic objects requires at least four degrees of freedom and this can be shown by a dimension counting argument: force closure requires four simultaneous contacts, and simultaneously satisfying four constraints generically requires at least four degrees of freedom. An object's ability to rotate and translate provides three degrees of freedom (x, y, θ). The fixturing system must provide at least one additional degree of freedom. In the fixture vise system, the separation σ is the fourth degree of freedom.

2.5 Testing Quartets of Edge Segments for Possible Force Closure Contacts

We can prune infeasible edge quartets by testing whether point contacts on four edge segments can possibly achieve force closure. In wrench space, the contacts at all of the edge segment endpoints specify a polytope with the property that force closure is achievable by point contacts on those edge segments (Figure 7) **if and only if** the polytope contains a ball around the origin. Using linear programming, we can test whether a point p lies within the convex hull of a set of vertices $\{v_i\}$ (equation (5)).

$$p \in Conv(\{v_i\}) \Leftrightarrow \{p = \sum_i \lambda_i v_i, \lambda_i > 0, \sum_i \lambda_i = 1\} \tag{5}$$

Furthermore, this test suggests heuristics for determining which quartets are most likely to afford fixtures; these heuristics could be used to order the enumeration of edge quartets, to hopefully generate fixtures more quickly. Possible heuristics include: the volume of the polytope, the radius of the largest enclosed sphere centered at the origin, etc.

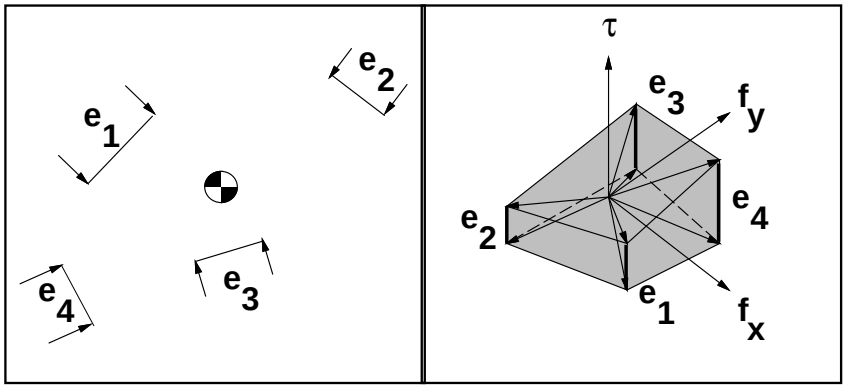


Figure 7: We can test whether a particular quartet of edges can provide force closure, by checking whether the polytope contains a ball around the origin

3 Algorithm Outline

In this section, we present an overview of the main algorithm which enumerates all fixture configurations via a generate-and-test strategy. We define terms, specify the algorithm, and explain the generate and test subroutines in more detail.

3.1 Overview

The *generate* subroutine generates all configurations providing simultaneous contact between (object model) edge segments and fixture pegs, and the *test* subroutine predicts the object poses corresponding to simultaneous contact and then tests for force closure.

3.2 Definitions

Definition 1 *Configuration* specifies the object's pose (X, Y, θ), the positions of the pegs $\vec{\mathcal{F}}$, and the jaw separation distance σ.

Definition 2 e refers to an edge segment, and $\vec{E}$ refers to an ordered quartet of edge segments $\vec{E} = (e_1, e_2, e_3, e_4)$.

Definition 3 $\vec{\mathcal{E}}$ refers to a quartet of *jaw-specified* edge segments where each of the edges is assumed to contact a peg on the left jaw or on the right jaw, *i.e.*, $\vec{\mathcal{E}} = (e_1 \times s_1, e_2 \times s_2, e_3 \times s_3, e_4 \times s_4), s_i \in \{left, right\}$.

Definition 4 $\vec{\mathcal{F}}$ refers to a quartet of positions of the peg centers contacting edge segments $\vec{\mathcal{E}}$. The pegs have been transformed to points by appropriately translating the edge segments by the peg radius.

3.3 Specification

The algorithm expects as input: a two dimensional polygonal model of a workpiece, a subset of edge segments representing the accessible boundary areas, a list of available peg radii, the sizes of the fixture plates mounted on the vise, and the maximum separation between the jaw plates. The algorithm provides as output a list of fixture configurations including the peg positions, part pose, and separation.

3.4 Generate and Test Subroutines

We now explain both the generate and test subroutines in more detail; the generate subroutine is sketched in Figures 8-10.

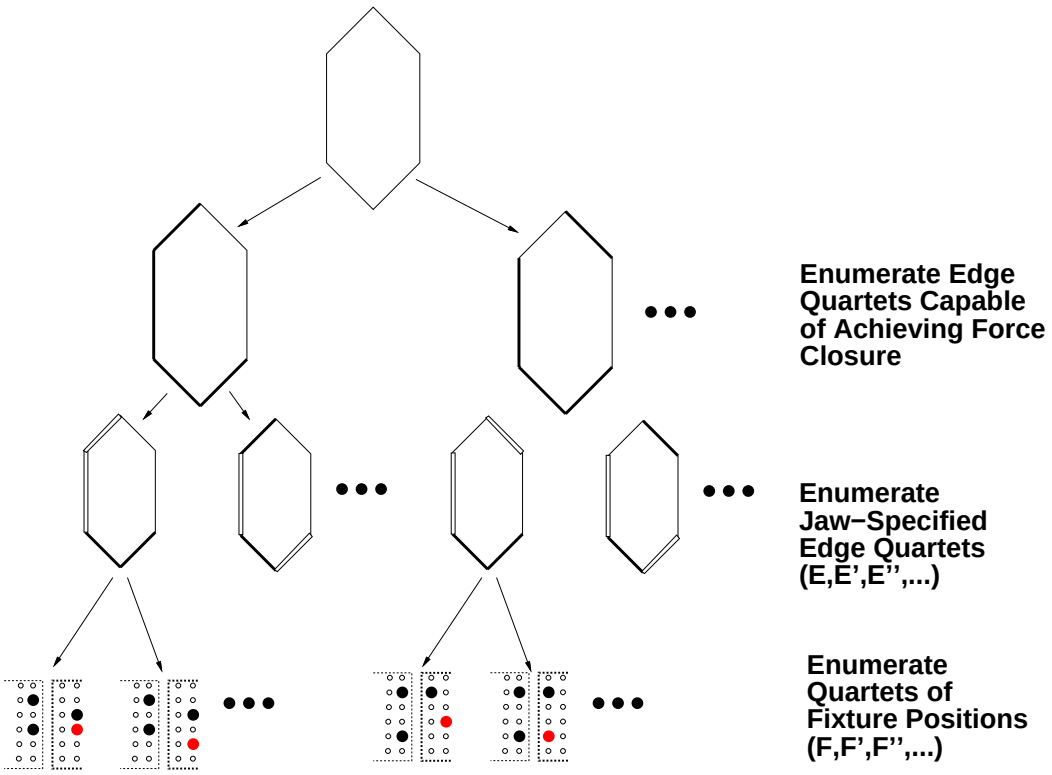


Figure 8: The generate algorithm enumerates quartets of jaw specified edge segments capable of producing force closure, and then, for each edge quartet, enumerates quartets of fixture positions (filled-in circles represent the peg positions).

