Immobilizing Grasps for Two- and Three-Dimensional Objects

Voorwerpen op hun plaats houden in het vlak en in de ruimte

(met een samenvatting in het Nederlands)

PROEFSCHRIFT

ter verkrijging van de graad van
doctor aan de Universiteit Utrecht
op gezag van de rector magnificus,
prof. dr. W. H. Gispen, ingevolge het
besluit van het college voor promoties
in het openbaar te verdedigen op
maandag 6 november 2006 des middags te 12.45 uur

door

Jae-Sook Cheong

geboren op 22 maart 1971,
te Seoul, Zuid-Korea

Promotor: Prof. dr. M. H. Overmars
Copromotor: Dr. ir. A. F. van der Stappen

Contents

1 Introduction **7**
- 1.1 Analysis and existence . . . 8
  - 1.1.1 General setting . . . 8
  - 1.1.2 Modular setting . . . 9
- 1.2 Grasp synthesis . . . 10
  - 1.2.1 Synthesis of form-closure grasps . . . 10
  - 1.2.2 Synthesis of force-closure grasps . . . 11
  - 1.2.3 Synthesis of second-order immobility grasps . . . 12
- 1.3 Immobility under uncertainty . . . 13
- 1.4 Thesis outline . . . 14

2 Grasp Analyses and Preliminaries **17**
- 2.1 Form closure . . . 17
  - 2.1.1 Analysis in the object plane: Reuleaux's method . . . 17
  - 2.1.2 Analysis in wrench space . . . 19
- 2.2 Second-order immobility . . . 20
  - 2.2.1 Analysis in configuration space . . . 22
  - 2.2.2 Analysis in the object plane for simple polygons . . . 23
- 2.3 Force closure . . . 25
- 2.4 Preliminaries . . . 26
  - 2.4.1 Projections of wrenches . . . 26
  - 2.4.2 Algorithms and data structures for intersection search problem . . . 29

3 Computing All Form-Closure Grasps of a Simple Polygon with Few Fingers **33**
- 3.1 Preliminaries . . . 34
  - 3.1.1 The shapes of wrench sets . . . 34
  - 3.1.2 Intersection search algorithms . . . 35
- 3.2 Computing all form-closure grasps with at most three fingers . . . 35
  - 3.2.1 Two concave vertices . . . 36
  - 3.2.2 One concave vertex and two edges . . . 36
  - 3.2.3 Two concave vertices and one edge . . . 37
  - 3.2.4 Three concave vertices . . . 39
- 3.3 Computing all form-closure grasps for rectilinear polygons . . . 40
  - 3.3.1 Four edges . . . 41
  - 3.3.2 One concave vertex and two edges . . . 43
  - 3.3.3 Two concave vertices . . . 44
  - 3.3.4 Two concave vertices and one edge . . . 45

3.4 Conclusion 48

4 Computing All Form-Closure Grasps of a Planar Semi-Algebraic Set 49
4.1 Preliminaries 50
4.1.1 Algebraic arcs, wrenches and their projections 50
4.1.2 Two-arc-cells and one-arc-cells 51
4.1.3 Intersection search algorithms 52
4.1.4 Computing all grasps on a given set of arcs and vertices 52
4.2 Computing all form-closure grasps with four fingers 53
4.2.1 Four arcs 53
4.2.2 Three arcs 54
4.3 Computing all form-closure grasps with at most three fingers 55
4.3.1 One concave vertex and two arcs 55
4.3.2 One concave vertex and one arc 56
4.3.3 Two concave vertices and one arc 56
4.4 Conclusion 57

5 Computing All Force-Closure Grasps of Polygons and Planar Semi-Algebraic Sets 59
5.1 Preliminaries 59
5.1.1 Edge wrench sets 60
5.1.2 Concave vertex wrench sets 60
5.1.3 Arc wrench sets 61
5.1.4 Intersection search algorithms 63
5.2 Computing all force-closure grasps of polygons 63
5.2.1 Two edges 63
5.2.2 One concave vertex and one edge 64
5.2.3 One concave vertex and two edges 65
5.2.4 Two concave vertices and one edge 65
5.3 Computing all force-closure grasps of planar semi-algebraic sets 67
5.3.1 One concave vertex and one arc 67
5.3.2 Two concave vertices and one arc 68
5.3.3 One concave vertex and two arcs 69
5.4 Conclusion 70

6 Computing All Second-Order Immobility Grasps of Polygons 71
6.1 Preliminaries 71
6.2 Computing all second-order immobility grasps with three fingers 72
6.3 Computing all second-order immobility grasps with two fingers 75
6.4 Conclusion 78

7 Computing All Independent Form-Closure Grasp Regions of Polygons 79
7.1 Preliminaries 79
7.1.1 Our approach 80
7.1.2 Edge wrench patches and vertex wrench sets 80
7.1.3 Intersection search algorithms 81
7.2 Computing all independent form-closure grasp regions of a polygon 81
7.2.1 Four edge patches 81
7.2.2 One concave vertex and two edge patches 82

7.2.3 Two concave vertices and one edge patch . . . 83
7.3 Computing all independent form-closure grasp regions of a rectilinear polygon . . 84
7.3.1 Four edge patches . . . 85
7.3.2 One concave vertex and two edge patches . . . 86
7.3.3 Two concave vertices and one edge patch . . . 86
7.4 Conclusion . . . 88

8 Computing All Form-Closure Grasps of a Rectilinear Polyhedron 89
8.1 Preliminaries . . . 90
8.1.1 Families of faces, concave edges and concave vertices . . . 90
8.1.2 The form closure condition and the projection scheme . . . 97
8.1.3 Intersection search algorithms . . . 100
8.2 Computing all form-closure grasps of a rectilinear polyhedron . . . 100
8.2.1 Seven faces . . . 101
8.2.2 Combinations of faces and concave edges . . . 104
8.2.3 Combinations of faces and concave vertices . . . 115
8.2.4 Combinations of concave vertices, concave edges and faces . . . 117
8.3 Conclusion . . . 120

9 Immobilizing Hinged Polygons 121
9.1 Immobility and robust Immobility . . . 122
9.2 Immobility of a serial chain of hinged polygons . . . 123
9.2.1 Polygons without parallel edges . . . 124
9.2.2 Immobility of hinged polygons with parallel edges . . . 125
9.3 Robust immobility of a serial chain of hinged polygons . . . 127
9.3.1 Robust immobility of polygons without parallel edges . . . 127
9.3.2 Robust immobility of polygons with parallel edges . . . 130
9.4 Immobilizing other types of hinged polygons . . . 130
9.4.1 A cycle of hinged polygons . . . 131
9.4.2 A chain of hinged polygons attached to a wall . . . 131
9.5 Conclusion . . . 132

10 Conclusion and Future Work 133

Chapter 1

Introduction

Automation of manufacturing processes is important and essential in modern industry. In manufacturing processes, immobilization is crucial to fixturing where a part needs to be held rigidly by some fixturing device so that a machine or people can perform operations on it. In robotics and assembly, immobilization is crucial to grasping where an object or part is held by a (robotic) hand to move it to a new location, or to perform simple operations on it. The fixturing devices are called contacts, fingers, or fixels. In this thesis, we will use the term fingers, or sometimes contacts. In order not to damage the part, we place fingers along the boundary of the part in such a way that motions of the part are prevented. Immobilizing an object involves many interesting problems. The followings are some of the fundamental questions.

- Analysis questions: How can immobilization be formally defined and how can it be analyzed?
- Existence questions: What kinds of parts can be immobilized and how many fingers are necessary and/or sufficient to do so?
- Synthesis questions: Given a part and a collection of fingers, can we efficiently construct a single immobilization, a large set of immobilizations, or even all immobilizations of the part with these fingers?

When fingers immobilize an object P, the fingers should be able to resist all external wrenches (i.e. forces and torques), or equivalently, any rigid motion of P (rotation and translation) causes at least one finger to penetrate the interior of P [33]. In 1876, Reuleaux [70] formulated the notion of *form closure*, which is a sufficient condition for immobilization. Intuitively, form closure is more stable immobility compared to second-order immobility, in the sense that the fingers can be slightly perturbed along the contact edges while they maintain the immobility. See Figure 1.2 (a) and (b). Reuleaux provided an analysis for form closure of a planar object, which is based on analysis of

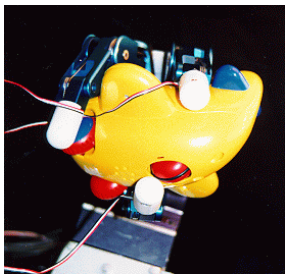


Figure 1.1: Grasping with a robot hand. http://www-static.cc.gatech.edu/gvu/people/faculty/nancy.pollard/grasp.html.

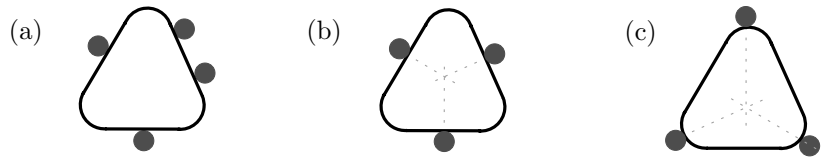


Figure 1.2: (a) An object in form closure. (b) An object in second-order immobility. (c) This object is not immobilized.

possible instantaneous velocity center in the object plane. Form closure can also be analyzed in so-called wrench space or in configuration space, which will be explained in Chapter 2. The analysis in wrench space generalizes to three-dimensional parts.

Note that form closure is for immobility of an object held with frictionless point fingers. When there is friction between the fingers and the part, we use the term *force closure* to refer the immobility of the object. With frictional fingers, one needs fewer fingers to achieve force closure, compared with form closure. In terms of analysis, force closure is very similar to form closure. We discuss force closure further in Chapter 2.

Quite a large class of planar objects can be immobilized with three frictionless point fingers. Figure 1.2 (b) illustrates such an object. Note that the fingers must be positioned carefully to keep the object immobilized. Czyzowicz, Stojmenovic and Urrutia [32, 33] provided a necessary and sufficient condition for simple polygons to be immobilized with three (frictionless point) fingers. Rimon and Burdick [74, 75] generalized the idea of incorporating curvature into theory of *second-order immobility*. The analysis for second-order immobility needs curvature information, while that for form closure use the geometry of the object and the fingers. More precisely, second-order immobility is based on a configuration space analysis of the possible placements of the part, in which the fingers are seen as obstacles, and the current placement should be an isolated point in the collision-free configuration space. More information can be found in Chapter 2.

In many manufacturing environments, modular fixturing systems are often used, where the fixels are constrained to grid points. There has been extensive research performed in this area, as well as in the general setting where fingers can be placed anywhere on the object boundary. In Section 1.1, we will discuss what has been known for existence of grasps in the general setting, as well as in a modular setting. In Section 1.2, grasp synthesis algorithms will be presented. Section 1.3 will cover the immobility for non-rigid parts. Finally, Section 1.4 will describe the contributions of this thesis.

1.1 Analysis and existence

1.1.1 General setting

Reuleaux [70] showed that four (frictionless) fingers are necessary for form closure of a planar object. Mishra et al. [56] and Markenscoff et al. [51] independently showed that four frictionless point fingers are sufficient and often necessary to put a planar object in form closure. They also showed that seven fingers are necessary and sufficient to put a three-dimensional object in form closure. There exist objects that cannot be immobilized, even with infinitely many fingers. They are circles, three-dimensional objects with rotational symmetry, and screws [46, 51, 56, 80].

The result of Czyzowicz et al. [33] implies that any polygon without parallel edges can immobilized. Rimon and Burdick [73] showed that three frictionless fingers with sufficiently flat curvature can immobilize generic piecewise-smooth planar objects, with their analysis in configuration space. Rimon [72] extended these results to 3D; four frictionless fingers with sufficiently

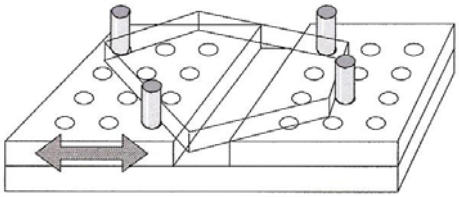


Figure 1.3: A modular fixture vice with two fixture table jaws in [87].

flat curvature can immobilize generic piecewise-smooth three-dimensional objects.

Other than point fingers, one can use edge fingers to put a planar object in form closure. Wentink et al. [92] showed that any polygon which has no edge parallel to the edges of its convex hull can be held in form closure with one edge fingers and two point fingers, and with two edge fingers and one point finger. Moreover, two orthogonal edge fingers and one point finger can achieve form closure for any convex polygon.

Bose et al. [13] used parallel jaw grippers to hold polyhedra securely (clamp them). They showed that all simple convex polyhedra, terrain polyhedra, or orthogonal polyhedra can be held securely regardless of the gripper size. They also showed that every simple polyhedron can be held securely with a gripper of a specific size. Goldberg, and Rao and Goldberg also have studied the problem of grasping polygons [66] and algebraic parts [68] without friction, and also with friction [67].

Czyzowicz et al. [33] also studied immobilization of high dimensional objects. One of the results is that six points suffice to immobilize any polyhedron. They also showed that $2d$ points are sufficient and sometimes necessary to immobilize a d-dimensional polytope, and further, the expected number of points necessary to immobilize a simple d-dimensional polytope is d or $(d+1)$—for convex polytopes, it is $(d+1)$.

1.1.2 Modular setting

In an automated manufacturing system, modular fixturing devices are useful because of reusability and rapid reconfigurability; they offer a limited set of contacts and less freedom to place the contacts. Modular fixturing involves a regular (often square) grid of lattice holes, at which fixture elements (fixels) such as clamps and locators can be placed. Locators are rigid cylinders, while clamps can extend along the grid lines [42]; the object rests against these fixels which constrain its motions. There are many variations such as T-slots where locators can move along the slots [42], or vices where two lattice boards can move along a line [87].

Zhuang and Goldberg [94] showed that three locators and one clamp cannot immobilize all parts with diameter larger than D, for any D (on a unit distance lattice). They also showed that four clamps can fixture rectilinear polygons and convex polygons. Mishra [55] showed that any rectilinear polygon whose edges all have length at least four can be held with in form closure by six clamps. Wentink et al. [92] showed that four point fingers along grid lines can achieve form closure for any polygon without parallel edges and with all edges of length greater than $\sqrt{3}$ on a unit-resolution lattice. Van der Stappen [81] showed that four point fingers on two perpendicular lines can achieve form closure for any polygon without parallel edges.

1.2 Grasp synthesis

1.2.1 Synthesis of form-closure grasps

Mishra et al. [56] gave an algorithm to produce a form closure grasp with six and twelve frictionless point fingers for two and three-dimensional objects. From $O(n)$ fingers that immobilize the object, they repeatedly removed one finger until no finger can be removed without collapsing the immobilization. Four to six fingers remain for planar objects, and seven to twelve for three-dimensional objects. Markenscoff et al. [51] used a maximal inscribed circle for planar objects to find a form closure grasp with four fingers.

Van der Stappen et al. [82] proposed the first algorithm that enumerates all edge quadruples of a polygon that have at least one form closure grasp with four frictionless point fingers. The algorithm is output-sensitive, and it reports all K such edge quadruples in $O(n^{2+\epsilon} + K)$, where ϵ is an arbitrarily small positive number, and n is the number of edges. The algorithm uses range search techniques from computational geometry. Gopalakrishnan and Goldberg [39] studied the problem of immobilizing parts in form closure using two fingers at two concave vertices. They placed two fixtures at external or internal concavities in polygonal parts with polygonal holes and polyhedral parts with polyhedral holes using a gripper with two vertical cylindrical jaws. The algorithm also ranks the solutions based on a quality metric. In this work, one can pursue an alternative quality metric that considers only the local shape around the jaws such as a measure of the "capture region"—the volume of C-space that is guaranteed to converge to the desired grip, and one can try to extend the definition of v-grips to curved parts, and also to more general jaw shapes.

For a three-dimensional object, Meyer [53] presented a way to construct a form closure grasp with seven fingers for a convex polyhedron. It first finds the furthest vertices, and place six fingers on the six faces incident to these vertices (three for each vertex). Splitting one of them into two fingers produces a form closure grasp. It runs in $O(n^{3/2}\sqrt{\log n})$ time, where n is the number of faces of the convex polyhedron. There are also incremental constructions of form closure grasps for three-dimensional objects. Ding et al. proposed algorithms to construct a form closure grasp for three dimensional objects [35] and for three dimensional curved objects [37].

Other than using point fingers, Bose et al. [13] studied the grasping problem with parallel jaw grippers. They presented an $O(n + k)$ time algorithm to compute all valid clamp positions of a simple convex polyhedron, where n is the number of faces, and k is the number of antipodal pairs of features.

Brost and Goldberg [15] studied modular fixturing for polygons using three round locators and one clamp. The algorithm produces all fixture configurations that put the polygon in form closure and obey geometric access constraints. The fixture configurations are ranked by a quality metric. They used a negated cone condition on a force sphere to find a set of wrenches that span the wrench space positively. The algorithm guarantees to find such form closure fixtures, if one exists, in $O(n^5d^5)$ time, where n is the number of edges, and d is the diameter of the object in lattice units. This time bound is later improved to $O(K)$, where K is the output size [14]. This work is extendible to frictional fixturing or a more generalized objects, such as curved objects or three-dimensional objects.

There are other modular fixturing devices, one of which is fixture vise toolkit. It consists of two fixture table jaws on a vise, and pegs, where the table jaws can move towards and away from each other. Wallack and Canny [87] used this system and gave an algorithm that enumerates all form closure fixture vice configurations[1] with four pegs, and the corresponding object poses. The

[1] Wallack and Canny used the term "force closure" in [87], but they assumed frictionless point contacts between the part and the pegs.

object is two and half dimensional prismatic polygonal part, so it is actually two dimensional fixturing problem. The algorithm runs in $O(A)$ time, where A is the number of configurations which simultaneously contact the object.

Wallack and Canny extended their work in [86]. They gave a complete algorithm to automatically design fixtures in the fixture vise toolkit for generalized polyhedral prismatic parts. A generalized polyhedral prismatic part is defined to have a generalized polygonal silhouette, with a boundary composed of linear edges and circular arcs. All configurations holding a polygon in form closure on a vise with four locators can be computed in time $O(n^4d^4r^2)$, where n is the size of the polygon, d is the diameter of the object in lattice units, and r is the largest range of distance between points on two edges. In addition, they showed that the maximum number of form closure configurations is also $O(n^4d^4r^2)$.

Wagner et al. [85] used seven frictionless adjustable-length struts mounted on four boards of regular lattices—one floor and three walls. The struts exert only compressive forces, but no bending moments. To test if given seven points on the part achieve form closure, they used simplified algebraic representation of the test proposed by Goldman and Tucker [38]. They provided an algorithm that enumerates all possible form closure grasps to fixture a three-dimensional polyhedral part in a given pose. All grasps are ranked based on a quality metric.

Overmars et al. [59] used a simple modular fixturing device with an edge fixel, a locator, and a clamp. They gave an output-sensitive algorithm to enumerate all valid modular fixtures for a polygon, while sliding the polygon along the edge fixel, and find the fixtures, using range trees on the angles of the edge normals. It runs in $O(n(n+p)^{\frac{4}{3}+\epsilon}+K)$-time, where K is the output size, n is the number of edges of the polygon, and p is the polygon's perimeter in grid units.

Wang [90], and Wang and Pelinescu [88] proposed algorithms to search for form closure grasps for three-dimensional objects in a point set domain. They chose the possible locations of six locators and one clamp from a set of discrete points. The algorithms are to avoid the prohibitively large cost of an exhaustive search of all combinations of seven locations. Wang [90] used a greedy approach to find a satisfactory solution rather than an optimal one. Wang and Pelinescu [88], on the other hand, first randomly chose seven points, and from these found a new set of seven points that achieves form closure. Ding et al. [36] presented heuristics for selecting fixturing surfaces on a polyhedron.

Instead of locators and clamps, Wentink et al. [91, 92] used an edge fixel and an angle fixel. The angle fixels can be fixed or adjustable. In the paper [92], they gave a linear time algorithm to construct a configuration of an edge fixel, a locator and a clamp, such that they hold some class of rectilinear polygons in form closure. They also showed that all configurations holding a given polygon in form closure with a fixed-angle fixel and a clamp can be enumerated in time $O(n(n+p))$, where n is the number of edges of the polygon, and p is the perimeter in lattice units.

1.2.2 Synthesis of force-closure grasps

Two frictional fingers on a planar object can achieve force closure [54, 58]. Nguyen [58] provided an algorithm to find force closure grasps with two fingers. He also studied the grasps for three-dimensional objects with two soft and three hard fingers. Mirtich and Canny [54] gave an $O(n)$ and $O(n^3)$-time algorithm to construct grasps with two and three rounded frictional fingers for two and three dimensional convex objects respectively.

Computing two-finger force-closure grasps on planar curved objects attracted many researchers [11, 44, 63]. Blake and Taylor [11] presented a more general method to compute force-closure grasps on smooth objects. This method is more general, in the sense that it does not require prior

knowledge of the coefficient of friction. Ponce et al. [63] proposed an algorithm to produce many force-closure grasps with two frictional hard point fingers. They worked on the planar curved objects, such that the boundaries are collections of polynomial parametric arcs. Force closure grasps are characterized by systems of polynomial constraints as in [58]. They compute rectangular regions in the grasp configuration space regions satisfying the force closure constraint, such that this rectangular regions have maximal curve segments where fingers can be positioned independently. Jia [44] used a numerical method using curve functions to tackle the problem of force-closure grasps with two frictional antipodal point fingers. He assumed that the boundary curves are closed, simple, twice continuously differentiable, and planar. He presented an algorithm that finds all pairs of antipodal points on the boundary, up to numerical resolution. Chen and Burdick [20] used *grasping energy function* to compute an antipodal frictional finger grasping for arbitrarily shaped smooth two and three dimensional objects. The energy function is proportional to the square of the distance between the antipodal points. With grasping energy function, finding antipodal points is turned into a constrained optimization procedure. They assumed that the curve representation is uniform cubic B-spline for 2D, and spherical product surfaces for 3D.

Three frictional fingers on planar objects can achieve force closure [49, 50, 60]. Ponce and Faverjon [60] presented an efficient algorithm for computing equilibrium and force-closure grasps with linear constraints. They provided a sufficient condition to compute such grasps. Li et al. [49, 50] provided a necessary and sufficient condition for three fingers on polygons to achieve force closure. Cornellá and R. Suárez [28] computed optimal position for the fourth finger, for given three finger positions on a polygon, such that they achieve force closure.

A three-dimensional object can be immobilized with four frictional fingers [61]. Ponce et al. [61] gave an algorithm to compute concurrent grasps of a polyhedron, by computing the stable grasp regions in configuration space, which is equivalent to eight dimensional projection of an eleven dimensional polytope. To avoid heavy computations and checking the combinations of faces, heuristics can be used [64, 65]. Prado and Suárez [64, 65] find three-finger force-closure grasps on a polyhedron fast using heuristics.

1.2.3 Synthesis of second-order immobility grasps

Czyzowicz et al. [32, 33] gave a linear time algorithm to find a set of three points that immobilize a convex polygon, and an $O(n \log n)$-time algorithm for a polygon without parallel edges, using a maximal inscribed circle of the polygon.

Rimon and Burdick [71, 73, 76] showed that two fingers can immobilize a two dimensional object, and four fingers can immobilize any generic polyhedral or smooth three dimensional object, when the fingers are allowed to have an arbitrary curvature. They also established how the curvature affects second-order immobility in configuration space, by comparing it with form closure. They proposed the name *second-order immobility*; form closure is equivalent to first-order immobility in their terms. With Rimon and Burdick's mobility theory, Ponce et al. [62] synthesized all immobilizing grasps of a part bounded by polynomial splines. They presented an algorithm which uses exact cell decomposition and homotopy continuation techniques to construct an explicit description of the immobilization regions in the contact configuration space.

Sudsang et al. [83] used a modular device to hold a three-dimensional object in second-order immobility with four fingers. The device consists of two parallel plates with locator holes along a rectangular grid. The distance between the plates can be adjusted continuously. They provided simple sufficient conditions for immobilization and stability of polyhedra, and proposed efficient geometric algorithms to enumerate all stable immobilizing grasps.

1.3 Immobility under uncertainty

When fixturing an object in practice, the object cannot always be positioned as it should be. Naturally, a robust fixturing plan that works even with positioning error is desirable. Bone and Du [12] introduced a grasp planning method based on a new metric for measuring the sensitivity of a grasp to positioning errors. Nguyen [58] showed how to compute a maximal independent region on a polygon for two frictional fingers, where fingers can be independently positioned maintaining force closure. Ponce and Faverjon [60] and Ponce et al. [61] reported maximal independent regions by linear optimization within the valid configuration space regions for planar objects [60] and for polyhedra [61]. Cornellá and Suárez [29, 30, 31] also provided algorithms to produce independent regions for frictional and frictionless fingers on planar objects.

Another type of uncertainty that we should deal with in manufacturing is the shape uncertainty; in most cases, the surface will be slightly different from the nominal boundary. To be able to use the precomputed fixturing plan in practice, the plan has to work for those slightly different objects. One question that we can immediately ask is how different the shape can be to be able to use the computed fixture plan, i.e. what the *tolerance* for a given fixturing plan is. To answer this question, we first need a realistic model for the tolerance.

Requicha [69] proposed a mathematical theory of tolerancing that formalizes and generalizes current practices. It is a suitable basis to incorporate tolerances into GMSs (geometric (solid) modeling systems). A *tolerance specification* is a collection of geometric constraints on the surface features of an object. An object is in tolerance if its surface features lie within tolerance zones, which are regions of space constructed by expanding or shrinking the object's nominal boundaries.

Other than the theoretical model, there is an experimental tolerance modeler proposed by Moroni and Requicha [57]. They also described an associated API (Application Programming Interface) that hides the modeler's internal details. The tolerance modeler can be attached to any nominal-geometry modeler through a tolerance adaptor.

Akella and Mason [4] used a tolerance model in which the center of mass of the object and its vertices lie in circular tolerance zones around their nominal positions, for the sensor-based and sensorless part orienting problem. Chen et al. [21] used a similar tolerance zone model to orient convex polygonal parts on a conveyor belt and to fixture convex polygonal parts using a right angle fixture and one clamp. The tolerance zones used for fixturing are rectangles, while those for orienting are circles. They gave an $O(n)$-time algorithm for testing if an n-sided part is in the tolerance class for orienting and fixturing, and $O(n^2)$- and $O(1)$-time algorithm to compute the maximum tolerance zone size for orienting and fixturing respectively.

Another tolerance model regards the part as a toleranced polygon, whose edges can lie in a band around the edges. Cazals and Latombe [17] used this model, and they gave an efficient algorithm to compute the tolerance zone size.

Brost and Peters [16] presented an implemented algorithm that automatically designs fixtures and assembly pallets to hold three-dimensional parts, which is robust to force and part-shape variations, easy to load, and economical to produce. The algorithm finds the global optimum design. Wang [89] analyzed the problem of characterizing the accuracy of deterministic localization of fixtures using statistical framework. More details about tolerancing, metrology and techniques can be found in [78, 84, 93].

Fixturing deformable objects is another interesting problem. In the motion planning community, some research has been conducted including deformation [5, 43, 47]. Recently Gopalakrishnan and Goldberg [40, 41] proposed the concept of deform closure for immobilizing a deformable object. They also proposed a numerical algorithm to produce an approximation to the optimal deform

closure grasp with two contacts [40]. Another type of non-rigid bodies are objects connected by hinges. Motion planning community has some related work (on articulated robots), but no work has been found related to fixturing, as far as we know.

1.4 Thesis outline

This thesis attempts to provide efficient computations of all immobilizing grasps of two and three dimensional objects in the non-modular setting. In particular, we enumerate all form and force closure grasps of a polygon and a planar semi-algebraic set. In addition, we also report all second-order immobility grasps with two and three fingers on polygons. We also enumerate all independent form-closure grasp regions of a polygon. Efficient computation of arbitrary three dimensional objects is difficult, even for a polyhedron. The easiest (but still difficult) case to consider is that of a rectilinear polyhedron. We propose an algorithm to report all form-closure grasps and all independent form-closure grasp regions of a rectilinear polyhedron. Finally, we present a way of constructing some grasps of a serial chain of hinged polygons as a case study on non-rigid objects.

An intuitive analysis of form closure of a planar object takes place in the two-dimensional plane of the object itself. For synthesis of form or force closure grasps, however, the formulation in wrench space turns out to be more convenient and powerful. The form-closure condition on wrenches is transformed into specific combinations of geometric intersection problems, and these will be used to compute all form and force closure grasps of two and three dimensional objects. Throughout the thesis, we let n denote the number of edges of a polygon, or arcs of a semi-algebraic set, and m denote the number of concave vertices, unless stated otherwise.

Given a combination of concave vertices and edges or arcs, we can compute all form-closure grasps in constant time [82]. As a consequence, the combinatorial complexity of computing all form-closure grasps is determined by the number of such combinations. Clearly we would like to minimize the time spent on the combinations that admit no form-closure grasp. Our goal is to report all combinations of concave vertices and edges or arcs that allow at least one form-closure grasp, in an efficient and output-sensitive way. Almost all algorithms between Chapter 3 and 8 are efficient and output-sensitive, which means that their running times largely depend on the actual size K rather than the (often much larger) maximum size of the output.

Chapter 3 to 5 are about computing all immobilizing grasps for two-dimensional objects. More specifically, we compute all form-closure grasps of a polygon and a planar semi-algebraic set with at most four frictionless point fingers in Chapter 3 and in Chapter 4 respectively. In Chapter 7, we compute all independent form-closure grasp regions for polygons and rectilinear polygons. In Chapter 5, we use the approach in Chapter 3 and 4 to compute all force-closure grasps of a polygon and a planar semi-algebraic set. We also compute all second-order immobility grasps of a polygon and a planar semi-algebraic set in Chapter 6. The approach in wrench space is pushed into three-dimensional space, and in Chapter 8, we compute all form-closure grasps, and all independent form-closure grasp regions of a rectilinear polyhedron. Finally, in Chapter 9, we study the problem of immobilizing a serial chain of hinged polygons.

In Chapter 3, we propose efficient algorithms to compute all form-closure grasps of polygons using concave vertices. An efficient output-sensitive algorithm to compute all form-closure grasps of polygons on three or four edges has already been proposed by van der Stappen et al. [82], as mentioned earlier. Informally speaking, such vertices allow to have two fingers at the price of one, as a finger at a concave vertex can be regarded as lying on both incident edges or arcs. Computing all form-closure grasps with fingers at concave vertices was first studied by Gopalakrishnan and Goldberg [39], who gave an $O(m^2)$ time algorithm to find all K concave vertex pairs that allow

a two-finger form-closure grasp. We improve this to $O(m^{4/3}\log^{1/3} m + K)$ in Section 3.2.1. All form-closure grasps with three fingers can be reported in $O(n^2\log^4 n + K)$-time. When a polygon is rectilinear, we need less computation. In Section 3.3, we compute all combinations of edges and concave vertices of a rectilinear polygon that yield form-closure grasps with three or four point fingers efficiently. More specifically, we compute all such sets of: (i) four edges in $O(n\log n + K)$ time; (ii) one concave vertex and two edges in $O(n\log n + K)$ time; (iii) two concave vertices in $O(m\log^2 m + K)$ time; (iv) two concave vertices and an edge in $O(nm\log n + K)$ time.

Chapter 4 is about efficient computations of all form-closure grasps of a planar semi-algebraic set P with at most four frictionless point fingers. The boundary of P consists of n algebraic arcs with a constant degree, and P has m concave vertices. We enumerate all combinations of (i) four arcs, (ii) three arcs, (iii) one concave vertex and two arcs, (iv) one concave vertex and one arc, and (v) two concave vertices and one arc, such that three or four fingers on these combinations yield at least one form-closure grasp. Let ε be an arbitrarily small positive number. We can handle the cases stated above in the following time complexities: case (i) $O(n^{8/3}\log^{1/3} n + K)$; case (ii) $O(n^{5/2+\varepsilon} + K)$; case (iii) $O(n^2m^{1/2+\varepsilon} + K)$; case (iv) $O(nm)$ or $O(n^{3/2+\varepsilon} + K)$; case (v) $O(nm^2)$ or $O(n^{2+\varepsilon} + K)$. Case (iv) and (v) have multiple choices of time complexities, depending on the size of m in comparison to n.

In Chapter 5, we extend the approach used in Chapter 3 and 4 to compute all force-closure grasps of a polygon and a planar semi-algebraic set. For a polygon, we compute the combinations of (i) two edges, (ii) one concave vertex and one edge, (iii) one concave vertex and two edges, and (iv) two concave vertices and one edge, such that two or three fingers on these combinations yield at least one force-closure grasp. We can enumerate all these combinations in (i) $O(n^{4/3}\log^3 n + K)$ time, (ii) $O(n^{4/3}\log n + K)$ or $O(n^{4/3}\log^3 n + K)$ time, (iii) $O(n^2\log^4 m + K)$ time, and (iv) $O(m^2n)$ or $O(n^{4/3}\log^3 n + K)$ or $O(n^2\log^4 n + K)$ time respectively. For a planar semi-algebraic set, we compute the combinations of (i) one concave vertex and one arc, (ii) two concave vertices and one arc, and (iii) one concave vertex and two arcs, such that the fingers on these combinations yield at least one force-closure grasp. We can enumerate all these combinations in (i) $O(nm)$ or $O(n^{3/2+\varepsilon} + K)$ time, (ii) $O(nm^2)$ or $O(n^{2+\varepsilon} + K)$ time, and (iii) $O(n^2m^{1/2+\varepsilon} + K)$ time respectively.

In Chapter 6, we enumerate all second-order immobility grasps with two and three frictionless point fingers for a polygon. More precisely, we compute all K edge triples that yield a second-order immobility grasp with three fingers in $O(n^2\log^3 n + K)$ time, and all K pairs of an edge and a concave vertex that yield a second-order immobility grasp with two fingers in $O(n\log^4 m + (nm)^{2/3}\log^{2+\varepsilon} m + K)$ time. We use the necessary and sufficient geometric condition for second-order immobility grasps of a polygon, proposed by Czyzowicz, Stojmenovic and Urrutia [33].

In Chapter 7, we provide output-sensitive algorithms to report all sets of independent form-closure grasp regions of a specified width ε on edges of a polygon, and a rectilinear polygon. The independent form-closure grasp regions are such that any placement of three or four frictionless point fingers inside these regions will put the polygons in form closure. The practical implication is that we yield form-closure grasps that are insensitive to misplacements of each of the individual fingers by a distance of $\varepsilon/2$. For a polygon, we enumerate (i) all K edge patch quadruples, (ii) all K triple of one concave vertex and two edge patches, and (iii) all K triples of two concave vertices and one edge patch, such that three or four fingers on these combinations yield at least one form-closure grasp. We can report these K sets in (i) $O(n^{8/3}\log^{O(1)} n + K)$ time, (ii) $O(n^2\log^4 m + K)$ time, and (iii) $O(m^2n)$ or $O(n^2\log^6 n + K)$ time. For a rectilinear polygon, we enumerate (i) all K edge patch quadruples, (ii) all K triples of one concave vertex and two edge patches, and (iii) all K triples of two concave vertices and one edge patch, such that three or four fingers on these

combinations yield at least one form-closure grasp. We can report these K sets in (i) $O(n \log n+K)$ time, (ii) $O(n \log n + K)$ time, and (iii) $O(n \log^2 n + m^2 \log n + K)$ time.

Chapter 8 is on computing all form-closure grasps and all independent form-closure grasp regions of a rectilinear polyhedron. A form-closure grasp needs at least seven lines of force. We propose algorithms to compute all combinations of edges, concave edges and concave vertices, such that four to seven fingers on these combinations achieve form closure.

In this chapter, we report all sets of (i) seven faces, (ii) faces and concave edges, (iii) faces and concave vertices, and (iv) faces, concave edges and concave vertices, such that four to seven frictionless point fingers on these sets yield a form-closure grasp. All K sets among all such sets can be reported in (i) $O(n^2K' \log^4 n + K)$ time, (ii) $O(n^2K' \log^4 n + K)$ time, (iii) $O(n^2 \log^2 n + nK' \log^4 n + K)$ time, and (iv) $O(n^2 \log^2 n + nK' \log^4 n + K)$ time, where n is the number of faces, and K' is the size of the intermediate output. All of them except one are sensitive to K' and K; the exception is sensitive to K' only.

When the object is not rigid, the complexity of an immobilizing grasp can be high. For example, when polygons are connected by hinges at the vertices, the configuration space of this set of objects has a high dimension, which is proportional to the number of polygons. In Chapter 9, we study the problem of immobilizing hinged polygons in a given placement with frictionless point fingers. We define new notions of *immobility* and *robust immobility*, which are comparable to second-order immobility and to form closure for a single object. Robust immobility is an immobilization that is insensitive to small perturbations of fingers along the edges. Notice that this perturbation is arbitrarily small; we do not guarantee that one can perturb each finger by a given value ε, which is the difference between robust immobility and independent form-closure grasps. We show, by construction, that $(n + 2)$ frictionless point fingers suffice to immobilize any serial chain of $n \neq 3$ polygons without parallel edges; it is unclear whether five fingers can immobilize three hinged polygons. At most $(n + 3)$ fingers suffice to immobilize a serial chain of n arbitrary polygons. We also show, by construction, that $\lceil \frac{6}{5}(n + 2) \rceil$ and $\lceil \frac{5}{4}(n + 2) \rceil$ fingers suffice to robustly immobilize a serial chain of n hinged polygons without, and with parallel edges respectively.

Chapter 2

Grasp Analyses and Preliminaries

In this section, we introduce the notions of form closure (Section 2.1), second-order immobility (Section 2.2.2) and force closure (Section 2.3), and the corresponding analyses. In Section 2.4, we present a condition for form closure, and the intersection algorithms and the data structures used in this thesis.

2.1 Form closure

In this section, we present two types of analyses for form closure: one is Reuleaux's method [70] on the object plane, and the other is in wrench space. Reuleaux's method is more intuitive, but it only applies to planar objects. The analysis in wrench space, on the other hand, is for two and three dimensional objects. Form closure can also be analyzed in configuration space [74, 75], which we will explain in Section 2.2.1. We do not include this analysis, because its major contribution is to explain how curvatures affect a set of fingers to achieve second-order immobility.

Screw theory provides yet another way of modeling the effect of force. It is used by many researchers for kinematic analysis of body motions [6, 8, 27, 77]. Unfortunately, it does not provide a nice characterization to check immobility as far as we know, therefore we do not discuss screw theory further in this thesis. Ponce et al. [61] provided an excellent explanation of screw theory.

2.1.1 Analysis in the object plane: Reuleaux's method

Every infinitesimal motion can be seen as a rotation around a point in a counterclockwise/clockwise direction [70]. When a point finger is in the interior of a straight edge, the normal line divides centers of counterclockwise and clockwise rotations. The left side of the normal line[1] has centers

[1]The boundary line is not included.

(a) (b)

Figure 2.1: (a) When a point finger is in the interior of an edge. (b) When a point finger is at a concave vertex. The points on the thick part of normal line allow clockwise and counterclockwise rotations, while those on the thin part of the normal line do not allow any.

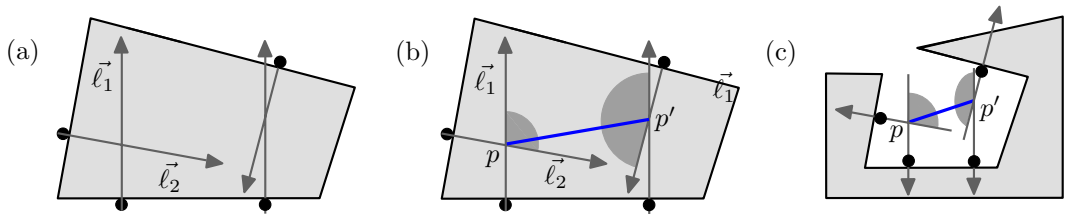


Figure 2.2: (a) A polygon P in form closure with four fingers. (b) The line $\overline{pp'}$ is in the positive cones of the normal lines. (c) $\overline{pp'}$ is in the negative cones of the normal lines.

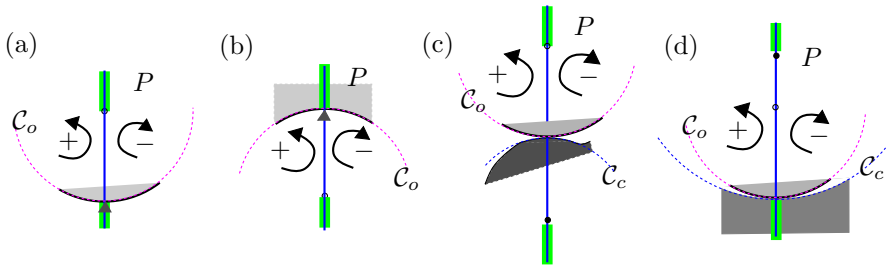


Figure 2.3: (a) (b) Possible rotational centers on the normal line when a point finger is on the curved boundary of an object. (c) (d) Possible rotational centers on the half planes when a curved finger is on the curved boundary of an object.

of counterclockwise rotations and the right side has those of clockwise rotations, when facing the interior of the object P from the finger. (See Figure 2.1.) In other words, for the point finger rules out all infinitesimal rotations of P except counterclockwise rotations around a point on the left side of the normal line, and clockwise rotations around a point on the right side. When a point finger is at a concave vertex, it induces two normals to the incident edges at the vertex. The common regions of counterclockwise and clockwise rotations induced by the two normals have centers of possible rotations as in Figure 2.1 (b). The object P is in form closure, if and only if the oriented half planes induced by the point fingers have an empty intersection. See Figure 2.2 (a). This can be seen in another way [58]. Two directed lines $\vec{\ell}_1$ and $\vec{\ell}_2$ divide the plane into four regions. Let the intersection point of $\vec{\ell}_1$ and $\vec{\ell}_2$ be the origin. We take the region bounded by $\vec{\ell}_1$ and $\vec{\ell}_2$ with the outgoing directions, and call it *positive cone*—see Figure 2.2 (b). Similarly, we call the region bounded by the incoming rays *negative cone*. We pair the normal lines, and connect the intersection points p and p' by a line segment. The object P is in form closure, if and only if $\overline{pp'}$ is in the positive cones of the normal lines, or in the negative cones of the normal lines. Figure 2.2 (b) and (c) illustrates these cases.

What about the points on the normal line? Which rotations around these points are allowed? It turns out that it depends on the curvatures of P and the fingers. In other words, the information about these curvatures can be included in the half plane analysis. When the curvatures of the object boundary and the finger at the contact are considered, we can distinguish which rotation is possible around the points on the normal line—the other part is the same as with the case of a point finger along a straight edge. Let $\mathcal{C}_o$ and $\mathcal{C}_c$ be the tangent circle at the finger of the object boundary and the contact respectively. The points between the finger and the center of $\mathcal{C}_o$, and those between the finger and the center of $\mathcal{C}_c$ do not allow any rotation—the finger and the center of the circles are not included. In Figure 2.3, no rotation is possible around the points on the thin parts of the normal lines, while any rotation is allowed around those on the thick parts of the normal lines. This

also explains the case of a point finger touching a straight edge in the interior in Figure 2.1 (a). The points on the normal on the side where the object lies locally do not allow any rotation, while the points on the other part of the normal line allow clockwise and counterclockwise rotations. (See Figure 2.1.) Unfortunately, these insights cannot be used to obtain a graphical method for immobilization analysis that takes into account curvature.

2.1.2 Analysis in wrench space

When a force is applied to an object P at position p, it will make the object translate and/or rotate. The force is applied along an inward normal line of the boundary of P at p, and we call this inward normal line *line of force*. We let η be the normalized direction vector of a line of force. How this force moves P depends on the line of force and the magnitude of force applied along it. A line of force with the normalized direction vector η can be represented as a point in a three-dimensional space. We call this point a *wrench* of the finger pushing P. In other words, a wrench is a three-dimensional description of a directed line. The space of all wrench points is called *wrench space*. The wrench of a finger at p with a unit direction vector η is defined as $w = (\eta, p \times \eta) = (\eta, \tau) = (\eta_x, \eta_y, p \times \eta)$ for a two-dimensional object. For a three-dimensional object, the wrench of a finger at p with a unit direction vector $\eta = (\eta_x, \eta_y, \eta_z)$ is a six-dimensional vector $w = (\eta, p \times \eta) = (\eta, \tau) = (\eta_x, \eta_y, \eta_z, \tau_x, \tau_y, \tau_z)$. Note that a finger induces a wrench vector λw, where $\lambda > 0$, depending on how much force is applied through the finger.

We first introduce a characterization of form closure [38, 56, 58, 79] for planar objects. Most of our approaches in this thesis are based on this.

Theorem 2.1 *Given a set of κ (≥ 4) wrenches $w_1, w_2, \cdots, w_\kappa$ on a two-dimensional object P, the following three conditions are equivalent:*

(i) P is in form closure.

(ii) Any wrench w_F can be written as $-w_F = \lambda_1 w_1 + \cdots + \lambda_\kappa w_\kappa$ with $\lambda_i \geq 0$.

(iii) The origin O lies in the interior *of the convex hull of $w_1, w_2, \cdots, w_\kappa$.*

The equivalence of (i) and (ii) relies on the fact that the fingers together can be seen to apply any wrench that is a non-negative combination of the individual finger wrenches. Intuitively, P is in form closure if and only if any wrench applied to P can be cancelled by such a non-negative combination of finger wrenches. The equivalence of (ii) and (iii) can be verified easily: if we set w_F to be a zero vector, Theorem 2.1 (ii) becomes an algebraic formulation of Theorem 2.1 (iii). When $\kappa = 4$, all λ_i's must be positive to make a zero vector, i.e. $\sum_{i=1}^{4} \lambda_i w_i = \vec{0}$ for $\lambda_i > 0$. Theorem 2.1 (iii) and $\sum_{i=1}^{4} \lambda_i w_i = \vec{0}$ with $\lambda_i > 0$ are geometric and algebraic statements that the κ wrench vectors positively span the three-dimensional wrench space—this is why κ is at least four, the dimension plus one. The theorem implies that the magnitudes of wrench vectors are not important; only the direction matters. This again justifies that we take wrenches with unit force vector η. Note that if the origin lies on the *boundary* of the convex hull of w_1, w_2, w_3 and w_4,[2] the object is not in form-closure, but it may still be in second-order immobility—see Chapter 2 and 6.

Now we introduce a necessary and sufficient condition for a two or three dimensional object held with κ fingers to achieve form closure.

[2]Such grasps are called *equilibrium grasps*. This means that the object held with some fingers does not move, i.e. it is in equilibrium. Note that all immobilizing grasps are equilibrium grasps, but not all equilibrium grasps are immobilizing grasps. The equilibrium grasp where the origin of the wrench space is strictly inside the convex hull of the wrenches is called *nonmarginal equilibrium grasp* ([58]). Nguyen [58] showed that a nonmarginal equilibrium grasp is a form-closure grasp.

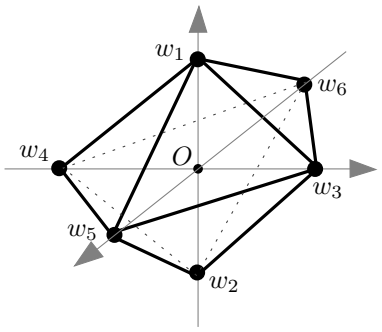


Figure 2.4: In a three-dimensional space, the convex hull of $w_1, \cdots, w_6$ contains the origin O in the interior. When one of the points is removed, the convex hull of the remaining points does not contain O in the interior.

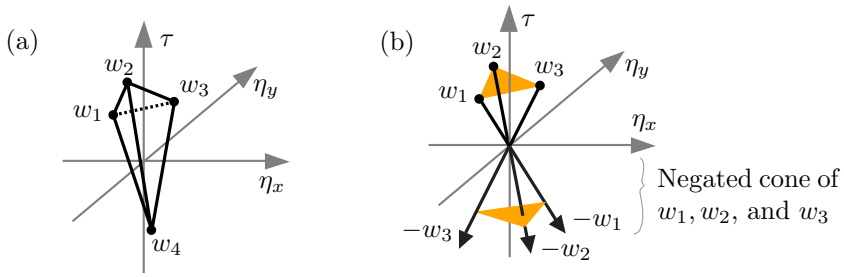


Figure 2.5: The negated cone of three wrench vectors w_1, w_2, and w_3 is the cone of $-w_1$, $-w_2$, and $-w_3$.

Theorem 2.2 *[38, 79] An object P held with κ fingers is in form closure, if and only if the κ wrenches $w_1, \cdots, w_\kappa$ positively span the wrench space.*

Since the wrench space for a three-dimensional object is six-dimensional, κ must be at least seven. In other words, seven fingers are necessary to put a three-dimensional object in form closure [46, 51, 56]. Recall that four fingers are necessary to put a planar object in form closure. However, we sometimes need six wrench points to positively span the wrench space [56, 51]; we cannot remove any of the six wrench points to keep the wrench space spanned positively. This happens when the six wrenches have three pairs of collinear wrench vectors, and two wrenches in each pair have opposite directions. One good example is the six points on the coordinate axes of a three-dimensional space. In Figure 2.4, two vectors are on the positive and the negative side of each axis. The same applies to six-dimensional wrench space; there are cases when one needs to have twelve wrenches to positively span the six-dimensional wrench space. In general, we may not be able to remove any vector from $2d$ vectors to positively span the d-dimensional space (Steinitz Theorem [56]).

To check if a given set of κ wrenches $w_1, \cdots, w_\kappa$ achieve form closure, one can use the negated cone formed by the others [14]. The convex hull of the κ wrenches contains the origin strictly inside, if and only if w_κ is in the negated cone of $w_1, \cdots, w_{\kappa-1}$, i.e. the cone of $-w_1, \cdots, -w_{\kappa-1}$. Figure 2.5 illustrates the negated cone formed by w_1, w_2 and w_3. This is an easy checking method, but unfortunately, it does not lead to an efficient algorithm.

2.2 Second-order immobility

Four frictionless point fingers are necessary to put two-dimensional objects in form closure, but often, three can immobilize planar objects. Applying Reuleaux's method implies that the three

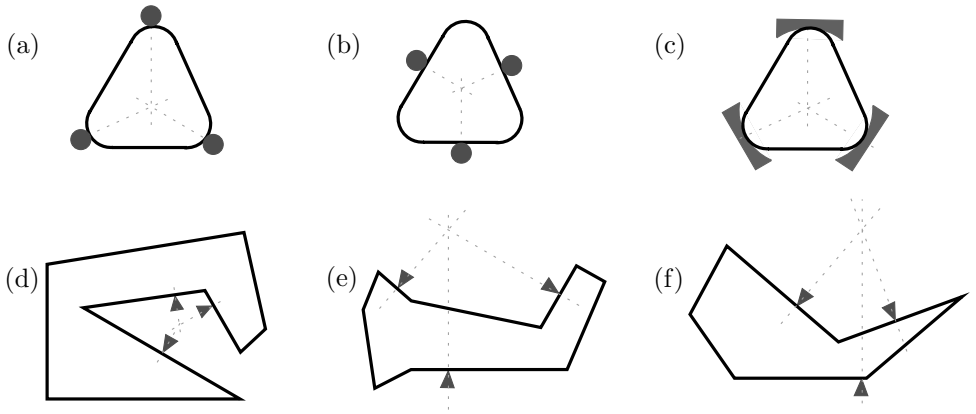


Figure 2.6: (a) A polygon that needs four point fingers to be immobilized. Many types of objects held with three point fingers on the boundary are shown in (b)–(f).