1. **Generate:**

   (a) Enumerate all jaw-unspecified edge segment quartets (combinations of four edge segments where edge segments can appear twice) $\{\vec{E}, \vec{E}', \vec{E}'', \ldots\}$ for which we can possibly achieve force closure with four point contacts (section 2.5).

   (b) For each quartet of jaw-unspecified edge segments $\vec{E}$, enumerate all different combinations of jaw-specified contacts $\{\vec{\mathcal{E}}, \vec{\mathcal{E}}', \vec{\mathcal{E}}'', \ldots\}$. Up to symmetry, there are seven different situations to be considered: four where three edge segments contact the left jaw, and three where two edge segments contact each jaw.

   (c) For each edge segment quartet $\vec{\mathcal{E}}$, compute fixture configurations $\{\vec{\mathcal{F}}, \vec{\mathcal{F}}', \vec{\mathcal{F}}'', \ldots\}$ providing simultaneous contact with $\vec{\mathcal{E}}$ (Figure 9). The pegs correspond to points after appropriately translating the corresponding edge segments by the peg radius. The fixture vise configurations $\{\vec{\mathcal{F}}, \vec{\mathcal{F}}', \vec{\mathcal{F}}'', \ldots\}$ are enumerated in the manner shown in

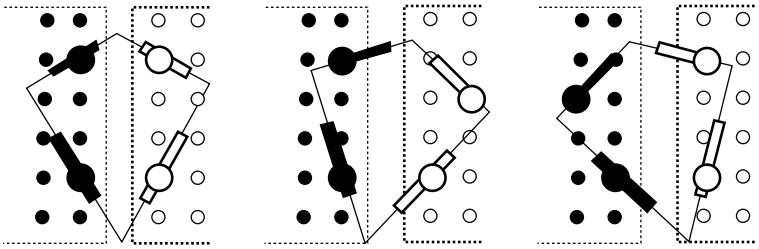


Figure 9: Different peg configurations $\vec{\mathcal{F}}$, $\vec{\mathcal{F}}'$, $\vec{\mathcal{F}}''$ simultaneously contacting edge segments $\vec{\mathcal{E}}$.

Figure 10. We assume that the first peg lies at the origin. Then, we enumerate all of the positions for the second peg. For each peg, we compute bounds on the possible peg positions. For each position of the second peg, we enumerate all of the positions of the third peg. Then we enumerate all of the positions of the fourth peg given the first three peg positions (section 4).

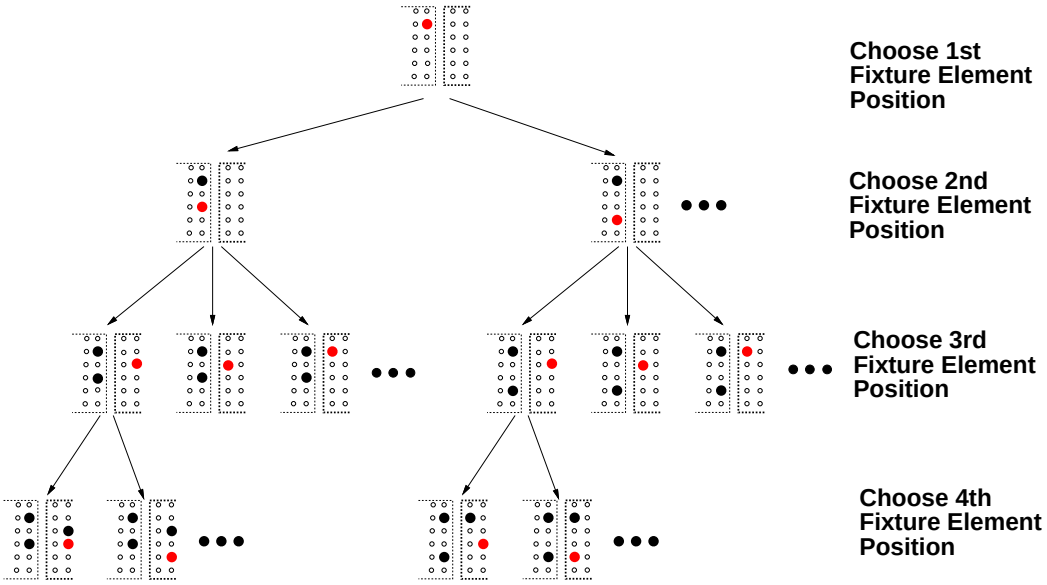


Figure 10: The fixture vise configurations are constructed by first choosing the position of the second peg, then choosing the position of the third peg and so on (filled-in circles represent the peg positions).

2. **Test:**

   (a) Compute all object poses and jaw separations achieving simultaneous contact between edge segments $\vec{\mathcal{E}}$ and peg positions $\vec{\mathcal{F}}$.

   (b) Verify force closure for each simultaneous contact pose.

3.5 Notation

- The operator $\oplus$ refers to the Minkowski sum which can be defined as: $A \oplus B = \{a + b | a \in A, b \in B\}$; we use the Minkowski sum to map $S^1 \times S^1 \to S^1$ and $\mathbb{R}^1 \times \mathbb{R}^1 \to \mathbb{R}^1$

- λ_x and λ_y refer to the column and row spacing of the holes inlaid in the fixture plates on the vise jaws (Figure 11).

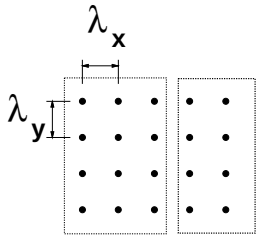


Figure 11: λ_x and λ_y refer to the spacing between the columns and rows for the fixture plates mounted on the jaws of the fixture vise.

4 Enumerating Fixture Configurations Contacting a Quartet of Edge Segments

4.1 Algorithm Outline

This subroutine efficiently enumerates all fixture configurations $\{\vec{\mathcal{F}}, \vec{\mathcal{F}}', \vec{\mathcal{F}}'', \ldots\}$ achieving simultaneous contact with a quartet of edge segments $\vec{\mathcal{E}}$. Our approach utilizes geometric constraints to enumerate A configurations in $O(A)$ time. Enumerating configurations using simpler heuristics would yield extraneous configuration hypotheses, increasing the running time.

Since the configurations are characterized by the relative peg positions, we are free to assume that the first peg is placed at the origin on the left jaw. The second peg position must lie within a wedge of an annulus defined by geometric constraints. The third (fourth) peg must lie in the region swept over by the third (fourth) edge segment while maintaining contact between the first and second edge segments and the first and second fixture pegs (Figure 12). Since the object only has three degrees of freedom (x, y, θ), the object's pose can be characterized by a single variable after the first two peg positions are chosen; the pose is parameterized by χ, which corresponds to the orientation of the extended intersection of the first two features (section 4.2.3). This parameterization simplifies the task of computing the regions swept over by the third and fourth edges.

For Type I configurations, enumerating the positions of the third and fourth pegs is more difficult than enumerating the positions of the second peg because the third and fourth pegs reside on the right vise jaw which can translate freely with respect to the left vise jaw (Figure 13). The regions swept over by the third and fourth edges were computed with respect to the origin. Therefore, the regions provide relative information about the possible x positions of the third and fourth pegs, and absolute positional information about the possible y positions of the pegs.

Because of the lattice structure, we do not need to completely characterize regions swept over by the edge segments; we only need to determine which horizontal rows are swept over by the edge segments, and then determine the contiguous ranges of x values along each row.

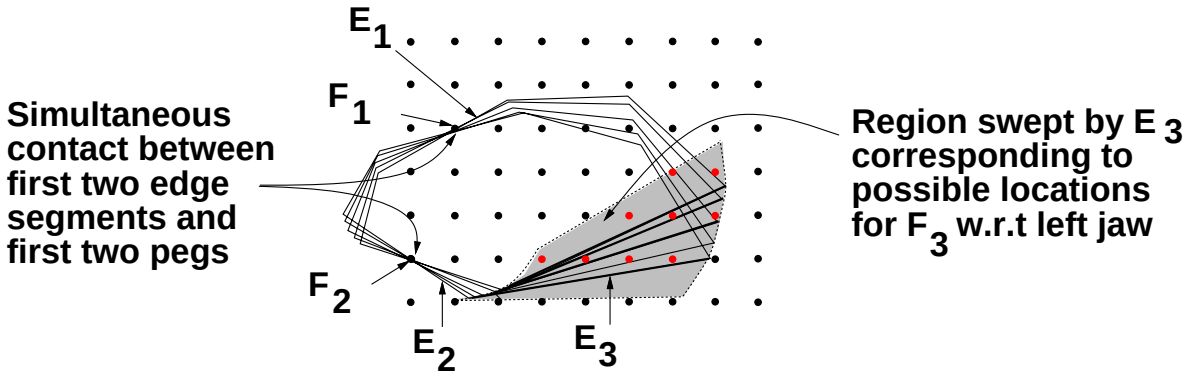


Figure 12: The third peg must lie in the region swept by $\mathcal{E}_3$ while maintaining contact between $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{F}_1$, $\mathcal{F}_2$.

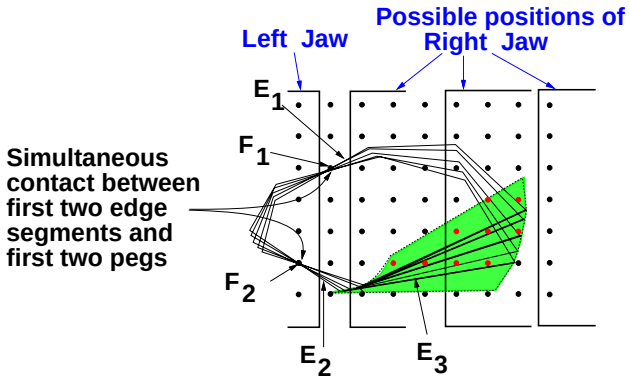


Figure 13: In Type I cases, since the right jaw is free to translate in x , the swept region does not directly correspond to the set of possible positions for the third peg.

4.1.1 Algorithm Listing

Given a quartet of jaw specified edge segments $\vec{\mathcal{E}}$, the algorithm below enumerates all possible quartets of fixture peg positions $\{\vec{\mathcal{F}}, \vec{\mathcal{F}}', \vec{\mathcal{F}}'', \ldots\}$.

1. Assume $\mathcal{F}_1 = (0, 0)$, *i.e.,* the first peg lies at the origin of the left jaw.

2. Compute the set of possible positions for the second peg $\{\mathcal{F}_2, \mathcal{F}_2', \mathcal{F}_2'', \ldots\}$:

   (a) Compute ranges of orientations for which at least one point on each left-jaw specified edge lies to the left of at least one point on each right-jaw specified edge (section 4.2.1).

   (b) Enumerate all of the possible positions for the second fixture peg $\{\mathcal{F}_2, \mathcal{F}_2', \mathcal{F}_2'', \ldots\}$ corresponding to lattice points inside sections of an annulus (section 4.2.2).

3. For each second fixture position $\mathcal{F}_2$, reparameterize the object's pose in terms of a single variable, χ, which implicitly maintains contact between the first two features and the first two pegs. χ characterizes to the orientation of the extended intersection, $P(\chi)$ of $\mathcal{E}_1$, $\mathcal{E}_2$ with respect to the center of a circle C (section 4.2.3). Then, compute the valid ranges of the extended intersection χ: $\{[\chi_{\min}, \chi_{\max}], [\chi'_{\min}, \chi'_{\max}], \ldots\}$

4. For Type I configurations (section 4.2.7):
   Compute the sets of possible positions of the third and fourth pegs:

   (a) Enumerate the fixture rows crossed by each edge by computing the minimal and maximal y values swept over by the third and fourth edges (section 4.2.5).

   (b) For each fixture row crossed, compute the ranges of x positions swept over by the third and fourth edges (section 4.2.6).

   (c) Enumerate pairs of third and fourth peg positions consistent with the range for each respective pair of rows $\{(\mathcal{F}_3, \mathcal{F}_4), (\mathcal{F}'_3, \mathcal{F}'_4), (\mathcal{F}''_3, \mathcal{F}''_4), \ldots\}$.

5. For Type II configuration (section 4.2.8):
   Compute the set of possible positions of the third peg:

   (a) Enumerate the fixture rows crossed by the third edge by computing the minimal and maximal y values achieved by the third edge (section 4.2.5).

   (b) Enumerate all of the lattice points on each row by computing range of relative x positions of the third edge along each row (section 4.2.6) $\{\mathcal{F}_3, \mathcal{F}'_3, \mathcal{F}''_3, \ldots\}$.

   (c) Compute the poses achieving simultaneous contact between the first three edges and the first three fixture elements. For each simultaneous contact pose, transform the fourth edge accordingly, and enumerate one position for the fourth peg $\{\mathcal{F}_4, \mathcal{F}'_4, \mathcal{F}''_4, \ldots\}$ on each row on the left jaw intersecting the fourth edge.

4.2 Algorithm Details

In this section, we sketch the subroutines of the algorithm.

4.2.1 Computing Orientation Ranges Consistent With Left/Right Constraints (2(a))

We constrain the set of possible orientations by exploiting the fact that the contact points (pegs) on the left jaw must lie to the left of the contact points (pegs) on the right jaw; *i.e.,* there must be a point on a left-jaw specified edge which lies to the left of a point on a right-jaw specified edge.