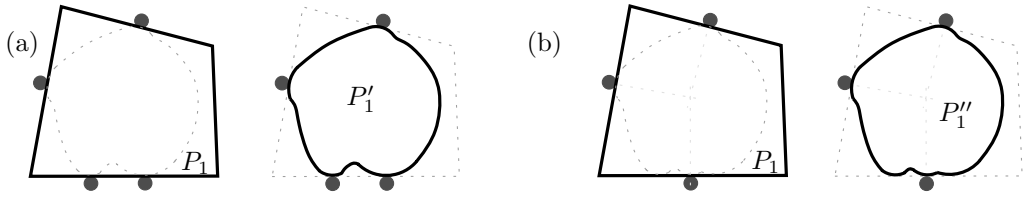


Figure 2.7: (a) Both of P_1 and P_1' are in form closure. (b) P_1 is in second-order immobility, but P_1'' is not.

normal lines must meet at one point. Obviously, this is not sufficient. In other words, Reuleaux's method cannot decide in which case the object is immobilized and in which case it is not. For example, the objects held with three fingers in Figure 2.6 (b), (c), and (f) are immobilized, and those in (a), (d) and (e) are not. Observe that the three normal lines meet at one point in all these cases. The objects in Figure 2.6 (b), (c), and (f) are in *second-order immobility*. To identify immobilized objects, we need to include curvatures in the analysis. Note that no information on curvature is used in form closure analyses; they include only the geometry of the normal lines.

We first look at the objects in Figure 2.6 (a), (b) and (c). It is easy to see that the objects in Figure 2.6 (b) and (c) are immobilized and that in (a) is not. We can understand this difference with the curvature of the object and the fingers. Now we look at the objects in Figure 2.6 (d), (e) and (f). It is not easy to see which one is immobilized and which one is not. The object in Figure 2.6 (f) is immobilized and those in (d) and (e) are not. Rimon and Burdick observed that the curvature of the motion of P should be considered to analyze second-order immobility [73, 74, 75, 76]. Note that the grasps in Figure 2.6 are all equilibrium grasps, and they are marginal equilibrium grasps. Form-closure grasps are non-marginal equilibrium grasps [56, 58].

To see the curvature effect more easily in comparison with form-closure grasps, let us take a polygon P_1, and put it in form closure with four fingers, and also put it in second-order immobility with three fingers. We replace P_1 with curved objects P_1' and P_1'' such that the normal lines remain the same. See Figure 2.7 (a) and (b). Then P_1' is still immobilized, while P_1'' is not immobilized. The objects P_1 and P_1' in Figure 2.7 (a) are in form closure, and P_1 in Figure 2.7 (b) is in second-order immobility.

In Section 2.2.1, we summarize the mobility theory developed by Rimon and Burdick [73, 74, 75, 76]. They explained in configuration space how the curvatures affect immobility. Czyzowicz, Stojmenovic and Urrutia [33] formulated a geometric condition on the object plane to analyze

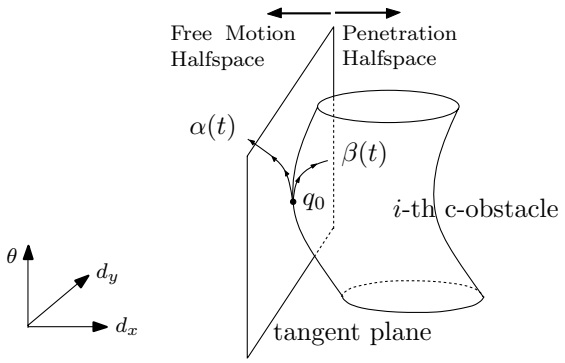


Figure 2.8: The first order approximation to the free motions of P at q_0.

second-order immobility for simple polygons. In Section 2.2.2, we will introduce this condition with a more visual and thorough explanation.

2.2.1 Analysis in configuration space

A placement of an object P can be described by a translation and rotation with respect to a reference placement. This is called *configuration*, and all possible configurations form *configuration space*, or c-space. The configuration space for a planar object is three-dimensional, and that for a three-dimensional object is six-dimensional. A finger prevents P from occupying certain configurations, and this set of configurations is called the *forbidden region*, and the region of configurations where P can be placed is called *free region*. The object cannot enter the forbidden region; it can freely move in the free region only. Let $\mathcal{CA}_i$ be the forbidden area of the configuration space because of the i-th finger $\mathcal{A}_i$ and, $\mathcal{S}_i$ is the boundary of $\mathcal{CA}_i$. A signed c-space distance function $d_i(q)$ measures the minimal Euclidean distance of a configuration point q from $\mathcal{S}_i$ as follows:

$$d_i(q) = \begin{cases} \text{distance}(q, \mathcal{S}_i) & \text{if } q \text{ is outside of } \mathcal{CA}_i \\ 0 & \text{if } q \text{ is on } \mathcal{S}_i \\ -\text{distance}(q, \mathcal{S}_i) & \text{if } q \text{ is in the interior } \mathcal{CA}_i \end{cases}$$

Let $\alpha(t)$ be a smooth c-space path such that at the starting configuration q_0, P touches a finger $\mathcal{A}_i$. This corresponds to a possible motion of P during which it maintains contact with $\mathcal{A}_i$. The set of first order free motions of P at q_0 is related to the first order Taylor expansion of d_i along $\alpha(t)$. When the motion is along $\mathcal{S}_i$, it is called first order roll-slide motions, and when the motion is strictly away from $\mathcal{S}_i$, it is called first order escape motions. Together, they are called first order free motions. This analysis with first order free motions is equivalent to form closure analysis, thus form closure is first-order immobility in their notion.

When the motion is along $\mathcal{S}_i$, and the first order term is zero, then the distance is decided by the second order term. With considering the second order term, when the motion is along $\mathcal{S}_i$, it is called a second order roll-slide motion, and when the motion is strictly away from $\mathcal{S}_i$, it is called a second order escape motion. Figure 2.8 illustrates the difference between the first and second order motions. The c-space curves $\alpha(t)$ and $\beta(t)$ in Figure 2.8 have the same tangent vector, and thus they are equivalent to the first order. But $\alpha(t)$ lies in the free space, while $\beta(t)$ does not. Note that all the free motions of P at an equilibrium grasp are necessarily roll-slide to first order. The objects held with three fingers in Figure 2.6 (d), (e) and (f) need analysis in the second order term. When the object turns out to be immobilized by checking the second order term for each of the fingers, we call this immobility *second-order immobility*. The first-order properties of the free paths and

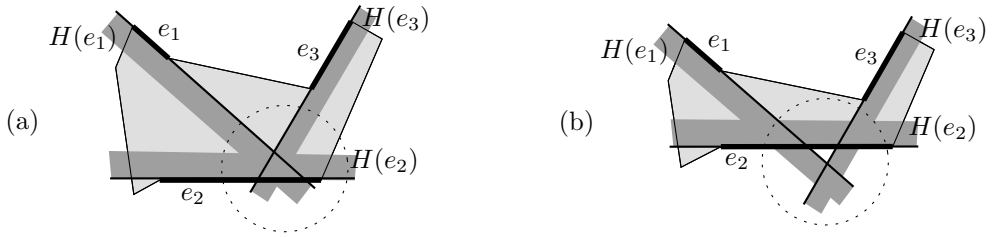


Figure 2.9: (a) The intersection of $H(e_1)$, $H(e_2)$ and $H(e_3)$ is a bounded triangle. (b) The intersection of $H(e_1)$, $H(e_2)$ and $H(e_3)$ is empty.

the c-obstacle boundaries (i.e. tangents and tangent hyperplanes) determine first order mobility of P.

2.2.2 Analysis in the object plane for simple polygons

Czyzowicz, Stojmenovic and Urrutia [33] provided a necessary and sufficient geometrical condition for a simple polygon to be in second-order immobility with three point frictionless fingers. Lemma 2.3 states the condition in [33]—we will refer to it as the *CSU condition*. Our algorithm in Chapter 6 to efficiently report all second-order immobility grasps on a simple polygon is based on the CSU condition. Unfortunately, to our best knowledge, there is no algorithm to efficiently report all second-order immobility grasps of an arbitrary planar object.

We first define the term *triangular triple*. Let the three fingers be on three edges e_1, e_2 and e_3. Let $H(e_1)$ be the open half-plane bounded by the supporting line of e_1, and it contains the interior of P around the contact position p_1 (see Figure 2.9). When the region $H(e_1) \cap H(e_2) \cap H(e_3)$ forms a (bounded) triangle, then e_1, e_2, and e_3 are said to be a *triangular triple*. Note that the object in Figure 2.6 (f) satisfies the triangular triple condition, and those in (d) and (e) do not.

Lemma 2.3 *[33] Three point fingers immobilize a polygon P, if and only if the following two hold:*

(i) The normals of the fingers meet at one point.

(ii) The contact edges form a triangular triple.

Now we show why the triangular triple condition is necessary in the second-order immobility condition. Let p_1, p_2 and p_3 be the three fingers at e_1, e_2 and e_3, such that the normal lines meet at one point. Instead of fixing the fingers, we fix P, and see how the triangle $\triangle p_1p_2p_3$ moves. Because p_1 and p_2 are on e_1 and e_2, we see how p_3 moves when p_1 and p_2 slide on e_1 and e_2 respectively.[3] Let ℓ_1, ℓ_2 and ℓ_3 be the supporting lines of e_1, e_2 and e_3, and let O be the intersecting point of ℓ_1 and ℓ_2. The three points O, p_1 and p_2 define a circle C, and let p denote the center of C.

Lemma 2.4 *If the normal lines at p_1, p_2 and p_3 meet at one point, and the triangle $\triangle p_1p_2p_3$ has two vertices p_1 and p_2 that slide along ℓ_1 and ℓ_2, then p_3 traces an ellipse.*

Proof: First we will show that p rotates around O. As p_1 and p_2 slide on ℓ_1 and ℓ_2, the circle C with O, p_1 and p_2 on the boundary also moves. The angle $\angle p_1Op_2$ has a fixed value α, therefore the interior angle $\angle p_1pp_2$ is also fixed—2α. Because $|\overline{p_1p_2}|$ is fixed, the radius of C $|\overline{Op}|$ is fixed, therefore, p rotates around O. Since $|\overline{Op}| = |\overline{pp_2}| = |\overline{pp_1}|$, the position of p is fixed with respect

[3]The lemmas on the sliding triangle are from http://whistleralley.com/ellipse/ellipse.htm.

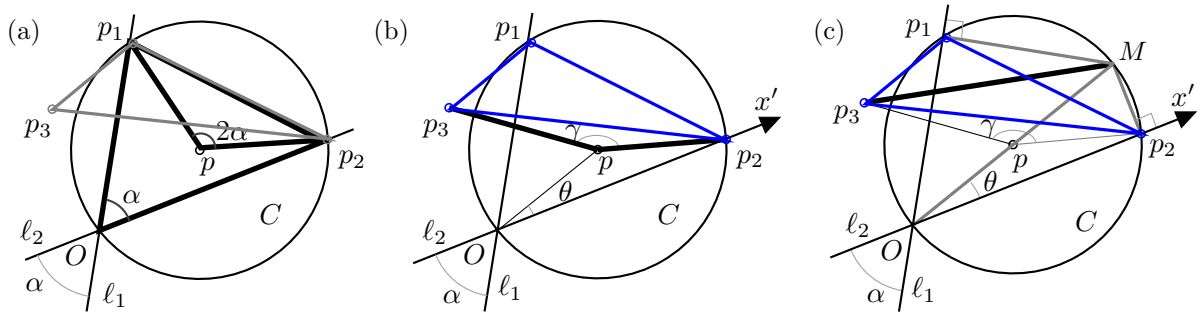


Figure 2.10: (a) The length of $|\overline{pp_3}|$ and angle $\angle p_2pp_3$ are constant. (b) Three points O, M and p_1 are on C, and $\angle Op_1M = \pi/2$.

to segment $\overline{p_1p_2}$, hence, with respect to the triangle $\triangle p_1p_2p_3$. Therefore, $\overline{pp_3}$ and angle $\angle p_2pp_3$ are constant. See Figure 2.10 (a).

Let θ be the angle $\angle pOp_2$, and γ be $\angle p_2pp_3$. When triangle $\triangle p_1p_2p_3$ slides along ℓ_1 and ℓ_2, θ changes, but not γ. See Figure 2.10 (b). When we consider ℓ_2 as x axis, the x coordinate for p_3 is $|\overline{Op}|\cos\theta + |\overline{pp_3}|\cos(\gamma-\theta)$, and the y coordinate is $|\overline{Op}|\sin\theta + |\overline{pp_3}|\sin(\gamma-\theta)$. Now we change our coordinate system; we rotate ℓ_2 around O by $\gamma/2$, and take this as new x axis. Then the coordinate of p_3 becomes $(|\overline{Op}|\cos(-\gamma/2+\theta) + |\overline{pp_3}|\cos(\gamma/2-\theta), |\overline{Op}|\sin(-\gamma/2+\theta) + |\overline{pp_3}|\sin(\gamma/2-\theta)) = ((|\overline{Op}| + |\overline{pp_3}|)\cos(\theta-\gamma/2), (|\overline{Op}| - |\overline{pp_3}|)\sin(\theta-\gamma/2))$, which is a description of an ellipse. □

The next lemma shows that the edge e_3 keeps the ellipse that p_3 traces on one side.

Lemma 2.5 *If the normals of p_1, p_2 and p_3 meet at one point, e_3 is tangent to the ellipse that p_3 of the sliding triangle $\triangle p_1p_2p_3$ traces.*

Proof: Let M be the intersection of the two edge normal lines for e_1 and e_2 at p_1 and p_2. We first show that $\overline{OM}$ contains p, i.e. it is a diameter of C. The three points O, M and p_1 are on C, and the angle $\angle Op_1M$ is a right angle from the construction of M. See Figure 2.10 (b). The center of the circumcircle of a triangle lies on one of the triangle's sides, if and only if the triangle is a right triangle, therefore, $p \in \overline{OM}$. The coordinate of M with ℓ_2 as x-axis is $(2|\overline{Op}|\cos\theta, 2|\overline{Op}|\sin\theta)$. The edge normal line of e_3 at p_3 is the supporting line of $\overline{p_3M}$ from the construction—p_3 is chosen such that the three normal lines of e_1, e_2 and e_3 at p_1, p_2 and p_3 meet at one point.

Remember that $\overline{p_3M}$ is an edge normal of e_3. Thus if the tangent line of the ellipse at p_3 is perpendicular to $\overline{p_3M}$, e_3 is tangent to the ellipse at p_3. The tangent direction vector $\vec{t}$ of the ellipse at p_3 is $(-|\overline{Op}|\sin\theta + |\overline{pp_3}|\sin(\gamma-\theta), |\overline{Op}|\cos\theta - |\overline{pp_3}|\cos(\gamma-\theta))$, and $\overrightarrow{p_3M} = (2|\overline{Op}|\cos\theta - |\overline{Op}|\cos\theta - |\overline{pp_3}|\cos(\gamma-\theta), 2|\overline{Op}|\sin\theta - |\overline{Op}|\sin\theta - |\overline{pp_3}|\sin(\gamma-\theta)) = (|\overline{Op}|\cos\theta - |\overline{pp_3}|\cos(\gamma-\theta), |\overline{Op}|\sin\theta - |\overline{pp_3}|\sin(\gamma-\theta))$. Showing that $\overrightarrow{p_3M}\cdot\vec{t} = 0$ will finish the proof.

$$
\begin{aligned}
\overrightarrow{p_3M}\cdot\vec{t} &= (|\overline{Op}|\cos\theta - |\overline{pp_3}|\cos(\gamma-\theta))\,(-|\overline{Op}|\sin\theta + |\overline{pp_3}|\sin(\gamma-\theta)) \\
&\quad +(|\overline{Op}|\sin\theta - |\overline{pp_3}|\sin(\gamma-\theta))\,(|\overline{Op}|\cos\theta - |\overline{pp_3}|\cos(\gamma-\theta)) \\
&= -|\overline{Op}|^2\sin\theta\cos\theta + |\overline{Op}||\overline{pp_3}|\cos\theta\sin(\gamma-\theta) \\
&\quad +|\overline{Op}||\overline{pp_3}|\cos(\gamma-\theta)\sin\theta - |\overline{pp_3}|^2\sin(\gamma-\theta)\cos(\gamma-\theta) \\
&\quad +|\overline{Op}|^2\sin\theta\cos\theta - |\overline{Op}||\overline{pp_3}|\cos(\gamma-\theta)\sin\theta \\
&\quad -|\overline{Op}||\overline{pp_3}|\cos\theta\sin(\gamma-\theta) + |\overline{pp_3}|^2\sin(\gamma-\theta)\cos(\gamma-\theta) \\
&= 0.
\end{aligned}
$$

□

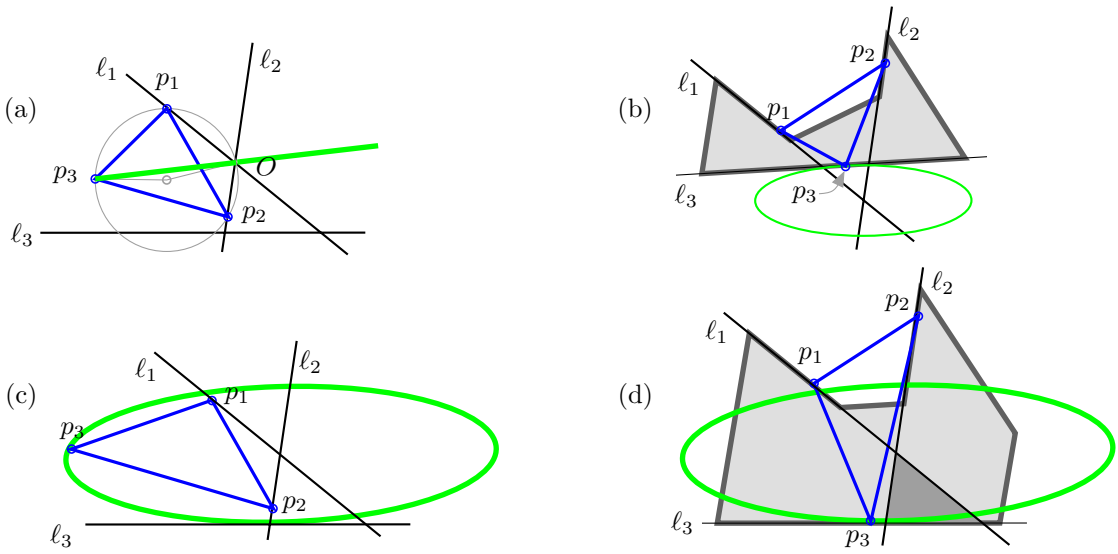


Figure 2.11: The curvature of motions of the fingers on three edges that make a triangular triple, and those that does not make a triangular triple.

When e_1, e_2 and e_3 meet at one point, p_3 follows a degenerate ellipse—a line segment (Figure 2.11 (a)). When e_1, e_2 and e_3 do not make a triangular triple, the ellipse is outside of $H(e_3)$. See Figure 2.11 (b). When the edges make a triangular triple, the ellipse is in $H(e_3)$, which means that the triangle cannot move any more without penetrating the interior of P, therefore, P is immobilized. See Figure 2.11 (d).

2.3 Force closure

When frictional fingers immobilize an object P, P is said to be in *force closure*. The word "force" is used to describe this immobility with frictional fingers, because friction assumes force. Fewer than four (frictional) fingers can achieve force closure, because a frictional finger applies force along a set of lines, while a frictionless finger does along only one line. We call the set of lines of force *friction cone*. In this thesis, we focus on hard point fingers, and use Coulomb friction model for analysis.[4] When a hard frictional point finger pushes an object P, the surface normal has a friction cone around it with half angle ϑ, which is the region where the lines of force lie. Figure 2.12 describes friction cones for two-dimensional and three-dimensional objects. The friction cone on a three-dimensional object P can be constructed as follows: take the normal line at the contact, rotate it around the contact by ϑ, and rotate this new line around the normal. The cone constructed outside P is the friction cone. Any line of force—it is in the friction cone—can be represented as a positive combination of the boundary lines.

Assuming Coulomb friction, three and four frictional hard point fingers are necessary to immobilize two and three dimensional objects respectively [51]. When we use rounded fingers with static friction, two can grasp any two-dimensional object, and three can grasp any two-dimensional object [54]. As in the case of form closure, a non-marginal equilibrium grasp achieves force closure for a two-dimensional object [60] and also for a three-dimensional object [61].

To analyze force closure, we can basically use the analysis for form closure: Reuleaux's method and the analysis in wrench space. For a two-dimensional object, two or three frictional fingers can

[4]There are many kinds of fingers. The friction cones for different fingers are described in [58].

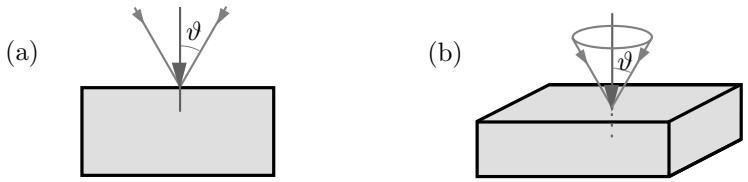


Figure 2.12: Coulomb friction model of a frictional point finger. (a) A friction cone of a finger on a planar object. (b) A friction cone of a finger on a three-dimensional object. Any boundary line forms angle ϑ with the normal line.

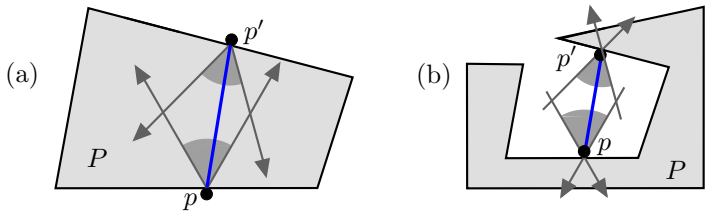


Figure 2.13: Force closure grasps with two frictional fingers. (a) Line $\overline{p_1p_2}$ is in the internal friction cones, and (b) $\overline{p_1p_2}$ is in the external friction cones.

achieve force closure. A necessary and sufficient condition in wrench space for the fingers to achieve force closure is similar to that for form closure; the wrenches of the boundary lines of the friction cones must positively span the wrench space to achieve force closure [79]. A necessary and sufficient condition on the object plane for two fingers to achieve force closure is as follows: a planar object with two frictional fingers achieve force closure, if and only if the joining line of the two contacts must be in the internal friction cones or in the external friction cones [58, 60]—see Figure 2.13. An equilibrium grasp with three fingers achieves force closure if and only if there exist three lines in the friction cones which positively span the plane and which intersect at one point [60]. Force closure grasps for three-dimensional objects with four Coulomb-frictional point-contact hard fingers have been studied by Ponce et al. [61]. They also provided a geometric characterization of force closure grasps with four fingers.

2.4 Preliminaries

In this section, we introduce the theorem of form closure, and data structures and algorithms to report all intersections. In Section 2.4.1, we state the form closure condition and the transformation of the problem into intersection problem. In Section 2.4.2, we introduce all the intersection algorithms and data structures.

2.4.1 Projections of wrenches

In the following, we define the segment $\overline{pq}$ to be the *relatively open* segment connecting p and q, that is, the set $\overline{pq} := \{\lambda p + (1-\lambda)q \mid 0 < \lambda < 1\}$. We have a simple geometric lemma.

Lemma 2.6 *Let w_1, w_2, w_3, w_4 be four points in $\mathbb{R}^3$. The origin O lies in the interior of the convex hull of $w_1, \ldots, w_4$ if and only if there are points $p_1 \in \overline{w_1w_2}$ and $p_2 \in \overline{w_3w_4}$ such that $O \in \overline{p_1p_2}$.*

Proof: The "if" direction is straightforward, so we show "only if" direction only. Suppose that the origin O lies strictly inside the tetrahedron formed by w_1, w_2, w_3 and w_4. Consider the plane Π containing w_1, w_2, and O. It intersects the segment $\overline{w_3w_4}$ in a point p_2. See Figure 2.14. The

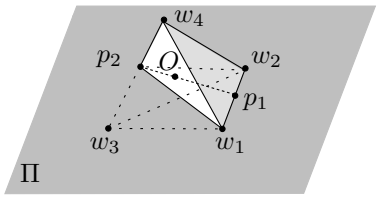


Figure 2.14: The origin O lies in the interior of the convex hull of w_1, w_2, w_3 and w_4, if and only if there are points $p_1 \in \overline{w_1 w_2}$ and $p_2 \in \overline{w_3 w_4}$, such that $O \in \overline{p_1 p_2}$.

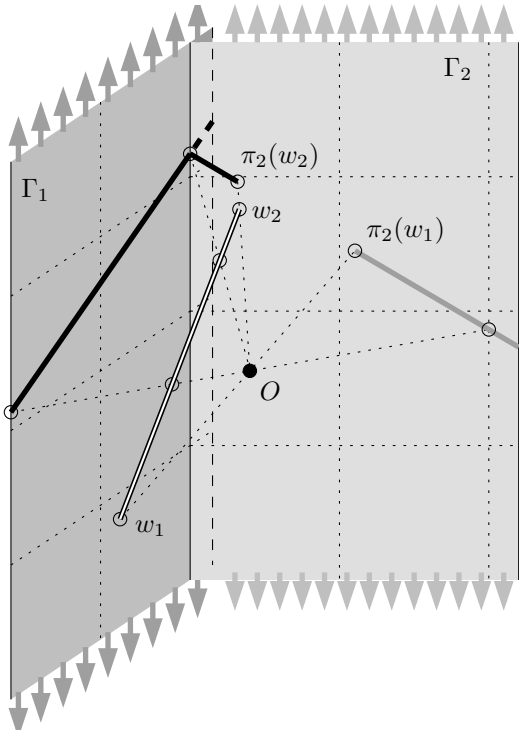


Figure 2.15: Screen Γ and the projection of a line segment.

intersection of the tetrahedron with Π is the triangle $\triangle w_1 w_2 p_2$. The point O lies in the interior of this triangle, and so the line Op_2 intersects $\overline{w_1 w_2}$ at an interior point p_1 of $\overline{w_1 w_2}$. □

If we project these four points on some planes, they have an interesting property. In wrench space, the horizontal dimensions η_x and η_y represent the direction of the force applied by a wrench, while the vertical dimension τ represents the torque that is caused by the force. We take two planes Γ_1 and Γ_2 for our screen Γ throughout the thesis, unless stated otherwise. They are defined as follows: $\Gamma_1 := \{(\eta_x, 1, \tau)^T \mid -1 < \eta_x < 1 + \varepsilon, \tau \in \mathbb{R}\}$ and $\Gamma_2 := \{(-1, \eta_y, \tau)^T \mid -1 < \eta_y < 1 + \varepsilon, \tau \in \mathbb{R}\}$, where ε is an arbitrarily small positive constant. The screens are extended by ε, so that an interior point of a wrench set is projected as an interior point on at least one plane of Γ, according to the following projection scheme. See Figure 2.15 and 2.16.