4.2.2 Enumerating Possible Positions of the Second Peg (2(b))

The possible positions of the second peg correspond to lattice points within a wedge of an annulus. The distance between $\mathcal{F}_1$ and $\mathcal{F}_2$ must agree with the range of distances between points from $\mathcal{E}_1$ and $\mathcal{E}_2$, and the orientation between $\mathcal{F}_1$ to $\mathcal{F}_2$ must be consistent with the Minkowski sum of the ranges satisfying the left/right constraints and the ranges of orientations between points on the edges.

4.2.3 Parameterization for Maintaining Contact Between Two Lines and Two Points (3)

One interesting aspect of our algorithm is the pose parameterization which implicitly maintains contact between the first two edges and fixture pegs. Parameterizing the pose by a single variable

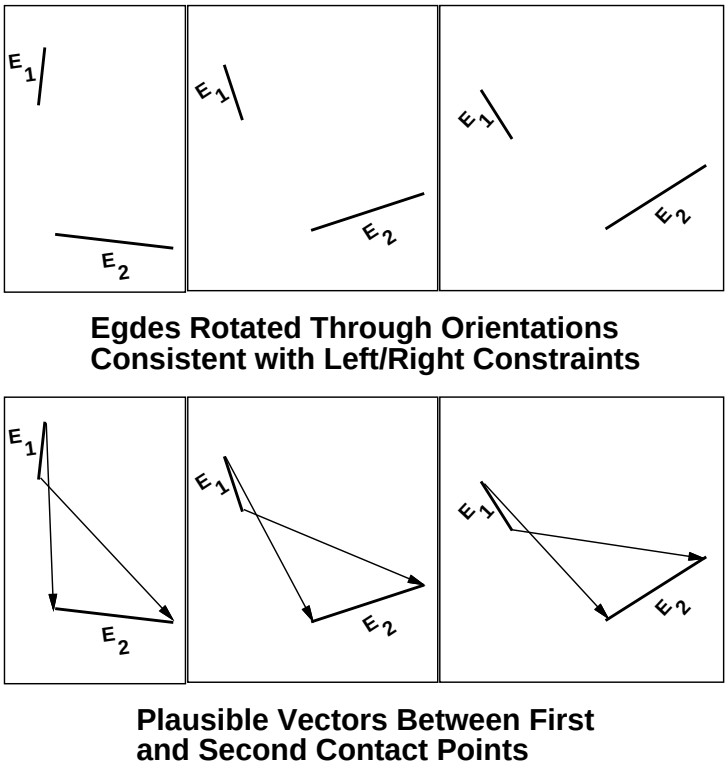


Figure 14: The admissible orientations between the first and second contact point depend jointly upon the ranges of orientations satisfying the left/right constraints, and the ranges of orientations of vectors between the two edges

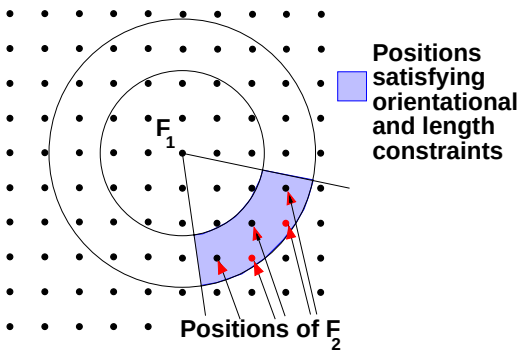


Figure 15: The possible positions for the second peg $\{\mathcal{F}_2, \mathcal{F}_2', \mathcal{F}_2'', \ldots\}$ correspond to the lattice points inside a wedge of an annulus centered at the origin where the minimum and maximum distance constraints are due to the edge segments, and the wedge orientations depend jointly upon the orientations of the vectors between the two edge segments and the orientations satisfying the left/right constraints.

simplifies the task of computing the region swept by the third and fourth edges. The object's pose is parameterized by the position of the extended intersection of the first two edge segments. The valid extended intersections sweep out a circular arc. This parameterization implicitly satisfies the contact constraints because of the geometric property that the interior angle between a point on a circle's boundary and a circular arc remains constant.

Let χ be the orientation of the extended intersection ($P(\chi)$) of the two edge segments, C be the circle including the two contact points $\mathcal{F}_1$, $\mathcal{F}_2$, and let A be an arc defined by the two points (Figure 16). Contact is maintained because the interior angle of A with respect to $P(\chi)$ is a if and only if $P(\chi)$ is on C's boundary (Figure 16). The parameterization breaks down when the two lines are parallel, but this case is irrelevant (section 2.2).

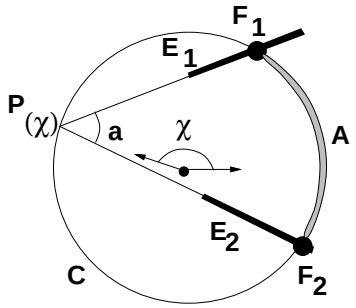


Figure 16: Parameterizing the object's pose by the position of the extended intersection χ on the circle's boundary implicitly maintains contact between $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{F}_1$, $\mathcal{F}_2$.

The valid χ ranges $\{[\chi_{\min}, \chi_{\max}], [\chi'_{\min}, \chi'_{\max}], \ldots\}$ correspond to three types of constraints: first, χ is constrained to be consistent with the left/right orientational constraints; second, the finite-length edge segments, must, in fact, contact the corresponding peg positions, and third, the third and fourth edge segments must intersect particular lattice rows.

4.2.4 Region Swept By Edge $\mathcal{E}$ While Maintaining Contact Between First Two Edges and Pegs (4(a,b),5(a,b))

Next, we compute the region swept over by the third or fourth edge while maintaining contact between the first two edges and the first two pegs. Fortunately, since we are working with a lattice, we do not need to actually compute the region swept over by the third or fourth edge, we only need to compute with the x intersection of the region along the lattice rows which intersect the edge.

4.2.5 Computing Lattice Rows Swept Over by Edge $\mathcal{E}$ (4(a),5(a))

The lattice rows crossed by the edge segment $\mathcal{E}$ are computed by determining the continuous range of y coordinates swept over by $\mathcal{E}$, while maintaining contact between the first two edge segments and two pegs (Figure 17). The lattice rows ($y = k\lambda_y | k \in \mathbb{I}$) crossed by the edge can be determined by computing minimum and maximum y values of the region, which are always due to the vertices v_1, v_2 of $\mathcal{E}$ (Figure 18). The range of y values for each vertex is computed by

formulating $Y(\chi)$, the y position of the vertex as a function of χ and then checking the extremal χ values $\{\chi_{\min}, \chi_{\max}, \chi'_{\min}, \chi'_{\max}, \ldots\}$, as well as the $\chi*$ values corresponding to local extrema of the map: $\{\chi * |(Y'(\chi*) = 0) \wedge (\chi* \in \{[\chi_{\min}, \chi_{\max}], [\chi'_{\min}, \chi'_{\max}], \ldots\})\}$. Finally, we recompute the valid χ ranges consistent with the edge intersecting each lattice row by finding the χ values for which either of the vertices contacts the lattice row; these χ values are found by solving $Y(\chi') = k\lambda_y$.