Now we will project wrenches w that do not lie on the τ-axis onto Γ as follows. The projection $\pi_i(w)$ of w on Γ_i ($i = 1, 2$) is the intersection, if it exists, of Γ_i with the line through w and the origin O. If w lies between O and $\pi_i(w)$, we color $\pi_i(w)$ ***blue***. If O lies between w and $\pi_i(w)$, we color $\pi_i(w)$ ***red***. It is easy to see that for each wrench w, at least one of $\pi_1(w)$ and $\pi_2(w)$ exists.

A segment $\overline{w_1 w_2}$ is projected onto Γ by projecting each point $w \in \overline{w_1 w_2}$. The projection $\pi(\overline{w_1 w_2})$ consists of at most four segments on Γ, where each segment is either blue or red. See

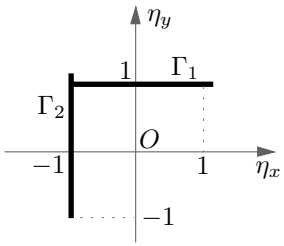


Figure 2.16: Screen Γ in wrench space, viewed from the positive τ axis.

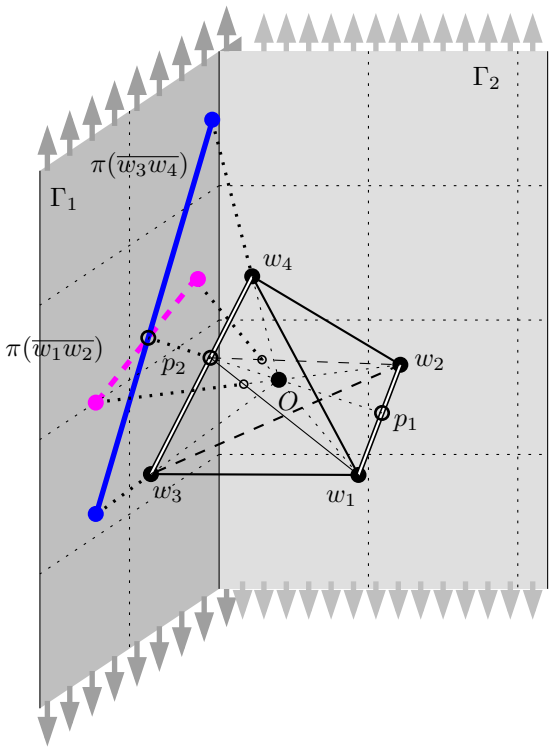


Figure 2.17: The origin is inside the convex hull of w_1, w_2, w_3 and w_4, if and only if red and blue projections of $\overline{w_1w_2}$ and $\overline{w_3w_4}$ intersect each other in the interior.

Figure 2.15. The following lemma is a reformulation of Theorem 2.1 (iii) in terms of projections. Figure 2.17 illustrates Lemma 2.7.

Lemma 2.7 *Given an object P with four contact wrenches w_1, w_2, w_3 and w_4, P is in form-closure if and only if a red part of $\pi(\overline{w_1w_2})$ intersects a blue part of $\pi(\overline{w_3w_4})$, or vice versa.*

Proof: By Theorem 2.1 and Lemma 2.6, the object is in form closure if and only if there exist $p_1 \in \overline{w_1w_2}$ and $p_2 \in \overline{w_3w_4}$ such that $O \in \overline{p_1p_2}$. Since $\pi(\overline{w_1w_2})$ and $\pi(\overline{w_3w_4})$ are line segments, neither $\overline{w_1w_2}$ nor $\overline{w_3w_4}$ passes through the origin. Furthermore, on a screen Γ_i where $\pi_i(p_1)$ exists (which must be true for at least one of the screens Γ_1 and Γ_2), we must have $\pi_i(p_1) = \pi_i(p_2)$ (since they lie on the same line through the origin) and the colors of these projections differ (since they lie on different sides of the origin)—see Figure 2.17. □

2.4.2 Algorithms and data structures for intersection search problem

In this thesis, we need to search for red and blue intersections between points, line segments, triangles, arcs and semi-algebraic sets. We will divide this section into two parts: the intersection search strategies between points, line segments and triangles, and those between points, arcs and semi-algebraic sets.

Intersections between points, line segments and triangles

We wish to search for red and blue intersections between the following pairs: (i) line segments, (ii) points and triangles. To find all intersecting red and blue line segments, we use a segment intersection algorithm and a segment intersection search structure. To report all red and blue intersections between points and triangles, we use a triangle search structure.

Segment intersection algorithm reports all K intersections between red and blue line segments among q red and blue line segments—there can be intersections between reds and also between blues. The algorithm by Agarwal [1], improved by Chazelle [18] does this in time $O(q^{4/3}\log^{1/3} q + K)$, (Chazelle's description mentions colorblind intersections only, but his approach also works for red-blue intersections). Throughout this thesis, we let k denote the output size for one query, and K be the overall output size, unless stated otherwise.

The *segment intersection search structure* supports a query of the following form: given a set of line segments, report all segments intersecting a query segment in the interior. The *triangle search structure* supports a query of the following form: given a set of points, report all points lying in a query triangle. For both structures we use one type of data structure called hierarchical cutting tree proposed by Matoušek [52]. Matoušek explains how we can build, for any set P of m points in the plane, and a prescribed parameter t such that $\log m \leq t \leq m$, a tree of height $O(\log m)$ with the following properties:

- the number of nodes at depth i is $O(\rho^{2i})$, for some constant ρ; each node v at depth i has an associated subset P_v of P of size $O(m/\rho^i)$;
- there are $O((m/t)^2)$ leaves v, and their sets P_v have size $O(t)$;
- for any half plane H, the points in $P \cap H$ are exactly the points in the sets associated with a set of non-leaf nodes (one node at each depth in the tree), plus some or all of the points in a single leaf. The set of non-leaf nodes and the leaf can be identified in $O(\log m)$ time.

The tree can be built in $O(m^2/t)$ time.[5] If we just store the sets P_v explicitly, this tree can obviously be used to answer half plane range reporting queries in $O(\log m + t + k) = O(t + k)$ time: find the leaf, check its complete contents, and find the non-leaf nodes, and just report their complete contents.

To extend this approach to a triangle search or segment intersection structure, we proceed as follows. We generalize the above tree a little bit. Instead of points p, we store tuples of points $(p_1, ..., p_\kappa)$. The half plane property of the tree will now read as: "For any half plane H, the tuples $\{(p_1, \cdots, p_\kappa) \mid p_1 \in H\}$ are exactly the tuples...". We call such a tree an order-1 tree. A tree of order j, for $j > 1$, will be just like an order-1 tree, with two exceptions. First, and most important: each set P_v for a node v of the order-j tree will be stored as a tree of order $(j-1)$ on the tuples in P_v. Second, the half plane property now reads as: "For any half plane H, the tuples $\{(p_1, ..., p_\kappa) \mid p_j \in H\}$ are exactly the tuples...".

[5] Theorem 5.1 from Matoušek [52], with $r = m/t$ and $d = 2$. Note that there is a typographical error in Matoušek's publication: it says $O(\dot{\rho})$ instead of $O(\rho^{di})$.

Lemma 2.8 *A tree of order j:*

- *can be built in time $O(m^2(\log^{j-1} m)/t)$, and*
- *can be used to report, for any set of j half planes $(H_1, ..., H_j)$, all tuples $\{(p_1, ..., p_\kappa) \mid \forall_{1\leq i\leq j} \quad p_i \in H_i\}$, in time $O(t \log^{j-1} m + k)$, where k is the number of tuples reported.*

Proof: We prove the lemma by induction on j.
For $j = 1$, it is obviously true.
The construction of a tree of order $j > 1$, consists of the construction of the main structure, in $O(m^2/t)$ time, and the construction of the associated trees of order $(j-1)$. By the induction hypothesis, the construction of an order-$(j-1)$ tree at depth i in the order-j tree takes $O((m/\rho^i)^2(\log^{j-2} m)/t)$ time. The construction times thus add up to:

$$O\left(\frac{m^2}{t}\right) + \sum_{i=0}^{O(\log m)} O(\rho^{2i}) O\left(\left(\frac{m}{\rho^i}\right)^2 \frac{\log^{j-2} m}{t}\right) = O\left(\frac{m^2 \log^{j-1} m}{t}\right)$$

The search in an order-j tree with H_j yields $O(\log m)$ nodes whose order-$(j-1)$ trees have to be searched. By the induction hypothesis, searching the order-$(j-1)$ trees costs $O(t \log^{j-2} m + k)$ time for each tree, which adds up to $O(t \log^{j-1} m + k)$. Furthermore, one leaf of size t has to be searched, for a cost of $O(t)$, so that the total time spent searching is $O(t \log^{j-1} m + k)$. □

Corollary 2.1 *In $O(m^2 \log m)$ time, we can build a triangle search structure on a set S of m points that answers queries in $O(\log^3 m + k)$ time, where k is the number of points in S that lie inside the query triangle.*

Proof: We build a tree of order 3 with $t = \log m$ and store each point $p \in S$ in it as a tuple (p, p, p). To answer a triangle query, we search the order-3 tree with the three half planes whose intersection is the query triangle. By Lemma 2.8, the tree can be built in $O(m^2 \log m)$ time and answers queries in $O(\log^3 m + k)$ time. [6] □

Corollary 2.2 *In $O(m^2)$ time, we can build a triangle search structure on a set S of m points that answers queries in $O(\log^4 m + k)$ time, where k is the number of points in S that lie inside the query triangle.*

Proof: We follow the same approach as in Corollary 2.1, but now with $t = \log^2 m$. □

Corollary 2.3 *In $O(m^2 \log^2 m)$ time, we can build a segment intersection structure on a set of m line segments that answers queries in $O(\log^4 m + k)$ time, where k is the number of line segments in S that intersect the query segment.*

Proof: We use the same transformation as, for example, in [3]. Assume that there are no vertical segments (if there are vertical segments, we must turn everything just a little bit to prevent degeneracies). We build an order-4 tree with $t = \log m$, storing each line segment $s = \overline{s_0 s_1}$ as a tuple $(l^*(s), l^*(s), s_0, s_1)$, where $l^*(s) = (a, b)$ is the dual of the supporting line $l(s) : y = ax + b$ of s. Observe that a query segment $q = \overline{q_0 q_1}$ intersects s if and only if the following two conditions are met:

[6] With Theorem 6.1 in Matoušek's publication [52], he improves the construction time for the triangle search structure to $O(m^2 \log^\varepsilon m)$ (with the same query time) for any constant $\varepsilon > 0$; the same technique could be used to improve the construction time for the segment intersection structure to $O(m^2 \log^{1+\varepsilon} m)$. However, these improvements don't affect our final bounds, so we will ignore them for simplicity.

- s_0 lies above $l(q)$ while s_1 lies below $l(q)$ (or the other way around), and
- q_0 lies above $l(s)$ while q_1 lies below $l(s)$, or equivalently: $l^*(s)$ lies below the dual line q_0^* of q_0 and above the dual line q_1^* of q_1 (or the other way around).

An intersection query with a line segment can thus be formulated as a query with four half planes, bounded by q_0^*, q_1^*, and $l(q)$ (twice) in the order-4 tree storing tuples $(l^*(s), l^*(s), s_0, s_1)$. □

Sometimes, we use a variation of Matoušek's hierarchical cutting trees with space and query time trade-off for triangle search structure. It stores the points in M space in $O(q^{1+\varepsilon} + M \log^{\varepsilon} q)$ time, and reports k points in a query triangle in $O(q/\sqrt{M} \log^3 \frac{M}{q} + k)$. If we set M to be $q^{4/3}$, the preprocessing time becomes $O(q^{4/3} \log^{\varepsilon} q)$, and the query time is $O(q^{1/3} \log^3 q + k)$. See Theorem 6.2 in [52].

Intersections between points, arcs and semi-algebraic sets

In this thesis, we need to search for red and blue intersections between algebraic arcs and line segments, and between points and semi-algebraic sets. To find all red and blue intersections between arcs and segments, we use a red-blue segment-arc intersection algorithm and a segment-arc query structure. To report all red and blue intersections between points and semi-algebraic sets, we use a semi-algebraic range search structure.

The *red-blue segment-arc intersection algorithm* by Koltun [45] works on q possibly intersecting red (blue) segments and q possibly intersecting blue (red) arcs. It reports all K red and blue intersecting arc and segment pairs in $O(q^{3/2+\varepsilon} + K)$-time, using $O(q)$ space. Alternatively, we can store q arcs in a *segment-arc query structure* by Koltun [45] in $O(q^{2+\varepsilon})$ time, and report all k arcs intersecting a query segment in $O(\log q + k)$ time.

The *semi-algebraic range search structure* by Agarwal and Matoušek [2] is to report all points in a query semi-algebraic set. It stores q points in $O(q \log q)$-time, and reports all k points in a query semi-algebraic set in $O(q^{1/2+\varepsilon} + k)$-time.

Chapter 3

Computing All Form-Closure Grasps of a Simple Polygon with Few Fingers

Many researchers studied the problem of reporting all form-closure grasps of polygons in the non-modular setting [39, 82, 91]. Four wrenches (normal lines) are necessary to achieve form closure for a two-dimensional object. Van der Stappen et al. [82] proposed an efficient output-sensitive $O(n^{2+\varepsilon} + K)$-time algorithm to compute all K edge quadruples of a polygon with n edges, which allow form-closure grasps with four frictionless point fingers. Fewer than four point fingers may suffice for form closure if the object has concave vertices. Computing all form-closure grasps involving concave vertices was first studied by Gopalakrishnan and Goldberg [39]. They checked all concave vertex pairs to find all K concave vertex pairs that allow a two-finger form-closure grasp.

In this chapter,[1] we propose efficient output-sensitive algorithms to enumerate all combinations of concave vertices and edges of a polygon that allow form-closure grasps with two or three frictionless point fingers. More specifically they are: (i) pairs of concave vertices, (ii) triples of one concave vertex and two edges, (iii) triples of two concave vertices and one edge, and (iv) triples of concave vertices. Here, we improve the result in [39]. The proposed algorithms are based on the analysis of form closure in wrench space. This turns out to work well for synthesis of all grasps, while it is not obvious how to compute all grasps with most intuitive analysis in two-dimensional plane of the planar object itself, as Reuleaux's method [70].

When polygons are rectilinear, all form-closure grasps can be enumerated faster, because the wrenches have a regular pattern. We propose efficient output-sensitive algorithms to enumerate all combinations of concave vertices and edges of a rectilinear polygon that allow form-closure grasps with two, three or four frictionless point fingers. The combinations include: (i) edge quadruples, (ii) triples of one concave vertex and two edges, (iii) pairs of concave vertices, and (iv) triples of two concave vertices and one edge.

This chapter is structured as follows. In Section 3.1, we introduce notations, form-closure conditions, projection schemes, wrench shapes and data structures for intersection search problems. In Section 3.2, we propose output-sensitive algorithms to report all combinations of edges and concave vertices of a polygon that allow form-closure grasps with less than four frictionless point fingers. Section 3.3 covers rectilinear polygons; we propose output-sensitive algorithms to report all combinations of edges and concave vertices of a rectilinear polygon that allow form-closure grasps with at most four frictionless point fingers.

[1] This chapter is based on "On computing all immobilizing grasps of a simple polygon with few contacts" [23] by J.-S. Cheong, Herman Haverkort and Frank van der Stappen in ISAAC (2003), and "On computing all immobilizing grasps of a simple polygon with few contacts" [24] by J.-S. Cheong, Herman Haverkort and Frank van der Stappen in Algorithmica (2006).

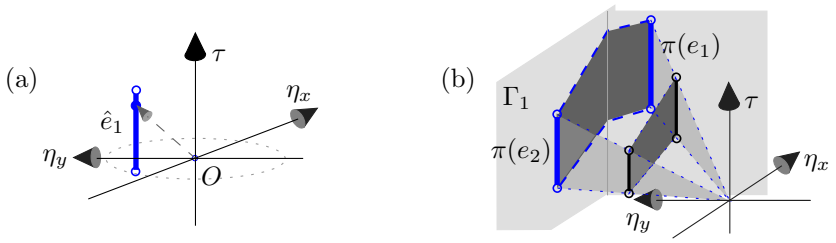


Figure 3.1: (a) The edge wrench induced by a finger on an edge. (b) The projection of a quadrilateral $r(e_1, e_2)$.

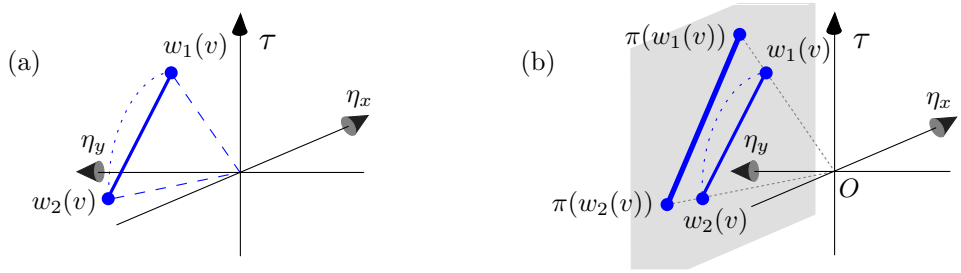


Figure 3.2: The wrench induced by a finger at a concave vertex and its projection. The projection is composed of at most four line segments in general.

3.1 Preliminaries

As a consequence of Theorem 2.1, our problem is to find all edge wrench sets that contain four points whose convex hull contains the origin of wrench space (η_x, η_y, τ). As seen in Section 2.4, this problem is transformed into red and blue intersection problems on the projected wrenches on screen Γ. More dtailed information on projections and screen Γ can be found in Section 2.4. First we will see the shape of a wrench set induced by a finger along an edge in Section 3.1.1. In Section 3.1.2, we discuss the data structures and algorithms that we use in this chapter.

3.1.1 The shapes of wrench sets

We place a finger at position p on an edge e. The corresponding wrench is $(\eta, \tau)^T = (\eta_x, \eta_y, p \times \eta)^T$, where $\eta = (\eta_x, \eta_y)^T$ is the inward normal line of e. As mentioned earlier in Section 2.4, η_x and η_y are the horizontal dimensions, and τ is the vertical dimension. When a finger slides along e, the inward normal direction η does not change; only the torque τ changes. Thus the set of wrench points forms a vertical line segment. See Figure 3.1 (a). We call this vertical segment the *edge wrench set of* e, and denote it with $\hat{e}$. An edge wrench set is a relatively open line segment, i.e. the endpoints are excluded, since we place a finger in the interior of an edge. Note that wrenches never lie on τ-axis, as the inward normal direction is never $(0, 0)^T$. The projection of a vertical line segment $\hat{e}$ is also a vertical line segment on Γ, which is denoted by $\pi(\hat{e})$.

When two fingers slide along two edges e_1 and e_2, the corresponding wrench points w_1 and w_2 also slide along $\hat{e}_1$ and $\hat{e}_2$. The union of the line segments connecting w_1 and w_2 for all $w_1 \in \hat{e}_1$ and $w_2 \in \hat{e}_2$ forms a trapezoid. Let $r(e_1, e_2)$ denote the projection of this trapezoid. We formally define $r(e_1, e_2)$ as follows: $r(e_1, e_2) := \bigcup\{\pi(\overline{w_1 w_2}) \mid w_1 \in \hat{e}_1, w_2 \in \hat{e}_2\}$. Observe that $r(e_1, e_2)$ is also a trapezoid, and it is composed of at most four trapezoids on Γ. Figure 3.1 (b) shows a trapezoid $r(e_1, e_2)$, which is composed of two trapezoids. Note that $r(e_1, e_2)$ is partially open; only the vertical boundary segments except the endpoints are included.

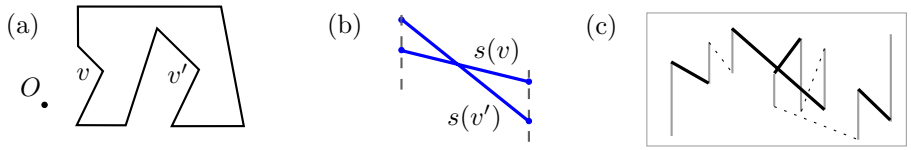


Figure 3.3: A polygon with two concave vertices whose vertex wrenches intersect each other. In (c), the concave vertex wrenches are in black solid lines; the dotted line segments correspond to convex vertex wrenches and gray solid vertical line segments correspond to edge wrenches.

A finger at a concave vertex v touches two incident edges e_1 and e_2 of v at position p. Let η_i be the normal line of edge e_i at p, and $w_i(v)$ be its wrench ($i = 1, 2$). A finger at v induces a set of lines of force between η_1 and η_2, thus a set of wrench points between $w_1(v)$ and $w_2(v)$. More precisely, they are $(\alpha_1\eta_1 + \alpha_2\eta_2, p \times (\alpha_1\eta_1 + \alpha_2\eta_2))$,[2] which becomes $\alpha_1 w_1(v) + \alpha_2 w_2(v)$, where $0 \leq \alpha_1 \leq 1$, $0 \leq \alpha_2 \leq 1$, and $\alpha_1 + \alpha_2 = 1$. Note that this line segment includes $w_1(v)$ and $w_2(v)$, because the finger *is* at an endpoint of each of e_1 and e_2. We call this line segment the *vertex wrench set (of v)*, and denote it by $\hat{v}$. The projection of the segment connecting $w_1(v)$ and $w_2(v)$ is the (closed) line segment connecting $\pi(w_1(v))$ and $\pi(w_2(v))$ on Γ. We let $s(v)$ denote this segment on Γ. See Figure 3.2. Note that vertex wrench sets can intersect each other—see Figure 3.3.

3.1.2 Intersection search algorithms

In Section 3.2, we need to perform two kinds of queries to report all red and blue intersections. For these queries, we use the following three: a segment intersection algorithm, a segment intersection search structure, and a triangle search structure. More information on these structures and algorithms can be found in Section 2.4.2.

In Section 3.3, we use an interval tree, a binary search tree and a two-level orthogonal search tree. All these trees can store q intervals or points in $O(q \log q)$ time. The query time for an interval tree and a binary search tree is $O(\log q + k)$, and the query time for a two-level orthogonal search tree is $O(\log^2 q + k)$, where k is the output size. More information on these trees can be found in [9]). Throughout this chapter, we let k denote the output size for one query, and K denote the overall output size.

3.2 Computing all form-closure grasps with at most three fingers

Throughout this chapter, we let n be the number of edges and m be the number of concave vertices of a polygon P. Before we introduce the algorithms to report all combinations of edges and concave vertices, we show how to compute the exact positions on a given set of edges and concave vertices that achieve form-closure. In particular, we take a combination of two edges e_1 and e_2, and a concave vertex v. Note that a concave vertex has a fixed position where a finger can be placed. Hence we focus on computing the positions of given edges that yield form-closure grasps with a given concave vertex. A given set (e_1, e_2, v) has form-closure grasps if and only if a blue/red trapezoid of $\hat{e}_1\hat{e}_2$ intersects a red/blue line segment of $s(v)$ in the interior. (Readers can find more detailed information in Section 3.2.2.) We take a point p in the intersection of red and blue $\hat{e}_1\hat{e}_2$ and $s(v)$. We have a range of points on $\hat{e}_1$ and $\hat{e}_2$, whose line segment contains p in the interior.

[2] If we let $\alpha_1\eta_1 + \alpha_2\eta_2$ be a unit vector, the set of the corresponding wrench points forms an arc between $w_1(v)$ and $w_2(v)$. Observe that this arc lies on the plane defined by three points O, $w_1(v)$ and $w_2(v)$. Hence the projection of this arc is a line segment connecting $\pi(w_1(v))$ and $\pi(w_2(v))$. For simplicity, we take the line segment connecting $w_1(v)$ and $w_2(v)$ as the set of wrench points induced by a finger at v.

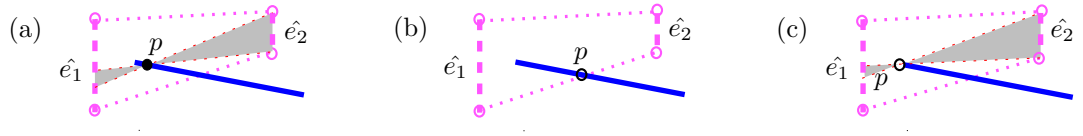


Figure 3.4: A range of points on $\hat{e}_1$ and $\hat{e}_2$ whose line segment contains p in the interior. In (b) and (c), p is denoted by an open disc, because that point is not included in the intersection region.

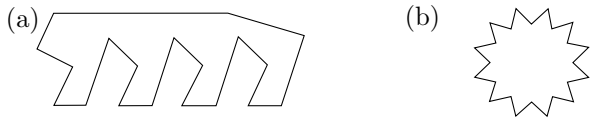


Figure 3.5: (a) This polygon has no concave vertex pair that allow a form-closure grasp. (b) This polygon has $O(m^2)$ concave vertex pairs that allow a form-closure grasp.

Figure 3.4 shows a range on $\hat{e}_1$ and $\hat{e}_2$ when a point p is given. We compute the ranges for all points in the intersection area. As p moves, the ranges gradually grow and shrink. It is enough to compute the ranges for the points along the intersection region boundary. The intersection region has a constant complexity, because trapezoids and line segments have constant complexities. Note that the non-vertical boundaries of $\hat{e}_1\hat{e}_2$ and the endpoints of $s(v)$ are not included in the intersection region.

3.2.1 Two concave vertices

We wish to report all pairs of concave vertices that allow form-closure grasps by placing two frictionless point fingers at these vertices. We assume that the concave vertices have already been identified. For each concave vertex v, we compute $s(v)$ on Γ, the projection of the line segment connecting $w_1(v)$ and $w_2(v)$. By Lemma 2.7, two concave vertices v and v' have a form-closure grasp with two frictionless point fingers, if and only if the interiors of $s(v)$ and $s(v')$ form a red-blue intersection on Γ.

The family $\{s(v)\}$ consists of at most $4m$ red and blue segments on Γ. It remains to compute all red-blue intersections in this set, which can be solved in time $O(m^{4/3}\log^{1/3} m + K)$, using the segment intersection algorithm. The following theorem summarizes the result. Note that the total output size K is $O(m^2)$, but it can be zero. See Figure 3.5. In the worst case, K is $O(m^2)$, but in most cases, K is smaller than this. The algorithm does not check all possible pairs to report $O(m)$ pairs, for example, which is an advantage of output-sensitive algorithms.