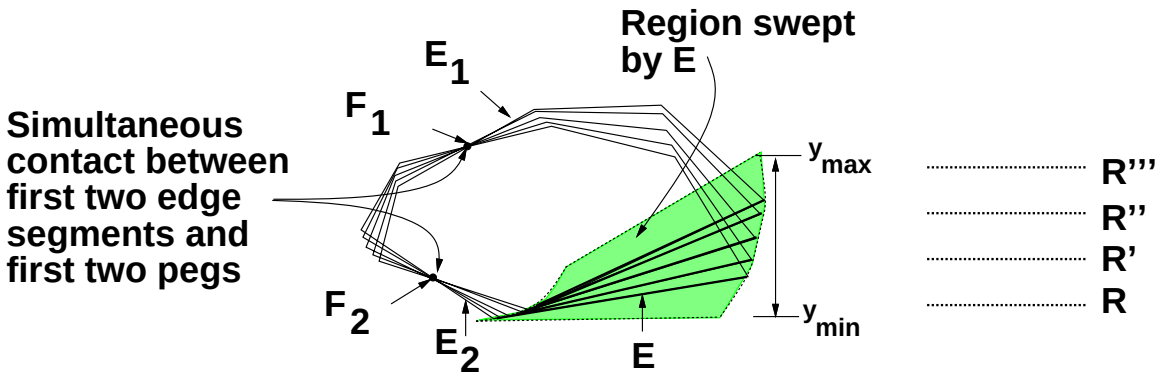


Figure 17: The lattice rows intersecting the region swept by $\mathcal{E}$ while maintaining contact between $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{F}_1$, $\mathcal{F}_2$.

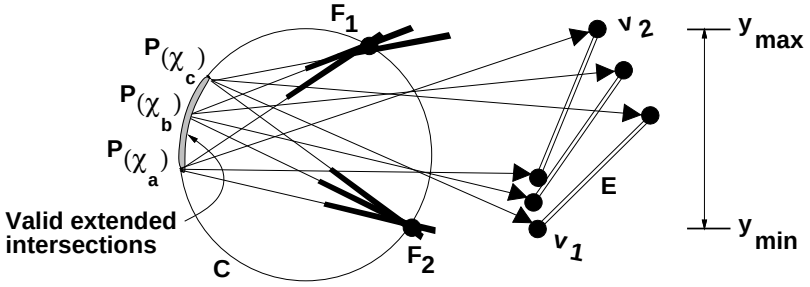


Figure 18: The y range of the region swept by edge $\mathcal{E}$ while maintaining contact between $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{F}_1$, $\mathcal{F}_2$ is found by computing the y ranges of the $\mathcal{E}$'s vertices v_1, v_2.

4.2.6 Computing X Coordinate Range of $\mathcal{E}$ Along Lattice Row $\mathcal{R}$ (4(b),5(b))

In this section we describe the computation of $[X_{\mathcal{E}}^{\mathcal{R}}]$, the range of x coordinates (with respect to the left jaw origin) along a row $\mathcal{R}$ ($y = k\lambda_y$) swept over by edge $\mathcal{E}$ (Figure 19) while maintaining contact between the first two edges and two pegs. Again, we compute the minimum and maximum values of the range $x_{\min}, x_{\max}$ by formulating an algebraic expression. In this case, we formulate the x coordinate the extended intersection $X(\chi)$ between the lattice row $\mathcal{R}$ and the edge $\mathcal{E}$ as a function of χ. We check the extremal values of χ $\{\chi_{\min}, \chi_{\max}, \chi'_{\min}, \chi'_{\max}, \ldots\}$, as well as the χ values corresponding to local extrema of the map: $\{\chi * |(X(\chi*) = 0) \wedge (\chi* \in \{[\chi_{\min}, \chi_{\max}], [\chi'_{\min}, \chi'_{\max}], \ldots\})\}$.

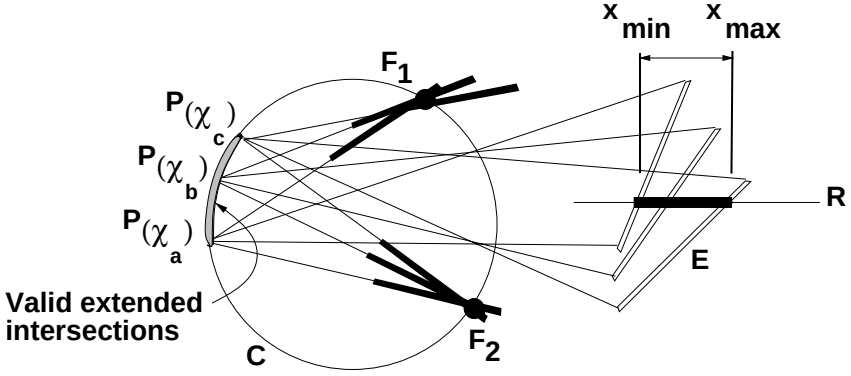


Figure 19: We compute the range $[X^{\mathcal{R}}_{\mathcal{E}}]$ along lattice row $\mathcal{R}$ swept over by $\mathcal{E}$ while maintaining contact between $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{F}_1$, $\mathcal{F}_2$.

4.2.7 Enumerating Peg Positions $\mathcal{F}_3$, $\mathcal{F}_4$ for Type I Configurations (4(c))

Enumerating the third and fourth peg positions involves enumerating lattice points defined with respect to a different coordinate frame. Since we performed all of our computations with respect to the origin on the left jaw, the regions swept by the third and fourth edges were computed with respect to the left jaw; recall that the contact points actually reside on the right jaw which may translate with respect to the left jaw. Therefore, the computed regions do not directly specify the possible peg positions. We compute the possible peg positions for the third and fourth pegs using the *differences* between the ranges of x positions of the swept regions, *i.e.,* the discrete vectors between points in the two ranges.

Consider all pairs of rows corresponding to the two edge segments: $(\mathcal{R}, \mathcal{R}^*)$. Even though the x ranges $[X^{\mathcal{R}}_{\mathcal{E}_3}]$, $[X^{\mathcal{R}^*}_{\mathcal{E}_4}]$ were computed with respect to the left jaw, the differential between the x coordinate ranges $[X^{\mathcal{R}}_{\mathcal{E}_3}] \oplus (\Leftrightarrow [X^{\mathcal{R}^*}_{\mathcal{E}_4}])$ for $\mathcal{E}_3$ and $\mathcal{E}_4$ is invariant with respect to translation in x. Since both edges are expected to lie on the left jaw, we intersect both ranges with $[\max(\mathcal{F}_1.x, \mathcal{F}_2.x), \infty]$. We then enumerate a pair of fixture peg positions $(\mathcal{F}_3, \mathcal{F}_4)$ for each discrete offset $(k\lambda_x \in ([X^{\mathcal{R}}_{\mathcal{E}_3}] \oplus (\Leftrightarrow [X^{\mathcal{R}^*}_{\mathcal{E}_4}]))|k \in \mathbb{I})$ (Figure 20).

4.2.8 Enumerating Peg Positions $\mathcal{F}_3$, $\mathcal{F}_4$ for Type II Configurations (5(c))

For Type II situations, we do not need to compute differences between regions since the third peg resides in the same jaw as the first two. We enumerate all of the lattice points within the regions $k\lambda_x \in [X^{\mathcal{R}}_{\mathcal{E}_3}]|k \in \mathbb{I}$ along rows $\{\mathcal{R}, \mathcal{R}', \ldots\}$, compute the object poses corresponding to simultaneous contact between the first three edge segments and the first three pegs, and then enumerate all of the lattice rows which overlap the fourth edge.

Since we are already know the object's pose, we can prune away contact points on the fourth edge segment which do not provide force closure. Given the object's pose, three contacts, and the edge of the fourth contact, we already know the forces and torques corresponding to three of the contacts, and the direction of the force of the fourth contact; the only unknown is the torque τ_4. In order to achieve force closure, all four of the three by three minors of M must be of the same sign. Three of the minors are linear functions of τ_4, and the fourth is a known constant. We constrain the linear functions to be of the same sign as the constant minor, arriving at a

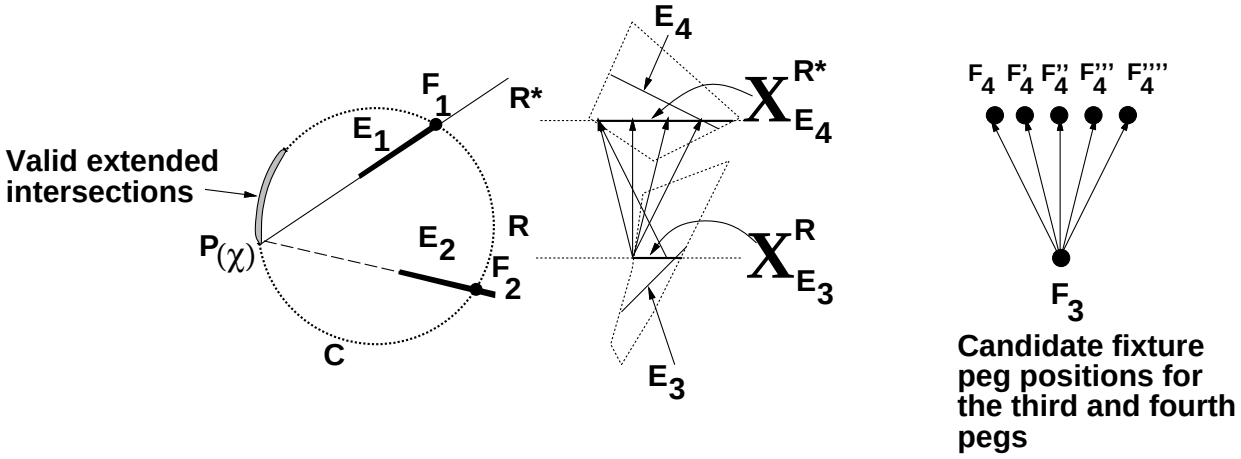


Figure 20: We enumerate all of the discrete vectors corresponding to pairs of plausible fixture positions $\{(\mathcal{F}_3, \mathcal{F}_4), (\mathcal{F}'_3, \mathcal{F}'_4), \ldots\}$ using the x coordinate ranges swept over by the third and fourth edges $X^{\mathcal{R}}_{\mathcal{E}_3}, X^{\mathcal{R}}_{\mathcal{E}_4}$.

range of torques which are necessary and sufficient to produce force closure. From this range of torques, we can compute the range of contact positions along the fourth edge. Pruning the fourth peg position in this manner will improve the algorithm's performance for Type II contacts by only enumerating force closure configurations. Unfortunately, we know of no similar heuristic for Type I cases because the object's pose cannot be determined after choosing the first three configurations because there three contact constraints which involve four degrees of freedom (x, y, θ, σ).

4.3 Algorithmic Complexity

In this section, we derive an upper bound on the algorithm's complexity by presenting an upper bound on A, the number of simultaneous contact configurations. Our worst case analysis involves computing the number of simultaneous contact configurations for a canonical edge quartet, and multiplying this amount by $O(n^4)$ to account for all edge quartets.

The canonical edge quartet is characterized by three values δ, ρ, Δ which characterize the object O. δ refers to the maximum distance between any two points in O, ρ refers to the largest range of distances between points on two edge segments, and Δ refers to the length of the longest edge ($\delta \geq \rho \geq \Delta$, and often $\delta >> \rho$).

First, bound the number of Type II configurations; the second and third pegs must lie within an annulus of area $2\pi(2\delta\rho + \rho^2)$, centered on the first peg; this yields $O((\frac{\delta\rho}{\lambda_x\lambda_y})^2)$ possible positions for the first three contacts. Since the object's pose(s) are determinable given the first three contacts, the fourth peg must lie along one of the rows crossed by the fourth edge (assuming that the fourth peg lies in the leftmost column of the right jaw), yielding $O(\frac{\Delta}{\lambda_y})$ possible peg positions. Therefore, there are at most $O((\frac{\delta\rho}{\lambda_x\lambda_y})^2 \frac{\Delta}{\lambda_y})$ Type II contacts for any quartet of edges.

For Type I contacts, the third and fourth peg positions reside on the right jaw which is free to translate in x; therefore, we cannot simply count lattice positions inside the annulus regions. Again, there are $O(\frac{\delta\rho}{\lambda_x\lambda_y})$ possible second peg positions. We count the number of possible third and fourth peg positions by counting the number of relative (with respect to x translation)

discrete lattice positions between the third and fourth regions. The annulus crosses at most $O(\frac{\delta}{\lambda_y})$ rows, covering $O(\delta)$ x along each row; this yields $O((\frac{\delta}{\lambda_y})^2\frac{\delta}{\lambda_x})$ possible positions for the third and fourth pegs, producing an overall bound of $O(\frac{\delta^4\rho}{\lambda_x^2\lambda_y^3})$ for the total number of Type I configurations for any quartet of edges.

Therefore, the number of simultaneous contact configurations, and consequently, the running time, A, is bounded by configurations of the first type $O(n^4(\frac{\delta^4\rho}{\lambda_x^2\lambda_y^3}))$. Note that if $\lambda_x = \lambda_y = 1$ and $\delta = \rho$ then the complexity is $O(n^4\delta^5)$, where δ is the object's diameter expressed in terms of lattice units. This estimate agrees with the number of configurations found in practice; for example, Table 1 shows that for the Hexnut object with spacing $\lambda_x = \lambda_y = 3.0$, there were 1495 configurations, and with spacing $\lambda_x = \lambda_y = 2.0$, there were 12643 configurations ($(\frac{3}{2})^5 = 7.59375 \approx \frac{12643}{1495} = 8.45686$).