Theorem 3.1 *Given a polygon with m concave vertices, all K form-closure grasps with two frictionless point fingers at two concave vertices can be computed in time $O(m^{4/3}\log^{1/3} m + K)$.*

3.2.2 One concave vertex and two edges

Form closure may also be achieved by placing three frictionless point fingers, one at a concave vertex v, and one on each of two edges e_1 and e_2. We now give an algorithm to report all such triples (v, e_1, e_2). Again, we have four wrenches: $w_1 \in \hat{e}_1$, $w_2 \in \hat{e}_2$, and the two wrenches $w_3(v)$ and $w_4(v)$, the endpoints of $s(v)$. All sets of line segments $\overline{w_1w_2}$ form a trapezoid $r(e_1, e_2)$. If $s(v)$ intersects $r(e_1, e_2)$ in the interior, there exists $w_1 \in \hat{e}_1$ and $w_2 \in \hat{e}_2$, such that $s(v)$ intersects $\overline{w_1w_2}$ in the interior. Therefore by Lemma 2.7, a triple (v, e_1, e_2) allows a form-closure grasp with

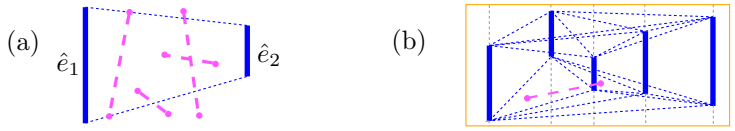


Figure 3.6: (a) Red line segments intersecting a blue trapezoid in the interior. (b) An arrangement of blue trapezoids and red line segments.

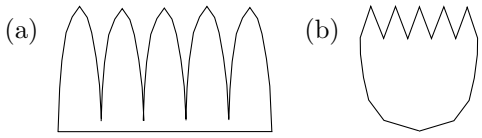


Figure 3.7: (a) This polygon has no triple of a concave vertex and two edges that allow a form-closure grasp, and (b) this polygon has $\Theta(mn^2)$ such triples.

three frictionless point fingers, if and only if the blue part of $s(v)$ intersects a red part of trapezoid $r(e_1, e_2)$, or vice versa.

There are m choices for $s(v)$ and $O(n^2)$ trapezoids induced by n edge wrench sets: at most four trapezoids for each pair of edges. It remains to solve the following problem: given a set of m line segments and a set of $O(n^2)$ trapezoids, find all intersections between a line segment and a trapezoid. See Figure 3.6. We observe that a segment s intersects a trapezoid r, if and only if s lies in r, or s intersects one of the sides of r. We report those that belong to each of the two cases separately.

For the first we use a *triangle search structure* of Corollary 2.1. We build, in $O(m^2 \log m)$ time, a triangle search structure on the set of midpoints of the m segments: this permits queries with a trapezoid (by decomposing it into triangles), identifying the k points inside the trapezoid in $O(\log^3 m + k)$ time. The triangle search structure will report the segments whose midpoints lie in r, but they intersect one of the sides of r. However, they will be reported at most twice. For the second we use a *segment intersection structure* of Corollary 2.3. We build, in $O(m^2 \log^2 m)$ time, a segment intersection structure for segment intersection queries on the $O(m)$ segments. Finding all k segments intersecting a given trapezoid boundary takes $O(\log^4 m + k)$ time.

The total output size K is $O(mn^2)$, but it can be zero. Figure 3.7 shows the polygons with $\Theta(mn^2)$ triples and zero triple of a concave vertex and two edges that yield form-closure grasps.

Theorem 3.2 *Given a polygon with m concave vertices and n edges, all K combinations of one concave vertex and two edges that yield form-closure grasps with three frictionless point fingers can be computed in time $O(n^2 \log^4 m + K)$.*

3.2.3 Two concave vertices and one edge

Placing two point fingers at a pair of concave vertices v, v' may not achieve form closure. Placing one more finger in the interior of an appropriate edge e, however, can achieve form closure with those at v and v'. Here, we present an output-sensitive algorithm to report all such triples (v, v', e). Consider a pair of concave vertices v, v' that does not achieve form closure. Let w_1, w_2 and w_3, w_4 be the wrenches induced by v and v', respectively, and let $W := \{w_1, w_2, w_3, w_4\}$. By Theorem 2.1 the origin O does not lie in the interior of the convex hull of W. An additional finger on e achieves form closure if and only if O lies in the interior of the convex hull of $W \cup \{w\}$, where w is the wrench induced by the finger.

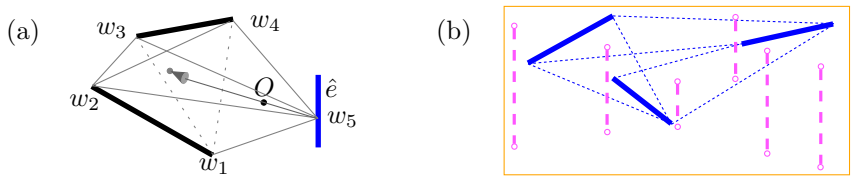


Figure 3.8: (a) The convex hull of $w_1, \cdots, w_5$ contains O strictly inside. (b) An arrangement of the blue convex hulls and the red line segments.

Let $W' := W \cup \{O\}$. The convex hull of W' is a convex polytope with four or five vertices,[3] one of which is O. Consider a facet f_i incident to O, and let H_i be the open half-space bounded by the supporting plane of f_i not containing W'. If O lies in the interior of the convex hull of $W \cup \{w\}$, for some w, then $w \in H_i$ for all i's. Conversely, if this is true for every facet incident to O, then O does lie in the interior of the convex hull of $W \cup \{w\}$.

It follows that an edge e can achieve form closure together with v and v' if and only if the edge wrench segment $\hat{e}$ intersects the intersection of three or four half-spaces. The bounding planes of these half-spaces pass through O, so we can again project everything onto a two-dimensional screen. Here, we do not wish to identify wrenches that are symmetric around the origin, so we use a screen Γ' enclosing the origin as follows:

$$\Gamma' := \{(\eta_x, \eta_y, \tau)^T \mid \max(|\eta_x|, |\eta_y|) = 1, \tau \in \mathbb{R}\}.$$

To prevent degeneracies, we would turn the screen a little so that no segment is projected onto an edge of the screen. We project the n segments $\hat{e}$ onto Γ', build a triangle search structure on their endpoints, and a segment intersection structure on the segments themselves.

For the triangle search structure we use a structure by Matoušek again; however, this time we use the variant that allows us to balance preprocessing and query time. More precisely, we can choose a parameter M (in fact the size of the structure) such that $n \leq M \leq n^2$, and will get a preprocessing time of $O(n^{1+\varepsilon} + M \log^{\varepsilon} n)$, for an arbitrarily small constant $\varepsilon > 0$, and a query time of $O((n/\sqrt{M}) \log^3 \frac{M}{n} + k)$. We choose $M = n^2/\log_m^{\varepsilon} n$, to get a preprocessing time which is, in any case, $O(n^2 \log^{\varepsilon} m)$, and a query time of $O((\log_m^{\varepsilon/2} n) \log^3 n + k)$. With $\varepsilon \leq 2$ this is certainly $O(\log^4 n + k)$.

Before we choose the segment intersection structure, observe that all segments to be stored are vertical. A query segment $q = \overline{q_0 q_1}$, where $q_i = (x(q_i), y(q_i))$, intersects a stored segment $s = \overline{s_0 s_1}$, where $s_i = (x(s_i), y(s_i))$, if and only if the following two conditions are met:

- s_0 lies above $l(q)$ while s_1 lies below $l(q)$ (or the other way around), and
- $x(s)$ lies between $x(q_0)$ and $x(q_1)$.

Therefore, we can solve our query problem with an order-2 structure, as explained in the previous section. The structure stores tuples (s_0, s_1), and stores the sets P_v associated with internal nodes in order-1 trees sorted by x-coordinate. We can pre-sort all segments by x-coordinate as an initialization step, and keep them sorted while distributing and copying them to subtrees, so that no further sorting is necessary. Thus, the complete structure can be constructed in the same time bound as a normal order-2 structure: with $t = \log n$, we get construction time $O(n^2)$. The query time of an order-2 structure with $t = \log n$ is normally $O(\log^2 n + k)$, but in this case, we cannot just report all

[3] If W' has four vertices, one of the wrenches is redundant. This means that form closure could also be achieved by placing point fingers on e, at one of the vertices v or v', and on one of the edges incident to the other vertex. This triple will be reported by the algorithm given above for finding all combinations of one concave vertex and two edges that yield a form-closure grasp.

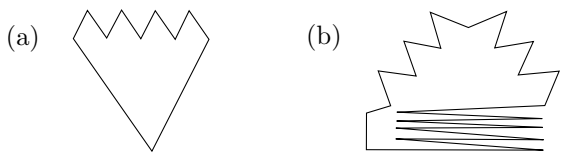


Figure 3.9: (a) This polygon has no triple of two concave vertices and one edge that allow form-closure grasps, and (b) this polygon has $\Theta(m^2 n)$ such triples.

contents of the internal nodes found: we have to do a binary search to report only those segments with x-coordinates between $x(q_0)$ and $x(q_1)$. This increases the query time to $O(\log^3 n + k)$.

In total, both data structures are built in $O(n^2)$ time. We now consider each pair (v, v') of concave vertices in turn. We compute the wrenches W induced by the two vertices, the convex hull of $W \cup \{O\}$, and the intersection R of the three or four relevant half-spaces. We then compute $R' := R \cap \Gamma'$, a polygonal area of constant complexity. We triangulate R', and find the k segment endpoints inside R' by triangle range queries in time $O(\log^4 n + k)$. Furthermore, we find all k segments intersecting the boundary of R' in time $O(\log^3 n + k)$ by a constant number of segment intersection queries. Since there are $\Theta(m^2)$ pairs of concave vertices, the total running time is $O(n^2 + m^2 \log^4 n + k)$.

To list *all* triples of two concave vertices and one edge that yield a form-closure grasp, we should also run the algorithm of Section 3.2.1, to get, in time $O(m^{4/3} \log^{1/3} m + k)$, all k pairs of concave vertices that yield a form-closure grasp, and combine the result with every edge of the polygon.

Theorem 3.3 *Given a polygon with m concave vertices and n edges, all K combinations of two concave vertices and one edge that yield form-closure grasps with three frictionless point fingers can be computed in time $O(n^2 + m^2 \log^4 n + k)$.*

It would be possible to trade some of the dependency on n in this bound for dependency on m, by exploiting the trade-off between preprocessing and query time for triangle search and intersection search structures. However, in the end it would not affect the final bounds for describing all three-point form-closure grasps, as that requires running the $O(n^2 \log^4 m + K)$-time algorithm from the previous section anyway. The latter will dominate the bound on the total running time. The total output size K is $O(m^2 n)$, but it can be zero. Figure 3.9 shows the polygons with $\Theta(m^2 n)$ triples and zero triple of two concave vertices and one edge that yield form-closure grasps. Here as well, our algorithm does not check all possible triples to report K triples with form-closure grasps.

3.2.4 Three concave vertices

A triple of concave vertices (v_1, v_2, v_3) induces three sets of wrench points. Each set is a line segment connecting two wrench points. Let $w_1, \cdots, w_6$ be the end points of the three line segments. Three point fingers in these vertices put an object in form closure if the convex hull of the six wrenches contains the origin in its interior. We can distinguish two cases:

1. a subset of five wrenches already contains the origin in the interior of its convex hull, and thus achieves form closure;

2. no subset of five wrenches contains the origin in the interior of its convex hull.

In the first case, only two of the concave vertices contribute two wrenches to the convex hull. The finger in the third vertex contributes only one wrench: it could just as well have been placed close

by on the corresponding edge which is incident to that vertex. The first case is thus very similar to the case discussed in Section 3.2.3. The algorithm of that section can easily be adapted to list all such cases. We will just use the triangle search structure only, not the segment intersection structure, and store only the edge end points that are actually concave vertices. Building the data structure takes $O(m^2 \log^{\varepsilon} m)$ time; we do $O(m^2)$ queries in $O(\log^3 m + k)$ time each; thus, we can list all triples of concave vertices of the first case in time $O(m^2 \log^3 m + K)$.

For the second case, we will make use of the following lemma from the theory of positive bases [34, 48]:

Lemma 3.4 *Let S be any set of six points in $\mathbb{R}^3$ such that the convex hull of S contains the origin in its interior, but no subset of five points of S contains the origin in the interior of its convex hull. It follows that S consists of six points on three lines through the origin: on each line, one point to each side of the origin.*

It follows that the wrenches induced by the three concave vertices must form three pairs of opposite wrenches. Since no vertex finger could induce opposite wrenches itself, it follows that we are looking for triples (v_1, v_2, v_3) where $w_1(v_2) = -w_2(v_1)$, $w_1(v_3) = -w_2(v_2)$, and $w_1(v_1) = -w_2(v_3)$.

A straightforward algorithm is now as follows. We sort all wrenches induced by concave vertices lexicographically. For every concave vertex v_1, we search in the sorted list for matching vertices v_2, that is, vertices v_2 with $w_1(v_2) = -w_2(v_1)$. For each vertex v_2 found, we do another search for a vertex v_3 such that $w_1(v_3) = -w_2(v_2)$ and $w_2(v_3) = -w_1(v_1)$. If such a vertex v_3 is found, we report the triple (v_1, v_2, v_3).

The sorting is done in $O(m \log m)$ time. The query for v_2, and testing for a matching v_3, takes $O(\log m)$ time per candidate-v_2 which is tested, which amounts to $O(m \log m)$ in the worst case. Searching for matching q and r for each vertex v_1 thus takes $O(m^2 \log m)$ time.

In total, both cases can be dealt with in $O(m^2 \log^3 m + K)$ time.

Theorem 3.5 *Given a polygon with m concave vertices, all K sets of three concave vertices that yield form-closure grasps with three frictionless point fingers can be computed in time $O(m^2 \log^3 m + K)$.*

3.3 Computing all form-closure grasps for rectilinear polygons

In this section, we propose efficient computations of all combinations of edges and concave vertices of a rectilinear polygon that allow form-closure grasps with at most four frictionless point fingers. The time complexities of the algorithms for rectilinear polygons are lower than that for arbitrary polygons, because rectilinear polygons have only four different normal directions, namely, $(1,0)^T$, $(-1,0)^T$, $(0,1)^T$ and $(0,-1)^T$.

We divide the edges into four families E, W, N and S. We let E, W, N and S be the sets of edges whose normal directions are $(1,0)^T$, $(-1,0)^T$, $(0,1)^T$ and $(0,-1)^T$, respectively. See Figure 3.10 (a) – (d). We also divide concave vertices into four families according to the incident edge families, namely, EN, WN, ES and WS. A finger at a vertex from EN induces a set of lines of force, which lie between $(0,1)^T$ and $(1,0)^T$. The lines of force induced by a finger at a vertex from WN, ES or WS lie between $(0,1)^T$ and $(-1,0)^T$, $(0,-1)^T$ and $(1,0)^T$, and $(0,-1)^T$ and $(-1,0)^T$, respectively. Figure 3.10 (e) – (h) illustrates these sets.

We use the projection scheme described in Section 2.4, but we use a different screen. We define screen Γ as follows: $\Gamma := \Gamma_1 \cup \Gamma_2$, where $\Gamma_1 = \{(\eta_x, 1, \tau)^T \mid \eta_x + \eta_y - 1 = 0, -\varepsilon \leq \eta_x \leq 1 + \varepsilon, \tau \in \mathbb{R}\}$ and $\Gamma_2 = \{(-1, \eta_y, \tau)^T \mid \eta_x - \eta_y - 1 = 0, -\varepsilon \leq \eta_x \leq 1 + \varepsilon, \tau \in \mathbb{R}\}$. These planes

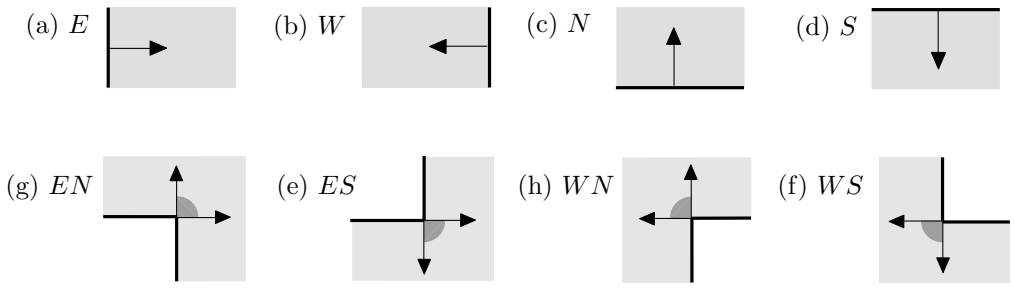


Figure 3.10: (a)–(d) Edges from E, W, N and S. (e)–(h) Concave vertices from EN, WN, ES and WS.

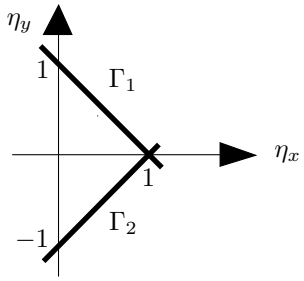


Figure 3.11: Top view of Γ_1 and Γ_2.

are extended by ε on the sides, so that Γ_1 contains two lines $(1,0)$ and $(0,1)$, and Γ_2 contains two lines $(1,0)$ and $(0,-1)$, where the line $(1,0)$ is defined as $\{(1,0,0)^T + \lambda(0,0,1)^T \mid \lambda \in \mathbb{R}\}$, the line $(0,1)$ is defined as $\{(0,1,0)^T + \lambda(0,0,1)^T \mid \lambda \in \mathbb{R}\}$, and the line $(0,-1)$ is defined as $\{(0,-1,0)^T + \lambda(0,0,1)^T \mid \lambda \in \mathbb{R}\}$. Also note that each plane is perpendicular to the (horizontal) $\eta_x\eta_y$-plane. Figure 3.11 shows a top view of these screens.

For simplicity, by "an edge wrench set from E, W, N or S", we mean "the wrench set induced by a finger along an edge from E, W, N or S", and by "a vertex wrench set from EN, WN, ES or WS", we mean "the wrench set of induced by a finger at a vertex from EN, WN, ES or WS". The projections of edge wrench sets are vertical line segments on one of the lines $(0,1)$, $(1,0)$ and $(0,-1)$ on Γ_1 or on Γ_2.

In this section, we enumerate all edge quadruples, concave vertex pairs, triples of one concave vertex and two edges, and triples of two concave vertices and one edge of a rectilinear polygon that yield form-closure grasps with at most four frictionless point fingers. Wentink reported all edge quadruples with form-closure grasps in $O(n \log n + K)$ time—see Section 4.1.2 in [91]. She used a formulation for form closure on the object plane. Here, we also report all edge quadruples with form-closure grasps in the same time bound $O(n \log n + K)$, but with a formulation for form closure in wrench space. Observe that any combination of edges and concave vertices that yield form-closure grasps must contain all normal directions $(1,0)^T$, $(-1,0)^T$, $(0,1)^T$ and $(0,-1)^T$.

3.3.1 Four edges

We wish to enumerate all edge quadruples of a rectilinear polygon that yield a form-closure grasp with four point fingers. As a result of Lemma 2.7, we need to find all edge quadruples e_1, e_2, e_3 and e_4, such that the trapezoid formed by two red $\pi(\hat{e}_1)$ and $\pi(\hat{e}_2)$ intersects the trapezoid formed by two blue $\pi(\hat{e}_3)$ and $\pi(\hat{e}_4)$ in the interior. Since all of $\pi(\hat{e}_1)$, $\pi(\hat{e}_2)$, $\pi(\hat{e}_3)$ and $\pi(\hat{e}_4)$ lie on one

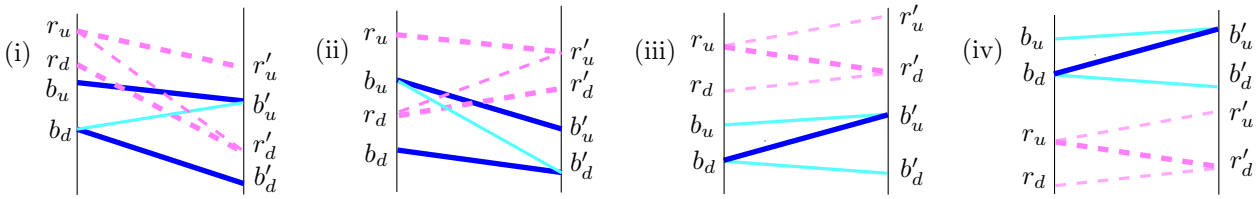


Figure 3.12: (i)–(ii) An illustration of Lemma 3.6. (iii)–(iv) Two cases to consider to prove Lemma 3.6.

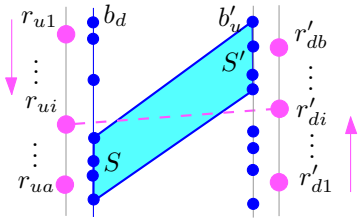


Figure 3.13: The sorted lists of b_u, b'_d, r_d and r'_u.

of the two liens $(1,0)$ or $(0,1)$, we can compute all intersecting red and blue trapezoids on Γ_1 only. Without loss of generality, we assume that $\pi(\hat{e}_1) = r_u r_d$ and $\pi(\hat{e}_3) = b_u b_d$ are on $(0,1)$ line, and that $\pi(\hat{e}_2) = r'_u r'_d$ and $\pi(\hat{e}_4) = b'_u b'_d$ are on $(1,0)$ line. We let r_u, r'_u, b_u and b'_u be the upper points of the corresponding (vertical) segments, and let r_d, r'_d, b_d and b'_d be the lower points of the corresponding (vertical) segments. The following lemma provides a necessary and sufficient condition for a red trapezoid to intersect a blue trapezoid in the interior. Figure 3.12 illustrates Lemma 3.6.

Lemma 3.6 *A red trapezoid $r_u r_d r'_d r'_u$ intersects a blue trapezoid $b_u b_d b'_d b'_u$ in the interior, if and only if one of the following two holds:*

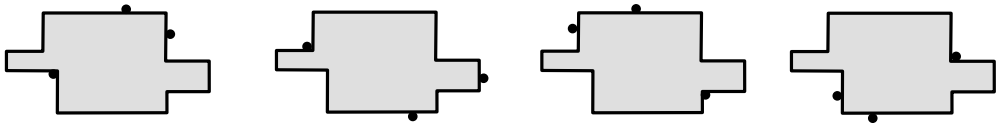
(i) $b_d < r_u$ *and* $b'_u > r'_d$;

(ii) $b_u > r_d$ *and* $b'_d < r'_u$.

Proof: The "if" direction: Condition (i) and (ii) imply that the diagonals $b_d b'_u$ and $b_u b'_d$ intersect $r_u r'_d$ and $r_d r'_u$ respectively. Since $b_d b'_u$ and $b_u b'_d$ are in the blue trapezoid, and $r_u r'_d$ and $r_d r'_u$ are in the red trapezoid, the two trapezoids intersect each other in the interior.

The "only if" direction: Suppose that $r_u r_d r'_d r'_u$ intersects $b_u b_d b'_d b'_u$ in the interior, and that $r_u r'_d$ does not intersect $b_d b'_u$ in the interior, and that $r_d r'_u$ does not intersect $b_u b'_d$ in the interior. We first focus on $r_u r'_d$ and $b_d b'_u$. Since they do not intersect, $r_u > b_d$ and $r'_d > b'_u$, or $r_u < b_d$ and $r'_d < b'_u$. We first look at the case $r_u > b_d$ and $r'_d > b'_u$. Note that $r'_d > b'_u$ imply $r'_u > b'_d$. Hence for $r_u r_d r'_d r'_u$ and $b_u b_d b'_d b'_u$ not to satisfy condition (ii), we must have $b_u < r_d$. Figure 3.12 (iii) depicts this situation. Obviously the supporting line of $b_u b'_u$ separates $r_u r_d r'_d r'_u$ and $b_u b_d b'_d b'_u$, which contradicts that $r_u r_d r'_d r'_u$ intersects $b_u b_d b'_d b'_u$. We can show this similarly when $r_u < b_d$ and $r'_d < b'_u$—see Figure 3.12 (iv). □

Now we explain how to identify all pairs of red and blue trapezoids that satisfy Lemma 3.6. There are $O(n)$ red and blue vertical segments, which are the projections of edge wrench sets. We build sorted lists of b_u, b_d, b'_u and b'_d in $O(n \log n)$ time. We also sort r_u and r_d from top to bottom, then r'_u and r'_d from bottom to top. See Figure 3.13. Here we show how to report all red and blue trapezoids that satisfy the first condition in Lemma 3.6. Those that satisfy the second condition in Lemma 3.6 can be reported similarly. Let $r_{u1}, r_{u2}, \cdots, r_{ua}$ and $r'_{d1}, r'_{d2}, \cdots, r'_{db}$ be the sorted



(a) (EN, W, S) (b) (ES, W, N) (c) (WN, E, S) (d) (WS, E, N)

Figure 3.14: Four cases of a triple a concave vertex and two edges that yield form-closure grasps.

lists of r_u's and r'_d's. We perform a binary search with r_{u1} on the list for b_d, and identify all k blue vertical segments such that $b_d < r_{u1}$ in $O(\log n + k)$ time, and put them in set A. We also perform a binary search with r'_{d1} on the list for b'_u, and identify all k' blue vertical segments such that $b'_u > r'_{d1}$ in $O(\log n + k')$ time, and put them in set A'. We report the Cartesian product of $A \times A'$, i.e. $\{(s, s') \mid s \in A, s' \in A'\}$. When we move from r'_{di} to $r'_{d(i+1)}$ or from r_{ui} to $r_{u(i+1)}$, we do not perform binary searches in the lists; we only check the neighbors in the list of b_d and b'_u, until they satisfy the queries. For r_{u1} we repeat this process for each of r'_{di} ($i = 1, 2, \cdots, b$). We also repeat the whole process for each of r_{ui} ($i = 1, 2, \cdots, a$). We can identify all K red and blue trapezoids that satisfy Lemma 3.6 in total time $O(n \log n + K)$.