5 Computing Simultaneous Contact Poses

In this section, we describe a technique for computing the object's poses achieving simultaneous contact between edge segments $\vec{\mathcal{E}}$ and pegs $\vec{\mathcal{F}}$ (Figure 21). We algebraically formulate the constraints in terms of the vectors between two contact points on the same jaw. Each pair of contacts on the same jaw provides one relative contact constraint, and we compute the object's pose by intersecting two of these constraints. For Type I situations, each pair of contacts on each jaw supplies a relative contact constraint, and for Type II situations, any two pairwise contact combinations from the left jaw supply two relative contact constraints. For each quartet of edges and pegs, there are at most four distinct Type I poses and at most two distinct Type II poses.

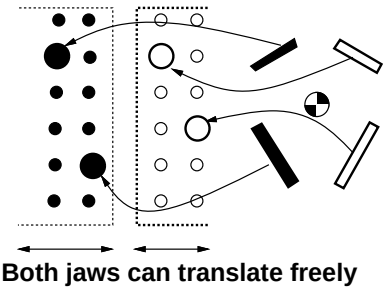


Figure 21: Determine the pose such that edge segments $\vec{\mathcal{E}}$ simultaneously contact pegs $\vec{\mathcal{F}}$ on fixture jaws which freely translate along the x axis.

Given the y positions of the contact points (y_α, y_β), we formulate the problem in terms of achieving a desired difference between the x intersections of two transformed features and two horizontal lines ($y = y_\alpha, y = y_\beta$). We express the position of the extended intersection of the transformed features with the corresponding horizontal lines. Extended intersection functions $I_\alpha(\theta, y)$ (section 5.1) characterize the x coordinate of the extended intersection between a horizontal line and an edge segment rotated around the origin by θ, and translated in y. Next, we formulate extended intersection difference functions which achieve a desired difference $I_\alpha(\theta, y) \Leftrightarrow I_\beta(\theta, y) = \Delta_{\alpha,\beta}$. Finally, we compute the object poses satisfying both pairs of relative contact constraints by solving the multivariate system: $\{I_\alpha(\theta, y) \Leftrightarrow I_\beta(\theta, y) \Leftrightarrow \Delta_{\alpha,\beta} =$

$I_\gamma(\theta, y) \Leftrightarrow I_\delta(\theta, y) \Leftrightarrow \Delta_{\gamma,\delta} = 0\}$. For brevity, we introduce only the extended intersection function (section 5.1).

We intersect these curves algebraically by using a trigonometric substitution $t = \tan(\frac{\theta}{2})$ to arrive at algebraic expressions. We solve that system of non-linear multivariate algebraic equations via resultant techniques, and interestingly, we can compute the intersections exactly since the resultant polynomial is a quartic expression in t. Resultants are a classical algebraic elimination technique which allows one to solve multivariate equations by reducing them to univariate equations [5, 6]; the details for this particular case are discussed in [14, 15].

5.1 Extended Intersection Functions $I(\theta, y)$

Extended intersection functions, $I(\theta, y)$ characterize the x coordinate of the extended intersection between a horizontal line and a corresponding edge segment $\mathcal{E}_\alpha$ rotated around the reference point by θ and translated by y (Figure 22). Extended intersection functions $I(\theta, y)$ are related to $X_{\mathcal{E}}^{\mathcal{R}}$, the range of x coordinates swept by an edge along a row; the major distinction between these abstractions is that the extended intersections $I(\theta, y)$ are parameterized by θ and y, the object's pose, whereas the row contacts $X_{\mathcal{E}}^{\mathcal{R}}$ were parameterized by χ, the orientation of the extended intersection.

$I_\alpha(\theta, y)$ is defined in equation (6) in terms of θ, y, R_α, D_α, α. α is the orientation normal to $\mathcal{E}_\alpha$ pointing outward (Figure 22). R_α refers to the minimum distance from the reference point to $\mathcal{E}_\alpha$, and D_α refers to the difference in y coordinates between the modular row $\mathcal{R}$ and the reference point.

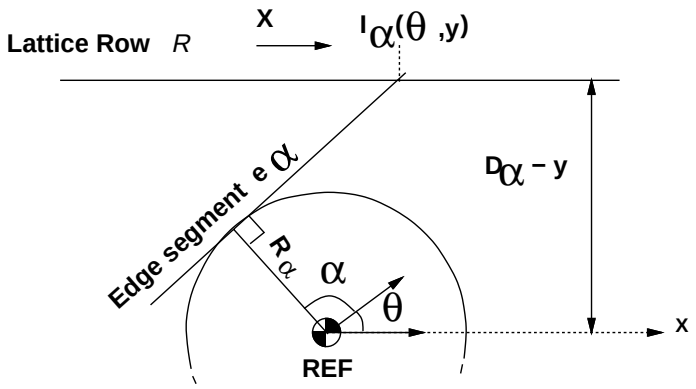


Figure 22: The extended intersection $I(\theta, y)$ between edge segment $\mathcal{E}_\alpha$ and lattice row $\mathcal{R}$.

$$I_\alpha(\theta, y) = R_\alpha \cos(\theta + \alpha) \Leftrightarrow \tan(\theta + \alpha)(D_\alpha \Leftrightarrow y \Leftrightarrow R_\alpha \sin(\theta + \alpha)) \tag{6}$$

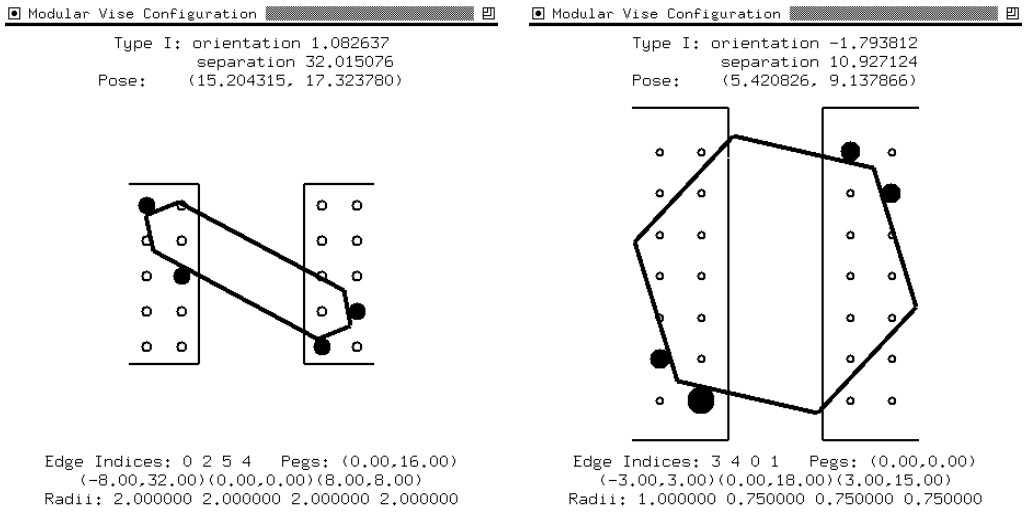
6 Results

In this section, we present experimental results on the number of valid fixture designs for various workpieces, λ_x, λ_y spacings, and peg radii sets (Table 1) as well as present example fixture vise configurations generated by our algorithm. The algorithm was run for three objects: an adapter, a hexnut, and a glue gun casing (the gluegun casing was also examined by Brost and Goldberg [2]). Fixture designs for the adapter and hexnut are shown in Figure 23, and fixture

designs for the gluegun are shown in Figure 24. These statistics account for lattice symmetries by counting each family of translated or symmetric fixtures once.