Theorem 3.7 *All K edge quadruples of a rectilinear polygon that yield form-closure grasps with four frictionless point fingers can be enumerated in $O(n \log n + K)$ time.*

3.3.2 One concave vertex and two edges

We wish to enumerate all triples of a concave vertex and two edges of a rectilinear polygon that yield a form-closure grasp with three point fingers. A triple of a concave vertex and two edges that yield form-closure grasps belong to one of the following four combinations of edge and vertex families: (EN, W, S), (ES, W, N), (WN, E, S) and (WS, E, N). See Figure 3.14. When a concave vertex is from EN or WS, its projection is a blue or red line segment on Γ_1; when a vertex is from WN (ES), its projection is a blue (red) line segment on Γ_2. See Figure 3.15. As a result of Lemma 2.7, our problem can be formulated as to enumerate all pairs of a red (blue) segment and a blue (red) trapezoid that intersect each other in the interior on Γ_1 or on Γ_2. The following lemma describes a necessary and sufficient condition for a red line segment to intersect a blue trapezoid. Figure 3.16 illustrates Lemma 3.8.

Lemma 3.8 *A red line segment $\overline{rr'}$ intersects a blue trapezoid $b_u b'_u b'_d b_d$ in the interior, if and only if one of the following holds:*

(i) $b_d < r$ *and* $b'_u > r'$*;*

(ii) $b_u > r$ *and* $b'_d < r'$*.*

Proof: It is straightforward to see the "if" direction, so we show "only if" direction. Suppose that $\overline{rr'}$ does not intersect any blue diagonal of $b_u b'_u b'_d b_d$ in the interior, but $\overline{rr'}$ intersects $b_u b'_u b'_d b_d$ in the interior. When $\overline{rr'}$ does not intersect any of the two blue diagonals, it does not intersect $b_u b'_u b'_d b_d$, the convex hull of the two blue diagonals. This is a contradiction. □

We follow the approach described in Section 3.3.1 closely. There are $O(n)$ red and blue vertical segments, and $O(m)$ blue and red line segments. Without loss of generality, we take blue trapezoids and red segments. Those with red trapezoids and blue segments can be identified similarly in the same time bound. We fist report all pairs of a red segment and a blue trapezoid that satisfy the first condition of Lemma 3.8. Those that satisfy the second condition of Lemma 3.8 can be identified

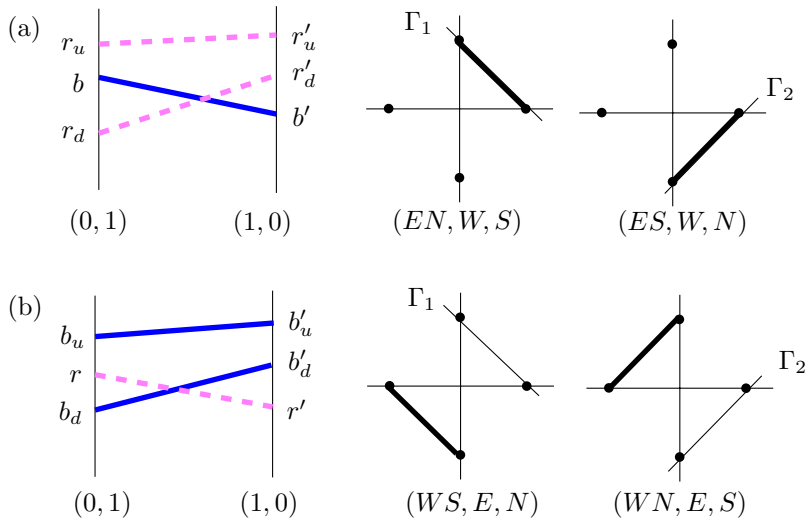


Figure 3.15: Topview of the wrench sets and their projections on Γ of the four combinations.

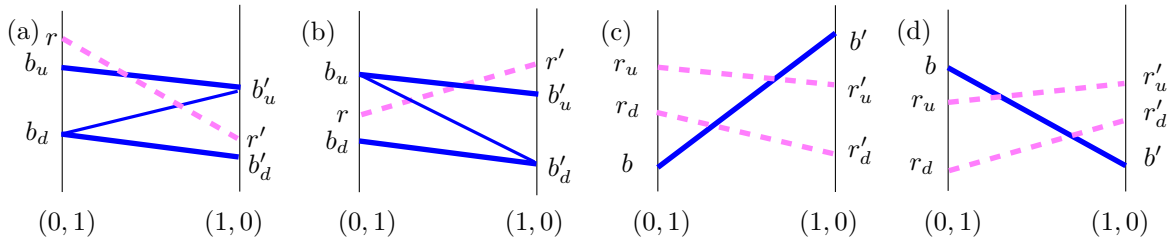


Figure 3.16: The description of Lemma 3.8.

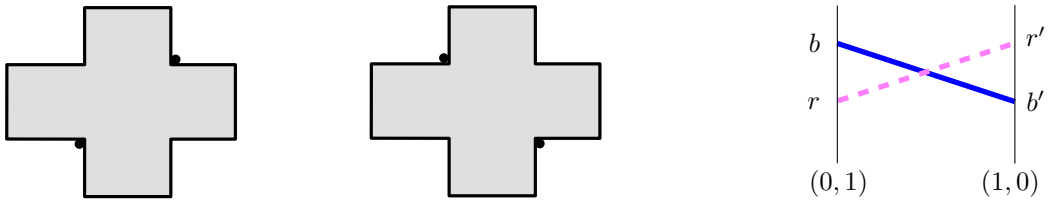
similarly. We sort b_d and b'_u of the blue vertical segments in $O(n \log n)$ time. We query the sorted list of b_d with r to identify the vertical segments such that $b_d < r$, and put them in set A. Then we query the sorted list of b'_u with r' to identify the vertical segments such that $b'_u > r'$, and put them in set A'. We report the Cartesian product of $A \times A'$. We can report all K solutions in $O(n \log n + m \log n + K) = O(n \log n + K)$ time.

Theorem 3.9 *All K triples of one concave vertex and two edges of a rectilinear polygon that yield form-closure grasps with three frictionless point fingers can be enumerated in $O(n \log n + K)$ time.*

3.3.3 Two concave vertices

We wish to enumerate all concave vertex pairs of a rectilinear polygon that yield a form-closure grasp with two point fingers. A pair of two concave vertices that yield form-closure grasps is either (EN, WS) or (ES, WN). See Figure 3.17 (a) and (b). As in Section 3.2.1, it is turned into the problem of reporting all pairs of red and blue line segments that intersect each other in the interior. One difference is that the endpoints lie on two vertical lines (see Figure 3.17 (c)), which makes the intersection checking process easier. The following lemma states a necessary and sufficient condition for two line segments to intersect each other in the interior. Since the proof is trivial, we omit the proof.

Lemma 3.10 *A blue line segment $\overline{bb'}$ intersects a red line segment $\overline{rr'}$ in the interior, if and only if one of the following holds:*



(a) EN and WS (b) ES and WN (c) EN and WS on Γ_1

Figure 3.17: (a)–(b) Two cases of two concave vertices that yield a form-closure grasp. (c) The projections of their wrench sets.

(i) $b < r$ *and* $b' > r'$*;*

(ii) $b > r$ *and* $b' < r'$*.*

There are $O(m)$ red and blue line segments. We store them in a two-level orthogonal range search tree in $O(m \log m)$ time; the left points for b are stored in the first level, and the right points for b' are stored in the second level. Then we query them with a red line segment $\overline{rr'}$. More precisely, we query with r on the first level, and with r' on the second level. All pairs of a red segment and a blue segment satisfying Lemma 3.10 can thus be reported in $O(m \log^2 m + K)$ time.

Theorem 3.11 *All K pairs of two concave vertices of a rectilinear polygon that yield a form-closure grasp with two frictionless point fingers can be enumerated in $O(m \log^2 m + K)$ time.*

3.3.4 Two concave vertices and one edge

We wish to enumerate all triples of two concave vertices and one edge of a rectilinear polygon that yield a form-closure grasp with three frictionless point fingers. Such a triple of two concave vertices and an edge belongs to one of the two cases: when the two concave vertices induce three different normal directions, and when they induce four different normal directions. Two vertices induce three different normal directions, when the two vertices and an edge belong to one of the following four combinations: (EN, ES, W), (WN, WS, E), (EN, WN, S) and (ES, WS, N). See Figure 3.18 (a). Two vertices induce four different normal directions, when they are from (ES, WN) or (EN, WS). Figure 3.18 (b) shows all possible combinations of two such vertex pairs and a face that yield a form-closure grasp.

We first look at the first case when the vertices induce three different normal directions. Without loss of generality, we take a triple (v_{EN}, v_{WN}, e_S) from families of EN, WN and S, and we take their projections on Γ_1. Note that the convex hulls of the red points and the blue points are triangles,[4] whose sides are either on $(0, 1)$ line or on $(1, 0)$ line. Observe that $\pi(\hat{e}_S)$ has two red points r_u and r_d, and $\pi(\hat{v}_{EN})$ has two blue points, one on $(0, 1)$ line, and the other blue point b' on $(1, 0)$ line. Also observe that $\pi(\hat{v}_{WN})$ has one red point r' on $(1, 0)$ line, and one blue point on $(0, 1)$ line. Among the blue points of $\pi(\hat{v}_{EN})$ and $\pi(\hat{v}_{WN})$ on $(0, 1)$ line, we let b_u denote the uppermost point, and let b_d denote the lowest one. See Figure 3.19. Here, we report all pairs of a red triangle and a blue triangle that intersect each other in the interior, where the two triangles are induced by two vertices from EN and WN, and an edge from S. The following lemma provides a

[4] Let A_r be a set of wrench points such that their projections on Γ are red, and A_b be a set of wrench points such that their projections are blue. The projection of the convex hull edges of the points in A_r is the convex hull edges of the red points on Γ, and the projection of the convex hull edges of the points in A_b is the convex hull edges of the blue points on Γ.

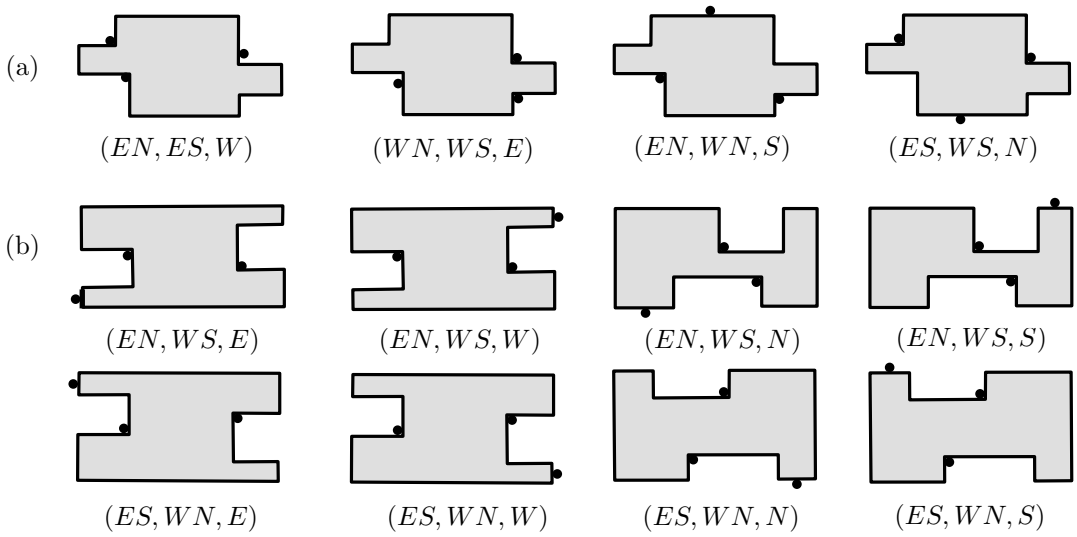


Figure 3.18: (a) Four different combinations of a triple of two concave vertices and an edge, such that it yields a form-closure grasp, and that the two concave vertices induce three different normal directions. (b) Eight different combinations of a triple of two concave vertices and an edge, such that it yields a form-closure grasp, and that the two concave vertices induce four different normal directions.

necessary and sufficient condition for a red triangle to intersect a blue triangle in the interior. Since the proof of the following lemma is similar to that of Lemma 3.6, we omit the proof.

Lemma 3.12 *A blue triangle $bb'_ub'_d$ (b_ub_db') intersects a red triangle $rr'_ur'_d$ (r_ur_dr'), if and only if one of the following four holds:*

(i) $b_u > r_d$ *and* $b' < r'$*;*

(ii) $b_d < r_u$ *and* $b' > r'$*.*

To enumerate all red and blue intersecting triangles that satisfy the first condition of Lemma 3.12, we first sort b_u and b'. We query the sorted list of b_u with r_d to identify the vertical segments such that $b_u > r_d$, and put them in set A. Then we query the sorted list of b' with r' to identify the

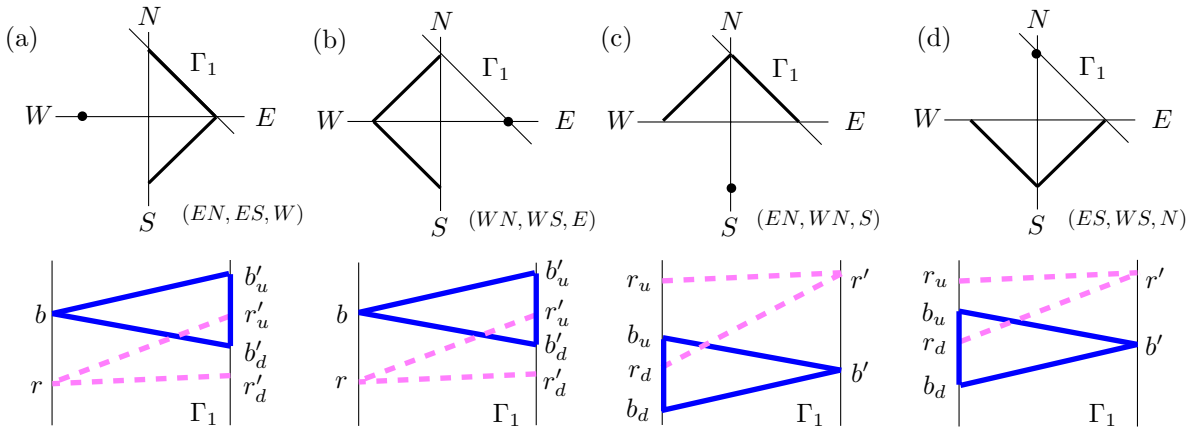


Figure 3.19: Above is the topview of the wrench sets of two concave vertices and an edge. Below is the corresponding red and blue triangle pairs on Γ_1.

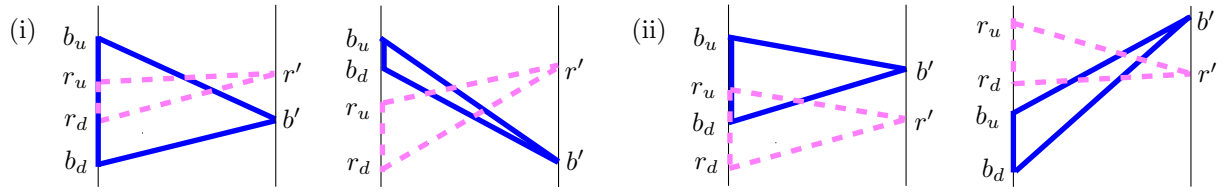


Figure 3.20: Some cases of a red triangle and a blue triangle that satisfy condition (i) and (ii) of Lemma 3.12.

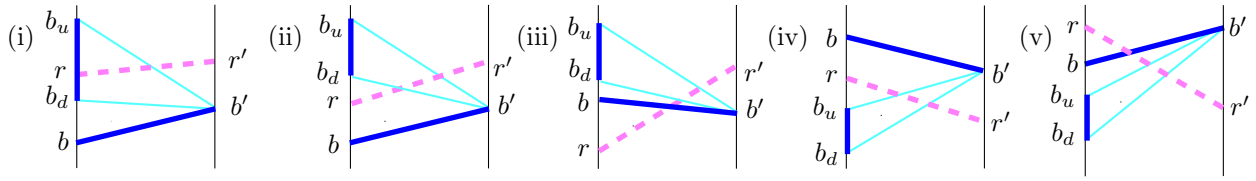


Figure 3.21: An illustration of Lemma 3.13.

vertical segments such that $b' < r'$, and put them in set A'. We report the Cartesian product of $A \times A'$. All red and blue intersecting triangles that satisfy the second condition of Lemma 3.12 can be identified similarly.

There are $O(nm)$ red triangles, because a concave vertex and an edge induce a red triangle. There are $O(m^2)$ blue triangles, because two concave vertices induce a blue triangle. Hence we have $O(m^2)$ points to store, and $O(nm)$ queries. Sorting $O(m^2)$ points takes $O(m^2 \log m)$ time, and we can report k intersecting blue triangles for a red query triangle in $O(\log m + k)$ time. We can report similarly all triples of two concave vertices and an edge from other combinations that yield form-closure grasps. One difference is that sometimes there are $O(nm)$ points to store and $O(m^2)$ queries, which leads to the time complexity of $O(nm \log n + K)$. Thus the total time complexity of this case is $O(nm \log n + K)$.

Now we look at the second case when the vertices induce four different normal directions. Without loss of generality, we take a triple of two concave vertices and an edge (v_{EN}, v_{WS}, e_N). The projections of the edge wrench set and the vertex wrench sets on Γ_1 will be as follows. A blue endpoint b of $\pi(\hat{v}_{EN})$, the blue segment $\overline{b_u b_d}$[5] of $\pi(\hat{e}_N)$ and a red endpoint r of $\pi(\hat{v}_{WS})$ are on the line $(0, 1)$. The other blue endpoint b' of $\pi(\hat{v}_{EN})$ and the other red endpoint r' of $\pi(\hat{v}_{WS})$ are on the line $(1, 0)$. See Figure 3.21. If a red segment and a blue triangle intersect each other in the interior, the corresponding set of two vertices and an edge yields a form-closure grasp. The following lemma provides a necessary and sufficient condition for a pair of a red segment and a blue triangle to intersect each other in the interior. Figure 3.21 illustrates Lemma 3.13. This lemma can easily be modified to check if a blue segment and a red triangle intersect each other in the interior.

Lemma 3.13 *A red segment $\overline{rr'}$ intersects the convex hull of $\overline{bb'}$ and $\overline{b_u b_d}$ in the interior, if and only if one of the following holds:*

(i) $b_d < r < b_u$*;*

(ii) $b < r$ *and* $r < b_d$*;*

(iii) $r < b_d$ *and* $r < b$ *and* $r' > b'$*;*

(iv) $b_u < r$ *and* $r < b$*;*

[5]Among the two endpoints of $\pi(\hat{e}_N)$, we let b_u denote the upper point, and b_d denote the lower point.

(v) $r > b_u$ *and* $r > b$ *and* $r' < b'$.

Proof: When condition (i) or (ii) or (iv) holds, r is in the convex hull of b and $\overline{b_u b_d}$, thus $\overline{rr'}$ intersects the convex hull of $\overline{bb'}$ and $\overline{b_u b_d}$ in the interior. Condition (iii) ($r < b$ and $r' > b'$) and (iv) ($r > b$ and $r' < b'$) imply that $\overline{rr'}$ intersects $\overline{bb'}$ in the interior. Hence $\overline{rr'}$ intersects the convex hull of $\overline{bb'}$ and $\overline{b_u b_d}$ in the interior. □

Here we describe how we report all pairs of a red segment and a blue triangle that satisfy Lemma 3.13. There are $O(n)$ choices for $\overline{b_u b_d}$ and $O(m)$ choices for $\overline{rr'}$ and $\overline{bb'}$. To identify all pairs of a red segment and a blue triangle that satisfy condition (i) of Lemma 3.13, we build an interval tree on blue vertical segments on $(0, 1)$ line in $O(n \log n)$ time. We report all k intervals where r of a red query segment $\overline{rr'}$ lies in $O(\log n + k)$ time. Any segment $\overline{bb'}$ from the $O(m)$ blue segments with any pair $\overline{b_u b_d}$ and $\overline{rr'}$ of the reported pairs will have a red and blue intersection in the interior.

To identify all pairs of a red segment and a blue triangle that satisfy the condition (ii) of Lemma 3.13, we build a binary search trees on b_d in $O(n \log n)$ time. For a given red query segment $\overline{rr'}$, we find, in $O(\log n + k)$ time, all blue segments $\overline{b_u b_d}$ such that $b_d > r$, and put them in set A_1. We also build a binary search tree on $O(m)$ points for b in $O(m \log m)$ time. Then we find, in $O(\log m + k)$ time, all k blue segments $\overline{bb'}$ such that $b < r$, and put them in set A_2. The Cartesian product of $A_1 \times A_2$ for a query segment $\overline{rr'}$ satisfy condition (ii) of Lemma 3.13. We can identify all triples of $\overline{bb'}$, $\overline{b_u b_d}$ and $\overline{rr'}$ that satisfy condition (iv) similarly in the same time bound.

To identify all pairs of a red segment and a blue triangle that satisfy the condition (iii) of Lemma 3.13, we build a two-level orthogonal search tree on $O(m)$ blue segments in $O(m \log m)$ time; b and b' are at the first and the second level respectively. We find, in $O(\log^2 m + k)$ time, all k blue segments $\overline{bb'}$ such that $b > r$ and $b' < r'$, and put them in set A_3. The Cartesian product of $A_1 \times A_3$ with $\overline{rr'}$ satisfy condition (iii) of Lemma 3.13. We can identify all triples of $\overline{bb'}$, $\overline{b_u b_d}$ and $\overline{rr'}$ that satisfy condition (v) similarly in the same time bound.

Hence in total time $O(n \log n + m \log^2 m + K)$, we can report all K triples of two concave vertices and an edge where the two vertices induce four different normal directions. The following theorem summarizes the result.

Theorem 3.14 *All the triples of two concave vertices and an edge of a rectilinear polygon P that yield form-closure grasps with three frictionless point fingers can be enumerated in $O(nm \log n + K)$ time.*

3.4 Conclusion

We proposed efficient output-sensitive algorithms to report all sets of edges and concave vertices of an arbitrary polygon and a rectilinear polygon that yield form-closure grasps. Our approach reduced the dimension of the problem by projections. In particular, when the polygons are rectilinear, the problems boil down to orthogonal range search problems. Another advantage of our approach is that the reformulated problem on planes can be solved with other tools, if one can find simpler, more efficient and more suitable ones for a given purpose or criteria.

Chapter 4

Computing All Form-Closure Grasps of a Planar Semi-Algebraic Set

In this chapter, we propose the first efficient output-sensitive algorithms to compute all form-closure grasps of a set of planar curved objects, which is called semi-algebraic sets. A planar semi-algebraic set is a bounded set on a plane, whose boundary is composed of a set of algebraic arcs. Many researchers have studied the problem of synthesizing immobilizing grasps of a planar curved object [20, 37, 44, 51, 56, 67]. However, no algorithm can enumerate *all* immobilizing grasps of planar curved objects, except [44]. Jia proposed an algorithm to report all antipodal grasps with two frictional fingers, but the asymptotic bound is high.

In this chapter,[1] we propose an output-sensitive algorithm to report all combinations of arcs and concave vertices that admit at least one form-closure grasp with at most four frictionless point fingers. More precisely, the combinations are: (i) arc quadruples and arc triples with four fingers, (ii) triples of one concave vertex and two arcs with three fingers, (iii) pairs of a concave vertex and an arc with three fingers, and (iv) triples of two vertices and an arc with three fingers. (Recall that the case of two points at two vertices was already solved in Chapter 3; the algorithm applies not only to polygonal parts but also to curved parts.) We focus on reporting all combinations of arcs and concave vertices, because once we have a set of arcs and concave vertices, we can compute form-closure grasps in constant time as mentioned in Chapter 3. Four fingers can achieve form closure on arc pairs and on one single arc (such as an ellipse) as well as on arc quadruples and triples, but we treat only arc quadruples and triples, because of an efficiency issue. Section 4.2 has more details. To deal with this algorithmic challenge of identifying these combinations, we use a geometric condition in wrench space and the approach taken in Chapter 3.

This chapter is organized as follows. In Section 4.1, we introduce notations, projection schemes, wrench shapes and data structures for intersection search problems. In Section 4.2, we propose output-sensitive algorithms to report all arc triples and quadruples that allow form-closure grasps with four frictionless point fingers. We also propose, in Section 4.3, output-sensitive algorithms to report all combinations of concave vertices and arcs that allow form-closure grasps with three frictionless point fingers. Discussion will follow after these in Section 4.4.

[1] This chapter is based on "Output-sensitive computation of all form-closure grasps of a part bounded by algebraic arcs" [25] by J.-S. Cheong and A.F. van der Stappen in ICRA (2005).

4.1 Preliminaries

We let n be the number of arc pieces of a semi-algebraic set P, and m be that of concave vertices of P. The problem of enumerating all combinations of arcs and concave vertices with form-closure grasps can be reformulated in terms of wrenches as follows: Given n arcs and m line segments in three-dimensional wrench space, report all combinations of arcs and line segments that contain four points that positively span wrench space. We will explain why we have arcs and line segments in wrench space in Section 4.1.1.

We closely follow the approach taken in Chapter 3. We project the wrench points on screen Γ, which is as defined in Section 2.4.1. Lemma 2.7 states that a set of arcs and line segments with four points that positively span wrench space must form a red and blue intersection on Γ. Thus our problem becomes to report all sets on Γ that have red and blue intersections.

This section is organized as follows. In Section 4.1.1, we first investigate the shapes of wrenches and their projections, when a finger slides along an arc a. In Section 4.1.2, we define two-arc-cell and one-arc-cell, and show that they have constant complexities. In Section 4.1.3, we introduce the data structures and algorithms to search for all sets of two-arc-cells, one-arc-cells and line segments that have red and blue intersections.

4.1.1 Algebraic arcs, wrenches and their projections

A semi-algebraic set P is a closed set bounded by a set of real algebraic arcs of bounded degree. A *real algebraic arc* is a piece of a real algebraic curve. We call this piece simply an *arc*. A real algebraic curve over field $\mathbb{R}$ satisfies an equation $\Psi(x, y) = 0$, where $\Psi(x, y)$ is a polynomial in x and y with coefficients in $\mathbb{R}$. Throughout the chapter, we assume that $\Psi(x, y) = 0$ has a constant degree. The boundary of P can contain a straight edge, because a line segment is a special case of an algebraic arc with zeros for some coefficients. We let n denote the number of arcs, and m the number of concave vertices of P.

Now we show that the set of wrench points induced by a finger sliding on an arc a is an algebraic arc in wrench space. We call this set the *arc wrench set (of a)*, and denote it by $\hat{a}$. Let $\Psi(x, y) = 0$ represent an algebraic curve that contains a boundary arc a of P. Without loss of generality, let $\Psi(x, y) > 0$ denote the immediate interior of P bounded by a. We also let Ψ_x and Ψ_y denote $\frac{\partial\Psi(x,y)}{\partial x}$ and $\frac{\partial\Psi(x,y)}{\partial y}$ respectively. We assume that $\Psi_x \neq 0$ and $\Psi_y \neq 0$; we can satisfy this condition by taking out the point of an arc where $\Psi_x = 0$ or $\Psi_y = 0$. The wrench (η_x, η_y, τ) at position $\overrightarrow{p} = (x, y)$ on a is $(\Psi_x, \Psi_y, x\Psi_y - y\Psi_x)$. Let $\mathcal{M}$ be a map $\mathcal{M} : \mathbf{x} = (x, y) \in \mathbb{R}^2 \mapsto (\Psi_x, \Psi_y, x\Psi_y - y\Psi_x) \in \mathbb{R}^3$. Since Ψ_x, Ψ_y and $x\Psi_y - y\Psi_x$ are all polynomials in x and y, the image of $\mathcal{M}$ is also algebraic. More precisely, the wrench points of a finger sliding along an arc a forms a semi-algebraic set.

We project the wrench arc $\hat{a}$ onto Γ. Portions of a wrench arc may end up on different planes of Γ and get different colors depending on where they are in wrench space with respect to the planes $\eta_x + \eta_y = 0$ and $\eta_x - \eta_y = 0$ (see Figure 4.1). Portions in region I and III turn into blue and red arcs respectively on Γ_1, and portions in region II and IV turn into red and blue arcs respectively on Γ_2. Let $q = (\Psi_x, \Psi_y, x\Psi_y - y\Psi_x)$ be a point on $\hat{a}$; then $\pi(q) = (\Psi_x/\Psi_y, 1, x - y(\Psi_x/\Psi_y))$ on Γ_1 if q is inside I or III, and $\pi(q) = (1, \Psi_y/\Psi_x, x(\Psi_y/\Psi_x) - y)$ on Γ_2 if q is inside II or IV. We assume that $\Psi_x \neq 0$ and $\Psi_y \neq 0$. We can fulfill this assumption, by cutting the arcs at the point where $\Psi_x = 0$ or $\Psi_y = 0$. Semi-algebraic sets are closed under projections. The projection $\mathcal{M}_1$ on Γ_1 and the projection $\mathcal{M}_2$ on Γ_2 are as follows.

$$\mathcal{M}_1 : \mathbf{x} = (x, y) \in \mathbb{R}^2 \mapsto (\Psi_x/\Psi_y, 1, x - y(\Psi_x/\Psi_y)) \in \mathbb{R}^3$$

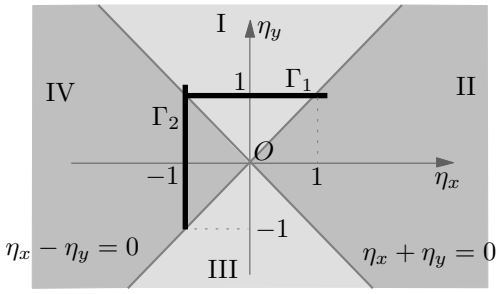


Figure 4.1: Screen Γ in wrench space and four regions I, II, III and IV, viewed from the positive τ axis.

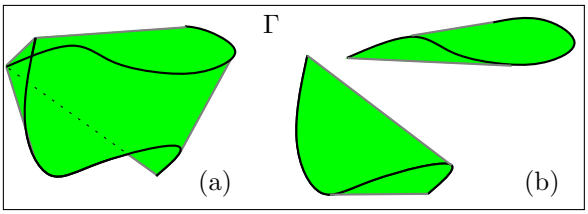


Figure 4.2: (a) A two-arc-cell (b) One-arc-cells

$$\mathcal{M}_2 : \mathbf{x} = (x, y) \in \mathbb{R}^2 \mapsto (1, \Psi_y/\Psi_x, x(\Psi_y/\Psi_x) - y) \in \mathbb{R}^3$$

Therefore the projected arcs are also semi-algebraic sets, thus algebraic arcs.

4.1.2 Two-arc-cells and one-arc-cells

When two fingers slide along two distinct arcs a and a', they induce two wrench points w and w' sliding along $\hat{a}$ and $\hat{a}'$. The line segment connecting w and w' will move when w and w' slide along $\hat{a}$ and $\hat{a}'$. We let $r(a, a') := \bigcup\{\pi(\overline{ww'}) \mid w \in \hat{a}, w' \in \hat{a}'\}$. We call this set the *two-arc-cell (of a and a')*: it is the region where $\pi(\overline{ww'})$ lie for all $w \in \hat{a}$ and $w' \in \hat{a}'$. Figure 4.2 (a) illustrates a two-arc-cell. A two-arc-cell is a semi-algebraic set with a constant complexity, which is shown in the following lemma.

Lemma 4.1 *A two-arc-cell $r(a, a')$ on Γ is bounded by portions of the arcs $\pi(\hat{a})$ and $\pi(\hat{a}')$ and a constant number of line segments.*

Proof: From the definition of $r(a, a')$, it is evident that the boundary of $r(a, a')$ consists of portions of $\pi(\hat{a})$ and $\pi(\hat{a}')$ and line segments. It remains to show that the number of line segments is constant. Observe that the boundary line segments of $r(a, a')$ is the projections of the common tangent line segments of $\hat{a}$ and $\hat{a}'$. The wrench arcs $\hat{a}$ and $\hat{a}'$ have a constant number of common tangent line segments, because they have a constant degree and there are four endpoints in total. This completes the proof. □

When two fingers slide along a single arc a, they also induce two wrench points w and w', both of which slide along $\hat{a}$ independently. The line segment connecting w and w' will move when w and w' slide along $\hat{a}$. We define $r(a)$ to be $\bigcup\{\pi(\overline{ww'}) \mid w \in \hat{a}, w' \in \hat{a}\}$. We call this set the *one-arc-cell (of a)*: it is the region where all $\pi(\overline{ww'})$ lie for all pairs of points w and w' on $\hat{a}$. One-arc-cell $r(a)$ turns out to be the convex hull of $\pi(\hat{a})$, which is again a semi-algebraic set—see Lemma 4.2. Figure 4.2 (b) illustrates one-arc-cells. A one-arc-cell also has a constant complexity as shown in the following lemma. The proof is basically the same as that of Lemma 4.1, thus we omit the proof.

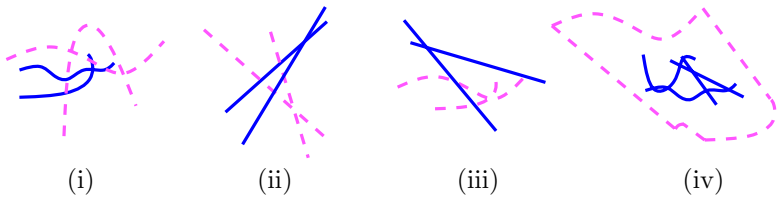


Figure 4.3: The first four types of the red-blue intersections. The dashed entities are in red.

Lemma 4.2 *One-arc-cell $r(a)$ is bounded by portions of $\pi(\hat{a})$ and a constant number of line segments.*

4.1.3 Intersection search algorithms

In this chapter, we need to report all red and blue intersections between semi-algebraic sets, line segments and triangles. These intersections can be identified by checking the following five subproblems on Γ: (i) intersections between red and blue arcs, (ii) intersections between red and blue line segments, (iii) intersections between red (blue) arcs and blue (red) line segments, (iv) red (blue) points contained in blue (red) semi-algebraic sets, or (v) red (blue) points contained in blue (red) quadrilaterals. Figure 4.3 shows the first four types. To handle these intersections, we use the algorithms and query structures below. More detailed information on these algorithms and query structures can be found in Section 2.4.2.

The intersections between red and blue arcs are reported in a brute-force manner; there is no efficient algorithm as far as we know. Fortunately, this does not affect the overall efficiency, as the number of arcs involved is always sufficiently low.

To enumerate all intersections between segments, we use the following two: red-blue line segment intersection algorithm and segment-segment query structure. Among q red and blue line segments, the red-blue line segment intersection algorithm can report K intersecting red and blue segments in $O(q^{4/3}\log^{1/3} q+K)$ time. The segment-segment query structure stores q line segments in $O(q\log^2 q)$ time, and reports k intersecting segments in $O(\log^4 q+k)$ time.

The intersections between red and blue arcs and segments can be identified in two ways as well: red-blue segment-arc intersection algorithm and segment-arc query structure. Among q red and blue line segments and arcs, the red-blue segment-arc intersection algorithm can report K intersecting pairs of a segment and an arc in $O(q^{3/2+\varepsilon}+K)$ time. The segment-arc query structure stores q arcs in $O(q^{2+\varepsilon})$ time, and reports k intersecting arcs in $O(\log q+k)$ time.

We use *semi-algebraic range search structure* and *triangle search structure* to report the intersections between points and semi-algebraic sets, and between points and triangles, respectively. The semi-algebraic range search structure stores q points in $O(q\log q)$-time, and reports all k points in a query semi-algebraic set in $O(q^{1/2+\varepsilon}+k)$-time. In the intersections of type (v), we decompose each quadrilateral into two triangles and query with the triangles. The triangle search structure stores q points in $O(q\log q)$ time, and reports all k points in a query triangle in $O(\log^3 q+k)$-time.

4.1.4 Computing all grasps on a given set of arcs and vertices

Once we are given a set of arcs and/or concave vertices admitting at least one form-closure grasp with at most four point fingers, we can compute the regions representing all grasps on this set in constant time [58, 82]. As an example, consider the case of four arcs a_1, a_2, a_3, and a_4. Every point q in the intersection of the red $r(a_1,a_2)$ and the blue $r(a_3,a_4)$ (or vice versa) corresponds

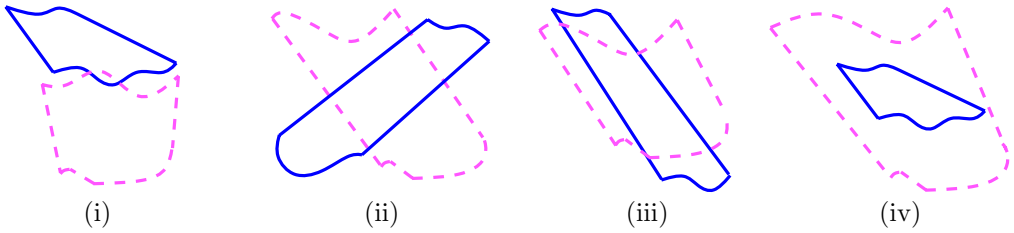


Figure 4.4: Four cases of the red-blue intersections. The dashed entities are in red.

to a set of placements of points along a_1, a_2, a_3, and a_4. The set of grasps associated with q is the Cartesian product of all lines through q intersecting $\pi(\hat{a}_1)$ and $\pi(\hat{a}_2)$, and all lines through q intersecting $\pi(\hat{a}_3)$ and $\pi(\hat{a}_4)$. These line segments form a continuous set. Once we know a point in the intersection region, we can compute the boundary of these sets of line segments in constant time. Furthermore, the boundary changes also continuously. Hence it is enough to compute the intersection region between $r(a_1, a_2)$ and $r(a_3, a_4)$, which has a constant complexity.

4.2 Computing all form-closure grasps with four fingers

In this section, we provide output-sensitive algorithms for reporting all quadruples and triples of arcs admitting at least one form-closure grasp with four frictionless point fingers. All single arcs allowing for grasps with four fingers on each of them can be easily enumerated in linear time, simply by checking each arc in constant time. Since we have to spend at least linear time, this is optimal.

Four fingers on two arcs can also achieve form closure. If all arc pairs allowing for form-closure grasps with four fingers can be computed in subquadratic time, it is more efficient than a naive approach that takes quadratic time. It is open to compute all such arc pairs in subquadratic time. Hence we do not consider the problem of reporting all arc pairs that yield form-closure grasps.

4.2.1 Four arcs

We wish to report all arc quadruples such that four frictionless point fingers on each quadruple (one point finger per arc) achieve form closure. By Lemma 2.7, an arc quadruple (a_1, a_2, a_3, a_4) allows form-closure grasps, if and only if the red two-arc-cell $r(a_1, a_2)$ intersects the blue two-arc-cell $r(a_3, a_4)$ in the interior. To identify all arc quadruples (a_1, a_2, a_3, a_4) admitting form-closure grasps, we must therefore report all intersecting red and blue two-arc-cells.

A red $r(a_1, a_2)$ intersects a blue $r(a_3, a_4)$, if and only if their boundaries intersect, or $r(a_1, a_2) \subseteq r(a_3, a_4)$, or $r(a_3, a_4) \subseteq r(a_1, a_2)$. We observe that if $r(a_1, a_2) \subseteq r(a_3, a_4)$, any point of $\pi(\hat{a}_1)$ lies inside $r(a_3, a_4)$, and if $r(a_3, a_4) \subseteq r(a_1, a_2)$, any point of $\pi(\hat{a}_3)$ lies inside $r(a_1, a_2)$. All intersecting red and blue two-arc-cells can thus be determined by computing all (i) intersecting red and blue boundary arcs, (ii) intersecting red and blue boundary line segments, (iii) blue (red) arcs intersecting red (blue) boundary line segments, and (iv) blue (red) representative points inside red (blue) two-arc-cells. (See Figure 4.4).

There are $O(n^2)$ boundary segments and $O(n)$ boundary arcs for all $O(n^2)$ red and blue two-arc-cells. We take one representative point from each arc, thus we have $O(n)$ representative points. Problem (i) can be solved in $O(n^2)$ time, by checking all red and blue arc pairs. We apply the

red-blue line segment intersection algorithm for problem (ii), which requires $O(n^{8/3}\log^{1/3} n+K)$-time. The segment-arc query structure can report the blue (red) arcs intersecting the red (blue) line segments. The preprocessing time is $O(n^{2+\varepsilon})$-time, and the query time is $O(\log n + k)$. There are $O(n^2)$ query segments. Thus the total time complexity for problem (iii) becomes $O(n^{2+\varepsilon} + K)$. The semi-algebraic range search structure can identify all blue (red) points contained in a red (blue) two-arc-cell. The preprocessing time is $O(n \log n)$ time, the query time is $O(n^{1/2+\varepsilon}+k)$, and there are $O(n^2)$ query two-arc-cells. The total time complexity for problem (iv) becomes $O(n^{5/2+\varepsilon}+K)$. The following theorem summarizes the results.

Theorem 4.3 *Given a planar semi-algebraic set with n arcs, all K arc quadruples admitting a form-closure grasp with four frictionless point fingers can be computed in $O(n^{8/3}\log^{1/3} n + K)$ time.*

4.2.2 Three arcs

We wish to report all arc triples (a_1, a_2, a_3) such that two point fingers on one arc a_1, and one finger on each of the remaining arcs a_2 and a_3 achieve form closure. By Lemma 2.7, such an arc triple allows form-closure grasps, if and only if red (blue) $r(a_1)$ intersects blue (red) $r(a_1)$ in the interior. To identify all arc triples (a_1, a_2, a_3) admitting form-closure grasps with four fingers, we must therefore report all red and blue intersections between a one-arc-cell and a two-arc-cell.

A red (blue) one-arc-cell intersects a blue (red) two-arc-cell, if and only if their boundaries intersect or the one-arc-cell is contained in the two-arc-cell, or vice versa. All red and blue intersecting pairs of a one-arc-cell and a two-arc-cell can thus be determined by computing all (i) intersecting red and blue arcs, (ii) intersecting red and blue segments, (iii) blue (red) arcs intersecting red (blue) boundary line segments, and (iv) blue (red) arcs inside red (blue) arc-cells.

We focus on the case of all intersecting pairs of a red one-arc-cell and a blue two-arc-cell. The case of all intersecting pairs of a red one-arc-cell and a blue two-arc-cell can be treated similarly with the same time bound. There are $O(n)$ red arcs, representative points and boundary segments, and there are $O(n^2)$ blue two-arc-cells, blue boundary segments and $O(n)$ blue arcs in total. Problem (i) can be solved again in $O(n^2)$ time. Problem (ii) can be solved in $O(n^2 \log^4 n + K)$ time as follows. We store the $O(n)$ red boundary segments in a segment-segment query structure in $O(n^2 \log^2 n)$ time, and query with each of the $O(n^2)$ blue boundary segments in $O(\log^4 n + k)$ time.

Problem (iii) can be solved in $O(n^{2+\varepsilon} + K)$ time as follows. We store the $O(n)$ red arcs in a segment-arc query structure in $O(n^{2+\varepsilon})$ time, and query with each of the $O(n^2)$ blue boundary segments in $O(\log n + k)$ time. We also store the $O(n)$ blue arcs in a segment-arc query structure in $O(n^{2+\varepsilon})$ time, and query with each of the $O(n)$ red boundary segments in $O(\log n + k)$ time.

Finally, problem (iv) can be solved in $O(n^{5/2+\varepsilon} + K)$ time as follows. We store the $O(n)$ red representative points in a semi-algebraic range search structure in $O(n \log n)$ time, and query with each of the $O(n^2)$ blue two-arc-cells in $O(n^{1/2+\varepsilon} + k)$ time. We also store the $O(n)$ blue representative points in a segment-arc query structure in $O(n \log n)$ time, and query with each of the $O(n)$ red one-arc-cells in $O(n^{1/2+\varepsilon}+k)$ time. We pair two arcs $\pi(\hat{a})$ and $\pi(\hat{a}')$ from the reported arcs, and compute $r(a, a')$. The set of all such two-arc-cells contain all two-arc-cells contained in the given query one-arc-cell.

Theorem 4.4 *Given a planar semi-algebraic set with n arcs, all K arc triples admitting a form-closure grasp with three frictionless point fingers can be computed in $O(n^{5/2+\varepsilon} + K)$ time.*

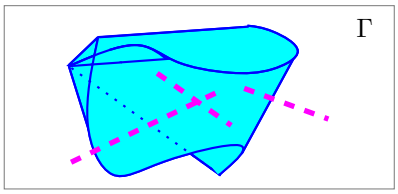


Figure 4.5: An intersecting pair of a blue two-arc-cell and a red line segment.

4.3 Computing all form-closure grasps with at most three fingers

If we take advantage of concave vertices, three frictionless point fingers on a semi-algebraic set can achieve form closure. In this section, we provide output-sensitive algorithms for reporting all combinations of arcs and concave vertices admitting at least one form-closure grasp with three frictionless point fingers. Such combinations include: (i) triples of one concave vertex and two arcs, (ii) pairs of one concave vertex and one arc, and (iii) triples of two concave vertices and one arc.

We let m denote the number of concave vertices of a semi-algebraic set P. The number of arcs are denoted by n. A finger at a concave vertex v induces a line segment in wrench space, and also on Γ—see Section 3.1.1. We let $s(v)$ denote the line segment on Γ, induced by a finger at v.

4.3.1 One concave vertex and two arcs

We wish to report all triples of a concave vertex and two arcs, such that three frictionless point fingers (one finger on each) achieve form closure. A triple (v, a, a') allows a form-closure grasp, if and only if the blue (red) segment $s(v)$ intersects the red (blue) two-arc-cell $r(a, a')$ by Lemma 2.7. See Figure 4.5. Thus our problem becomes to report all blue (red) segments intersecting red (blue) two-arc-cells. We focus on the case of all intersecting pairs of a blue segment and a red two-arc-cell. The case of all intersecting pairs of a red segment and a blue two-arc-cell can be treated similarly with the same time bound.

A blue segment $s(v)$ intersects a red two-arc-cell $r(a, a')$, if and only if $s(v)$ intersects the boundary of $r(a, a')$, or $s(v) \subseteq r(a, a')$. Observe that if $s(v) \subseteq r(a, a')$, the representative midpoint of $s(v)$ lies inside $r(a, a')$. All intersecting red two-arc-cells and blue segments can thus be reported by computing all (ii) intersecting red and blue line segments, (iii) red arcs intersecting blue line segments, and (iv) blue midpoints inside red two-arc-cells.

There are $O(n^2)$ red two-arc-cells and red boundary segments, and $O(n)$ red arcs. There are $O(m)$ blue line segments and representative points. Problem (ii) can be solved in $O(n^2 \log^4 m + K)$ time as follows. We store the $O(m)$ blue segments in a segment-segment query structure in $O(m^2 \log^2 m)$ time, and query with each of the $O(n^2)$ red boundary segments in $O(\log^4 m + k)$ time. Problem (iii) can be solved in $O(n^{3/2+\varepsilon} + K)$ time; we use the segment-arc intersection algorithm. Problem (iv) can be solved in $O(n^2 m^{1/2+\varepsilon} + K)$ time as follows. We store the $O(m)$ blue representative points in a semi-algebraic range search structure in $O(m \log m)$ time, and query with each of the $O(n^2)$ red two-arc-cells in $O(m^{1/2+\varepsilon} + k)$ time.

Theorem 4.5 *Given a planar part with m concave vertices and n algebraic arcs, all K triples of one concave vertex and two arcs admitting a form-closure grasp with three frictionless point fingers can be enumerated in $O(n^2 m^{1/2+\varepsilon} + K)$ time.*

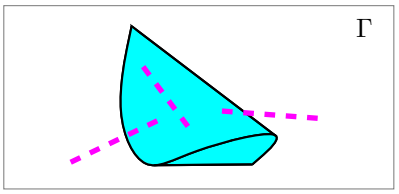


Figure 4.6: An intersecting pair of a blue one-arc-cell and a red line segment.

4.3.2 One concave vertex and one arc

We wish to report all pairs of a concave vertex v and an arc a, such that one point finger at v and two on a achieve form closure. A pair (v, a) allows a form-closure grasp, if and only if the blue (red) segment $s(v)$ intersects the red (blue) one-arc-cell $r(a)$ in the interior by Lemma 2.7. See Figure 4.6. Thus our problem becomes to report all blue (red) segments intersecting red (blue) one-arc-cells in the interior.

Here we focus on the case of all intersecting pairs of a blue segment and a red one-arc-cell. The case of all intersecting pairs of a red segment and a blue one-arc-cell can be treated similarly with the same time bound. A blue segment $s(v)$ intersects a red one-arc-cell $r(a)$, if and only if $s(v)$ intersects the boundary of $r(a)$, or $s(v)$ lies inside $r(a)$. All intersecting pairs of a red one-arc-cell and a blue segment can be identified by computing all red and blue intersections between (ii) line segments, (iii) an arc and a line segment, and (iv) a midpoint and a one-arc-cell.

There are $O(n)$ red one-arc-cells, red boundary segments and red boundary arcs. There are $O(m)$ blue line segments and representative points. When P has a small number of concave vertices relative to n, more precisely, when $m \leq \sqrt{n}$, a naive approach has a better time bound, which is $O(nm)$.

When $m > \sqrt{n}$, we do the following. First, we solve problem (iii) in $O(n^{3/2+\varepsilon} + K)$ time, using a segment-arc intersection algorithm. Problem (iv) can be solved in $O(nm^{1/2+\varepsilon} + K)$ time as follows. We store the $O(m)$ blue representative points in a semi-algebraic range search structure in $O(m \log m)$ time, and query with each of the $O(n)$ red one-arc-cells in $O(m^{1/2+\varepsilon} + k)$ time. A red-blue line segment intersection can be computed efficiently using two algorithms, depending on the size of m relative to n: $n^{1/2} < m < n^{2/3}$ and $n^{2/3} \leq m \leq n$. When $n^{2/3} \leq m \leq n$, a red-blue line segment intersection algorithm can report all pairs of intersecting segments in $O(n^{4/3} \log^{1/3} n + K)$ time. When $n^{1/2} < m < n^{2/3}$, we store the $O(m)$ blue segments in a segment-segment query structure in $O(m^2 \log^2 m)$ time, and query with each of the $O(n)$ red boundary segments in $O(\log^4 m + k)$ time. Thus we can report all intersecting pairs of a red segment and a blue segment in $O(m^2 \log^2 m + K)$-time. In total, it takes $O(n^{3/2+\varepsilon} + K)$ when $\sqrt{n} < m \leq n$.

The following theorem summarizes the result.

Theorem 4.6 *Given a planar part with m concave vertices and n algebraic arcs, all K pairs of a concave vertex and an arc admitting a form-closure grasp with three frictionless point fingers can be computed in $O(mn)$ time when $m \leq \sqrt{n}$, and in $O(n^{3/2+\varepsilon} + K)$ time when $m > \sqrt{n}$.*

4.3.3 Two concave vertices and one arc

We wish to report all triples of two concave vertices and one arc, such that three frictionless point fingers (one finger on each) achieve form closure. Let $r(v, v') := \bigcup\{\pi(\overline{ww'}) \mid w \in s(v), w' \in$

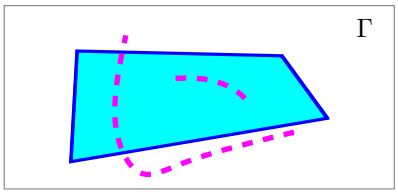


Figure 4.7: An intersecting pair of a blue quadrilateral and a red arc.

$s(v')\}$. Note that $r(v, v')$ is a quadrilateral.[2] A triple (v, v', a) allows a form-closure grasp, if and only if the blue (red) arc $\pi(\hat{a})$ intersects the red (blue) quadrilateral $r(v, v')$ in the interior by Lemma 2.7. See Figure 4.7. Thus our problem becomes to report all red (blue) arcs intersecting blue (red) quadrilaterals in the interior.

Here we focus on the case of all intersecting pairs of a red arc and a blue quadrilateral. The case of all intersecting pairs of a blue arc and a red quadrilateral can be treated similarly with the same time bound. There are $O(m^2)$ blue quadrilaterals and blue boundary segments. There are $O(n)$ red arcs and representative points. A blue segment $s(v)$ intersects a red one-arc-cell $r(a)$, if and only if $s(v)$ intersects the boundary of $r(a)$, or $s(v)$ lies inside $r(a)$. All intersecting pairs of a red arc and a blue quadrilateral can be identified by computing all red and blue intersections between (iii) an arc and a line segment, and (iv) a point and a quadrilateral.