Object	Spacing (λ_x, λ_y)	Peg Sizes	Total # Configurations	# Type I	# Type II
Adapter	8.0, 8.0	2.0	726	337	389
Adapter	8.0, 8.0	1.5, 2.0	8950	3967	4983
Adapter	6.0, 6.0	2.0	2962	1190	1772
Adapter	6.0, 6.0	1.5, 2.0	36835	14346	22489
Hexnut	3.0, 3.0	1.0	1495	639	856
Hexnut	3.0, 3.0	0.75, 1.0	20912	8320	12592
Hexnut	2.5, 2.5	1.0	3819	1825	1994
Hexnut	2.0, 2.0	1.0	12643	5391	7252
Gluegun	0.75, 0.75	0.25	3685	2473	1212
Gluegun	1.0,1.0	0.25	779	538	241

Table 1: The number of fixture design configurations for a hexagonal object with varying sets of pegs.

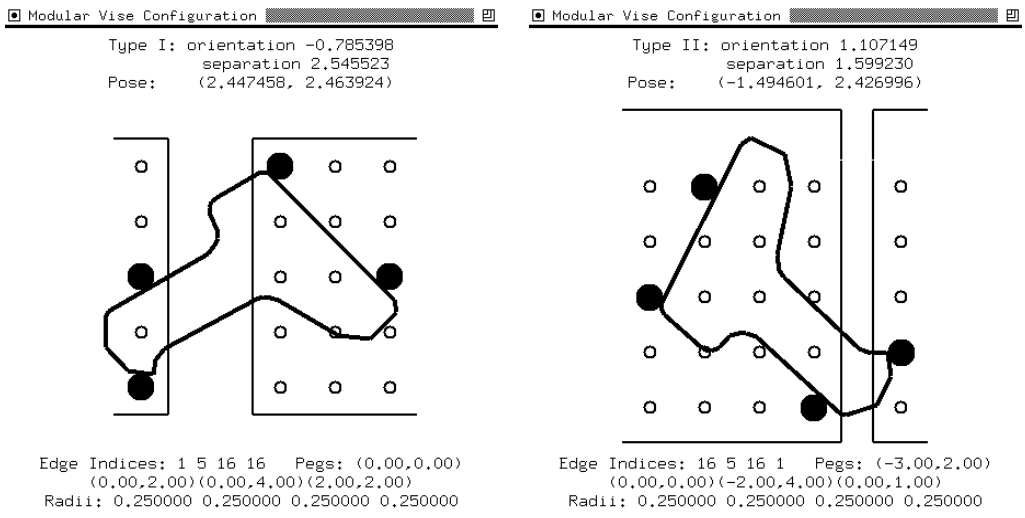


(a) (b)

Figure 23: (a) Type I workholding for an adapter using same radii pegs. (b) Type I workholding for a hexnut using different radii pegs.

7 Conclusion

In this report, we described a complete, efficient algorithm for designing fixtures for prismatic objects. This algorithm can be an integral part of an interactive fixture design system and/or a multi-step fixture planner. We focused on the fixture vise system because it has the property that a given workpiece can be fixtured in only a finite number of ways; this finite bound enabled



(a) (b)

Figure 24: (a) Type I workholding for a glue gun casing. (b) Type II workholding for a glue gun casing.

complete enumeration of all fixture configurations. We utilized a generate and test approach to generate simultaneous contact configurations, and then tested these configurations for force closure.

Acknowledgements

The authors wish to acknowledge: Ken Goldberg for introducing us to the problems of modular fixturing algorithms, Randy Brost, Dina Berkowitz, Steve Burgett, Sanjay Sarma, Jean-Paul Tennant, Joe Gavazza, and Brian Mirtich, and the editor and reviewers for helpful criticisms and suggestions on this report.

References

[1] H. Asada and A. By. Kinematic analysis of workpart fixturing for flexible assembly with automatically reconfigurable fixtures. *IEEE Journal on Robotics and Automation*, RA-1(2):86–94, December 1985.

[2] Randy C. Brost and Ken Y. Goldberg. A complete algorithm for synthesizing modular fixtures for polygonal parts. In *International Conference on Robotics and Automation*, pages 535–542. IEEE, May 1994.

[3] Y-C. Chou, V. Chandru, and M. M. Barash. A mathematical approach to automatic configuration of machining fixtures: Analysis and synthesis. *Journal of Engineering for Industry*, 111:299–306, 1989.

[4] F. Brack Hazen and Paul Wright. Workholding automation: Innovations in analysis. *Manufacturing Review*, 3(4):224–237, December 1990.

[5] F.S. Macaulay. *The Algebraic Theory of Modular Systems.* Stechert–Hafner Service Agency, New York, 1964.

[6] Dinesh Manocha. *Algebraic and Numeric Techniques for Modeling and Robotics.* PhD thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 1992.

[7] A. Markus, Z. Markusz, J. Farka, and J. Filemon. Fixture design using prolog: An expert system. *Robotics and Computer Integrated Manufacturing*, 1(2):167–172, 1984.

[8] Brian Mirtich and John Canny. Easily computable optimum grasps in 2-d and 3-d. In *IEEE International Conference on Robotics and Automation*, San Diego, California, 1994.

[9] B. Mishra. Workholding: Anaylsis and planning. Technical Report No. 259, New York University, Courant Institute of Mathematical Sciences, November 1986.

[10] M. Overmars, A. Rao, O. Schwarzkopf, and C. Wentink. Immobilizing polygons against a wall. In *Symposium on Computational Geometry*, pages 29–38, Vancouver, BC, June 1995. ACM, ACM Press.

[11] Bijan Shirinizadeh. Issues in the design of the reconfigurable fixture modules for robotic assembly. *Journal of Manufacturing Systems*, 12(1):1–14, 1993.

[12] J. C. Trinkle. On the stability and instantaneous velocity of grasped frictionless objects. *IEEE Transactions on Robotics and Automation*, 8:560–572, 1992.

[13] Richard Wagner, Yan Zhuang, and Ken Goldberg. Fixturing 3d polyhedral parts with modular struts. submitted to the 1995 IEEE Conference on Robotics and Automation, 1994.

[14] Aaron Wallack. *Algorithms and Techniques for Manufacturing*. PhD thesis, University of California at Berkeley, 1995.

[15] Aaron Wallack and John Canny. Planning for modular and hybrid fixtures. In *International Conference on Robotics and Automation*, pages 520–527. IEEE, May 1994.

[16] Yan Zhuang, Ken Goldberg, and Yin Chung Wong. On the existence of modular fixtures. In *IEEE International Conference on Robotics and Automation*, pages 543–549, San Diego, California, 1994.