When m is small relative to n, more precisely, when $m \leq \sqrt{n}$, a naive approach has a better time bound, which is $O(nm^2)$. When $m > \sqrt{n}$, we do the following. First, we solve problem (iii) in $O(n^{2+\varepsilon} + K)$ time, using a segment-arc query structure. We store the $O(n)$ red arcs in a segment-arc query structure in $O(n^{2+\varepsilon})$ time, and query with each of the $O(m^2)$ blue boundary segments in $O(\log n + k)$ time. Problem (v) can be solved in $O(n^2 \log n + m^2 \log^3 n + K)$ time as follows. We store the $O(n)$ red representative points in a triangle search structure in $O(n^2 \log n)$ time. We divide each of the $O(m^2)$ blue quadrilaterals into two disjoint triangles, and query the structure with each of the $O(m^2)$ triangles in $O(\log^3 n + k)$ time. These lead to the following theorem.

Theorem 4.7 *Given a planar part with m concave vertices and n algebraic arcs, all K triples of two concave vertices and an arc admitting form-closure grasps with three frictionless point fingers can be reported in $O(nm^2)$ time when $m \leq \sqrt{n}$, and in $O(n^{2+\varepsilon} + K)$ time when $m > \sqrt{n}$.*

4.4 Conclusion

We proposed the first efficient output-sensitive algorithms to report all sets of arcs and concave vertices of a semi-algebraic set that yield form-closure grasps. The projection method in Chapter 3 was general enough to be employed to tackle this problem. We proved that the shapes of the wrench sets induced by a frictionless point finger on an arc, and their projections are algebraic arcs.

Reporting all K arc pairs that yield form-closure grasps with two frictionless point fingers is equivalent to the problem of reporting all red and blue intersecting pairs of one-arc-cells, which are semi-algebraic sets. To solve this problem, we need an efficient algorithm to report all red and blue intersecting arc pairs. We can use the line sweeping algorithm by Basch et al. [7] for this. The algorithm can enumerate such K arc pairs in $O(\lambda_{t+2}(n + K) \log^3 n)$ time, where each pair of the n arcs intersects at most t times, and $\lambda_{t+2}(n + K)$ is the maximum length of an $(n + K, t + 2)$ Davenport-Schinzel sequence. $\lambda_{t+2}(n + K)$ is an almost linear function of $n + K$ for any fixed

[2] A triangle can be seen as a degenerate case of a quadrilateral, hence we consider $r(v, \phi)$ a quadrilateral.

$t + 2$. In our setting, the algebraic arcs can intersect at most t times, because the degree of the polynomials is bounded by a constant t. This provides an efficient way of computing all arc pairs with form-closure grasps, when $K = O(n^\alpha)$, $\alpha < 2$. It remains open to find truly output-sensitive computations of all such arc pairs, where the K term is additive to the other terms involving n or m.

When a planar part is bounded by arbitrary curve pieces, we believe that the projected wrench sets involve arbitrary curve pieces. Unfortunately, no efficient output-sensitive algorithm to detect the intersections of arbitrary curves has been proposed, as far as we know. If the intersection between the projected wrench sets can be efficiently computed, the approach presented in this chapter can be applied to compute all form-closure grasps of any planar curved object.

Chapter 5

Computing All Force-Closure Grasps of Polygons and Planar Semi-Algebraic Sets

Fewer fingers suffice for immobilization, if there is friction between the fingers and the part. This chapter is about the first output-sensitive computations of all force-closure grasps with frictional fingers of polygons and planar semi-algebraic sets. The beauty of the approach taken in Chapter 3 and 4 is that it can also be applied to compute all force-closure grasps of polygons and planar semi-algebraic sets. With this method, we identify all combinations of edges and concave vertices of a polygon such that two or three fingers on each of these combinations achieve force closure. The combinations that we consider include: (i) edge pairs, (ii) a concave vertex and an edge, (iii) one concave vertex and two edges, and (iv) two concave vertices and one edge. We also identify all combinations of arcs and concave vertices of a planar semi-algebraic set, such that two or three fingers on each of these combinations achieve force closure. The combinations include: (i) a concave vertex and an arc, (ii) two concave vertices and an arc, and (iii) a concave vertex and two arcs. Two frictional fingers on two arcs can also achieve force closure. However, we do not consider these cases in this chapter, because the algorithm to deal with arc-arc intersections is not efficient enough. We discuss this further in Section 5.4. Also note that three frictional fingers on three arcs can achieve force closure. This case is basically the same as that in Section 4.2.2, since it is the problem of reporting all intersecting pairs of a red semi-algebraic set and a blue semi-algebraic set. Hence this can be solved in the same time bound as in Section 4.2.2, which is $O(n^{5/2+\varepsilon} + K)$.

5.1 Preliminaries

We project the wrench sets on Γ, and report all red and blue intersections between the projected wrench sets on Γ. The projection scheme and screen Γ are as defined in Section 2.4.1. We first introduce a necessary and sufficient condition for two frictional fingers to achieve force closure.

Lemma 5.1 *Given an object with two contact wrench sets w_1 and w_2, the object is in force-closure, if and only if w_1 and w_2 have four points w_1', w_1'', w_2' and w_2'' such that the interior of a red part of $\pi(\overline{w_1'w_1''})$ intersects the interior of a blue part of $\pi(\overline{w_2'w_2''})$, or vice versa.*

The problem of enumerating all combinations of arcs/edges and concave vertices with force closure grasps can be formulated in terms of wrench sets as follows: (i) given n red and blue trapezoids and m red and blue line segments on Γ, report all combinations of trapezoids and line

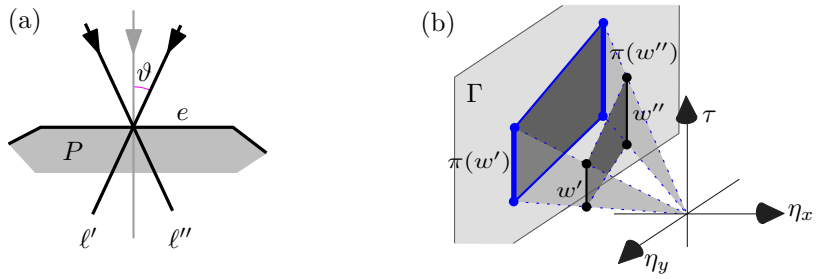


Figure 5.1: (a) The friction cone of a frictional point finger on e. (b) The projected wrench sets of the finger.

segments that define red and blue intersections in the interior; (ii) given n red and blue semi-algebraic sets and m red and blue line segments on Γ, report all combinations of semi-algebraic sets and line segments that define red and blue intersections in the interior. We investigate the shapes of wrench sets induced by a finger along an edge (Section 5.1.1), at a concave vertex ((Section 5.1.2), and along an arc (Section 5.1.3). In Section 5.1.4, we introduce the data structures and algorithms used in this chapter, to search for red and blue intersections.

5.1.1 Edge wrench sets

A frictionless point finger on an edge e induces one line of force—the normal line of e at the contact point. When a finger is frictional, it induces a set of lines of force; the set of lines of force through the contact is the friction cone. Recall that we assume Coulomb friction model (see Section 2.3). The bisector of the friction cone is the normal line at the contact point on the edge. Let ϑ denote the half-angle of the friction cone. See Figure 5.1 (a). Let ℓ' and ℓ'' be the boundary lines of the Coulomb friction cone; they make a positive or negative angle with the normal line.

A line through a point p with a direction vector $\eta = (\eta_x, \eta_y)$ determines a point $(\eta_x, \eta_y, p \times \eta)$ in wrench space. Recall that η is normalized, i.e. $|\eta| = 1$ (see Section 2.1.2). A frictional finger translating along an edge e induces a set of lines of force, and hence a set of wrench points, which forms a trapezoid on Γ. Let w' and w'' be the sets of the wrench points for ℓ' and ℓ'' translating along e respectively. Then w' and w'' are vertical line segments[1] in wrench space. We call the trapezoidal convex hull of w' and w'' in wrench space the *edge wrench set (of e)*, and let $\hat{e}$ denote it. The projection of this trapezoid $\hat{e}$ is also a trapezoid on Γ, because the projection of vertical line segments are also vertical line segments. We let $\pi(e)$ denote the projection of $\hat{e}$. See Figure 5.1 (b).

5.1.2 Concave vertex wrench sets

We now look at how the wrench set of a finger at a concave vertex looks. Let e_1 and e_2 be the incident edges of a concave vertex v, and let ℓ'_1 and ℓ''_1 be the boundary lines of the friction cone of edge e_1 at v, and ℓ'_2 and ℓ''_2 be those of e_2 at v. Without loss of generality, we assume that ℓ'_1 and ℓ''_2 are the boundary lines of the set of lines of force caused by the finger at the vertex. See Figure 5.2. A finger at a concave vertex induces a set of lines of force between two lines ℓ'_1 and ℓ''_2, thus a line segment $\overline{w'w''}$ in wrench space, where w' is the wrench point corresponding to ℓ'_1, and w'' is the wrench point corresponding to ℓ''_2. We let $s(v)$ denote the projection of $\overline{w'w''}$ on Γ.

[1] Observe that the first two components of w' and w'' are for the directions of ℓ' and ℓ''. Only the last component changes as ℓ' or ℓ'' translates along e. Hence w' and w'' are vertical line segments.

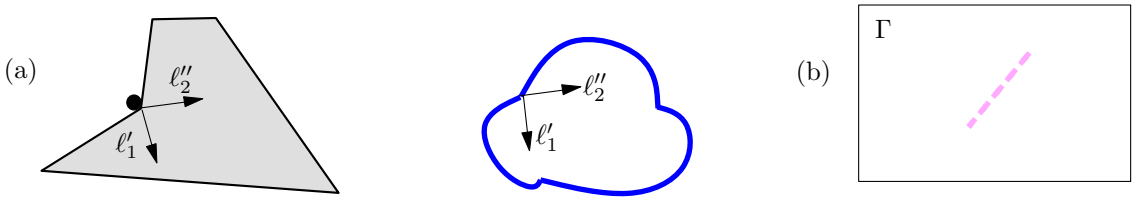


Figure 5.2: (a) A finger at a concave vertex induces a friction cone with ℓ'_1 and ℓ''_2 as boundary lines. (b) The red line segment (in dashed lines) is the wrench set induced by a finger at a concave vertex.

5.1.3 Arc wrench sets

When a frictional finger moves along an arc, it induces a set of wrench points. Let a be a boundary arc of a semi-algebraic set P'; a is a portion of an algebraic curve $\Psi(x,y)=0$ of constant degree. The interior of the semi-algebraic set P' could be $\Psi(x,y)>0$ or $\Psi(x,y)<0$. Without loss of generality, we assume that $\Psi(x,y)>0$ represents the interior of P' bounded by a locally. We let Ψ_x and Ψ_y denote $\frac{\partial\Psi(x,y)}{\partial x}$ and $\frac{\partial\Psi(x,y)}{\partial y}$ respectively. Then the wrench for the normal line at $p=(x,y)^T$ on a is $(\Psi_x,\Psi_y,x\Psi_y-y\Psi_x)$.[2] When a finger is frictional, it induces a set of lines of force in the friction cone. If we rotate the normal line around p by θ, it represents a line in the friction cone, when $\theta\leq\vartheta$.[3] In fact, any line in the friction cone can be represented this way. The wrench point for the rotated normal line by θ can be expressed as follows:

$$(\cos\theta\Psi_x-\sin\theta\Psi_y,\sin\theta\Psi_x+\cos\theta\Psi_y,x\sin\theta\Psi_x+x\cos\theta\Psi_y-y\cos\theta\Psi_x+y\sin\theta\Psi_y).$$

We assume that $\cos\theta\Psi_x-\sin\theta\Psi_y\neq 0$ and $\sin\theta\Psi_x+\cos\theta\Psi_y\neq 0$ for all values of θ. If an arc a does not satisfy this assumption, we divide a such that each portion of a satisfies the assumption. We call the wrench set of a finger translating along an arc a the *arc wrench set (of a)*, and let $\hat{a}$ denote it. In the following lemma, we show that an arc wrench set of a frictional finger forms a semi-algebraic set in wrench space.

Lemma 5.2 *The set of wrenches induced by all placements of a frictional point finger along an arc forms a semi-algebraic set.*

Proof: Let $\mathcal{N}$ be a map $\mathcal{N}:(x,y,\theta)\mapsto(\cos\theta\Psi_x-\sin\theta\Psi_y,\sin\theta\Psi_x+\cos\theta\Psi_y,x\sin\theta\Psi_x+x\cos\theta\Psi_y-y\cos\theta\Psi_x+y\sin\theta\Psi_y)$, where x and y are on a piece of an algebraic curve $\Psi(x,y)=0$, and θ in an interval of $I_\theta=[\theta_1,\theta_2]$. We show that the image of $\mathcal{N}$ is algebraic. We parametrize $\cos\theta$ and $\sin\theta$ in terms of u in a certain interval of I_u as follows:

$$\cos\theta=\frac{2u}{1+u^2}\quad\text{and}\quad\sin\theta=\frac{1-u^2}{1+u^2}.$$

Note that u is in an interval $[u_1,u_2]$. With this parametrization, $\mathcal{N}$ can be rewritten as follows.

$$\begin{aligned}\mathcal{N}:(x,y,u)\;\mapsto\;&(\tfrac{2u\Psi_x}{1+u^2}-\tfrac{(1-u^2)\Psi_y}{1+u^2},\tfrac{(1-u^2)\Psi_x}{1+u^2}+\tfrac{2u\Psi_y}{1+u^2},\\&\tfrac{x(1-u^2)\Psi_x}{1+u^2}+\tfrac{2ux\Psi_y}{1+u^2}-\tfrac{2uy\Psi_x}{1+u^2}+\tfrac{y(1-u^2)\Psi_y}{1+u^2}).\end{aligned}$$

Because all the components in the image are polynomials of x, y and u for a fixed value of u, the image is also an algebraic arc. The set of algebraic arcs for all values of u in $[u_1,u_2]$ forms a semi-algebraic set. □

[2] Note that the direction vector (Ψ_x,Ψ_y) is not normalized here, to make the formula simpler. But the arguments in this chapter still holds, with the normalization of (Ψ_x,Ψ_y).

[3] Remember that ϑ represents the half angle of the friction cone. See Section 2.3 in Chapter 2.

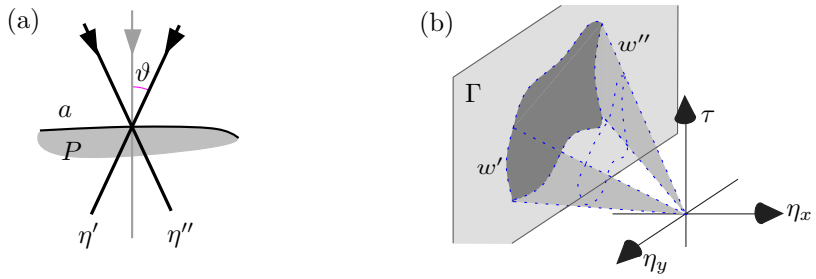


Figure 5.3: The projected wrench set of a frictional finger translating along an arc.

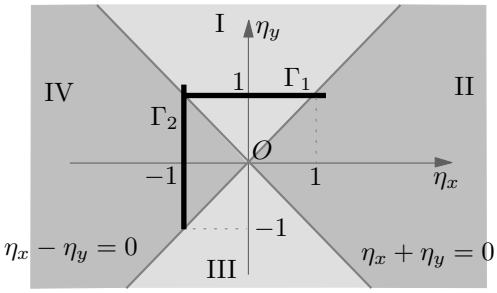


Figure 5.4: The regions I, II, III and IV, and screen Γ in wrench space, viewed from the positive τ axis.

Now we show that the projection of an arc wrench set $\hat{a}$ is also a semi-algebraic set. We let $r(a)$ denote the projection of $\hat{a}$. Abusing the notation slightly, we also call $r(a)$ the *arc wrench set of* a. See Figure 5.3.

Lemma 5.3 *The projected wrench set induced by a frictional finger translating on an arc forms a semi-algebraic set.*

Proof: The arc wrench sets are projected on different planes of Γ, depending on where they lie, just like in Section 4.1.1. See Figure 5.4. Those lying in region I and III are projected in blue and red on Γ_1, with the following description

$$(1, \frac{(1-u^2)\Psi_x + 2u\Psi_y}{2u\Psi_x - (1-u^2)\Psi_y}, \frac{x(1-u^2)\Psi_x + 2ux\Psi_y}{2u\Psi_x - (1-u^2)\Psi_y} - y),$$

and those in region II and IV are projected in red and blue on Γ_2, with the following description

$$(\frac{2u\Psi_x - (1-u^2)\Psi_y}{2u\Psi_y + (1-u^2)\Psi_x}, 1, x - \frac{2uy\Psi_x - y(1-u^2)\Psi_y}{2u\Psi_y + (1-u^2)\Psi_x}).$$

Because all the components in the image are polynomials of u, x, y, Ψ_x and Ψ_y, the image is also a piece of an algebraic curve. This means that $r(a)$ is bounded by algebraic arcs, therefore $r(a)$ is a semi-algebraic set. □

When two frictional point fingers p and p' slide along two arcs a and a' independently, they induce a set of line segments $\{\overline{ww'} | w \in \hat{a}, w' \in \hat{a'}\}$. The set of such line segments forms a semi-algebraic set bounded by $\hat{a}$, $\hat{a'}$ and a constant number of line segments. The projection of this set is also a semi-algebraic set bounded by $\pi(\hat{a})$, $\pi(\hat{a'})$ and a constant number of line segments. We let $r(a, a') := \bigcup\{\pi(\overline{ww'}) \mid w \in \hat{a}, w' \in \hat{a'}\}$. Note that an arc a could be a segment—a line segment is considered to be a special case of algebraic arcs.

5.1.4 Intersection search algorithms

In this chapter, we need to report all red and blue intersections[4] between semi-algebraic sets, line segments and triangles. More precisely, we wish to report all red and blue intersections between: (i) line segments; (ii) arcs and line segments; (iii) points and semi-algebraic sets; (iv) points and triangles; (v) real algebraic arcs.

We use two options to report all red and blue intersecting line segments: segment intersection algorithm and segment intersection search structure. All red and blue intersecting pairs of an arc and a line segment can also be reported in two ways: segment-arc intersection algorithm and segment-arc query structure. To report all points lying in a semi-algebraic set, we use a semi-algebraic range search structure. To identify all K points contained in a query triangle, we use the triangle search structure. More detailed information on the algorithms and the data structures can be found in Section 2.4.2.

Sometimes we use a variant of a triangle search structure whose preprocessing time is $O(q^{1+\varepsilon} + M\log^{\varepsilon} q)$, and the query time is $O(q/\sqrt{M}\log^3 \frac{M}{q} + k)$, where M is the space used to store the data. In this chapter, we set the space M to be $q^{4/3}$. This leads to $O(q^{4/3}\log^{\varepsilon} q)$ preprocessing time, and $O(q^{1/3}\log^3 q + k)$ query time. See Theorem 6.2 in [52].[5] Section 2.4.2 has more details about these algorithms and data structures involving arcs. All K red and blue arc-arc intersections are enumerated in a naive manner—we check every pair of arcs.

5.2 Computing all force-closure grasps of polygons

Let a polygon P have n edges and m concave vertices. In this section, we report all combinations of edges and concave vertices that yield at least one force-closure grasp with at most three frictional point fingers. The combinations are (i) two edges, (ii) one concave vertex and one edge, (iii) one concave vertex and two edges, and (iv) two concave vertices and one edge. We do not consider the problem of reporting all concave vertex pairs that yield a force-closure grasp with two frictional fingers, because this is basically the same problem as that solved in Section 3.2.1.

5.2.1 Two edges

We wish to report all edge pairs that yield a force-closure grasp with two frictional point fingers. An edge pair allows a force-closure grasp, if and only if their projected edge wrench sets on Γ have a red-blue intersection in the interior. Hence we have the problem of reporting all intersecting red and blue trapezoids. A red trapezoid intersects a blue trapezoid in the interior, if and only if either a red boundary segment intersects a blue boundary segment in the interior, or one is contained in the other.

There are n edges, each of which has at most four trapezoids on Γ. All intersecting red and blue line segments can be identified in $O(n^{4/3}\log^{1/3} n + K)$ time, using a red-blue segment intersection algorithm. To identify all red (blue) trapezoids contained in a blue (red) trapezoid, we take a vertex of each red trapezoid, and store them in a triangle search structure using $O(n^{4/3})$ space. The preprocessing time is $O(n^{4/3}\log^{\varepsilon} n)$ (ε is an arbitrarily small positive number). We triangulate a blue trapezoid, and query the structure with each of the triangle; the query time is $O(n^{1/3}\log^3 n + k)$. There are $O(n)$ queries, so the total time complexity is $O(n^{4/3}\log^3 n + K)$. Note that querying the triangle search structure will report all red trapezoids contained in a given blue trapezoid, but

[4] The red and blue intersection points must be the interior points of the red and blue sets.

[5] Matoušek's n is our q, and m is our $q^{4/3}$.

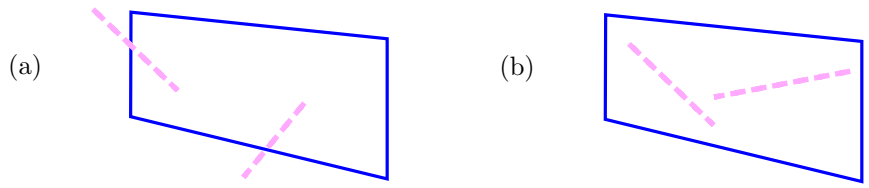


Figure 5.5: (a) The red line segments (in dashed lines) intersect the boundary segments of a blue trapezoid (in solid lines). (b) The red line segments are contained in a blue trapezoid.

also some red trapezoids intersecting the blue trapezoid. These red trapezoids intersecting the blue trapezoid will be reported at most twice, which does not affect the asymptotic time complexity. This argument will hold for the problems that we consider in the remaining sections.

Theorem 5.4 *Given a polygon with n edges, all K edge pairs that yield a force-closure grasp with two frictional point fingers can be computed in time $O(n^{4/3}\log^3 n + K)$.*

5.2.2 One concave vertex and one edge

We wish to report all pairs of one concave vertex and one edge that yield a force-closure grasp with two frictional point fingers. As a consequence of Lemma 5.1, a concave vertex and an edge allow a force-closure grasp, if and only if their wrench sets on Γ have a red-blue intersection in the interior.[6] Remember that a finger at a concave vertex induces a line segment, and a finger along an edge induces a trapezoid on Γ. Our problem is thus to report all intersecting red trapezoids and blue segments, and all intersecting red segments and blue trapezoids. Without loss of generality, we take red trapezoids and blue line segments. The case of intersecting blue trapezoids and red line segments can be treated similarly. A red trapezoid intersects a blue line segment in the interior, if and only if either a red trapezoid boundary segment intersects a blue trapezoid boundary segment in the interior, or the line segment is contained in the trapezoid. See Figure 5.5.

There are $O(n)$ blue trapezoids, and m red line segments. All intersecting red and blue line segments can be identified in $O(n^{4/3}\log^{1/3} n + K)$ time using the red-blue segment intersection algorithm. To identify all red segments in blue trapezoids, we store the midpoints of all the red line segments in a triangle search structures in $O(M)$ space. We adjust the space M, depending on the size of m compared to n. When $m \leq n^{2/3}$, we take the triangle search structure with $O(m^2 \log m)$ preprocessing time and $O(\log^3 m + k)$ query time. There are n queries, so the time complexity in this case is $O(m^2 \log m + n\log^3 m + K)$. When $n^{2/3} \leq m \leq n$, we set $M = m^{4/3}$, which gives us $O(m^{4/3}\log^{\varepsilon} m)$ preprocessing time (ε is an arbitrarily small positive number), and $O(m^{1/3}\log^3 m + k)$ query time. There are n queries, so the time complexity is $O(nm^{1/3}\log^3 m + K)$. The following table shows the data structures and algorithms used in each case and their time complexities. The number of queries and the data size[7] are asymptotic.

Range of m	Name of the algorithm	(Query,Data)	Time complexity
$m \leq n^{2/3}$	segment-segment intersection algorithm triangle search structure	 (n, m)	$O(n^{4/3}\log^{1/3} n + K)$ $O(m^2 \log m + n\log^3 m + K)$
$n^{2/3} \leq m \leq n$	segment-segment intersection algorithm triangle search structure	 (n, m)	$O(n^{4/3}\log^{1/3} n + K)$ $O(nm^{1/3}\log^3 m + K)$

[6]The intersection must have at least one point which is the interior point of the red set, and the interior point of the blue set.
[7]From here on, by data size in the table, we mean the size of the data to be stored in the data structure.

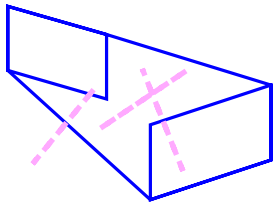


Figure 5.6: The red line segments (in dashed lines) intersect the convex hull of two blue edge wrench sets (in solid lines).

Theorem 5.5 *Given a polygon with m concave vertices and n edges, all K pairs of one concave vertex and one edge that yield a force-closure grasp with two frictional point fingers can be computed in time $O(n^{4/3}\log^3 n + K)$. More precisely, they are:*

(i) $O(n^{4/3}\log^{1/3} n + m^2 \log m + K) = O(n^{4/3}\log n + K)$*-time when* $m \leq n^{2/3}$*, and*

(ii) $O(n^{4/3}\log^{1/3} n + nm^{1/3}\log^3 m + K) = O(n^{4/3}\log^3 n + K)$ *time when* $n^{2/3} \leq m \leq n$.

5.2.3 One concave vertex and two edges

We wish to report all triples of one concave vertex and two edges that yield a force-closure grasp with three frictional point fingers. As a consequence of Lemma 5.1, our problem is to report all intersecting pairs of a red (blue) line segment and the blue (red) convex hull of two edge wrench sets on Γ. Observe that the convex hull of two edge wrench sets on Γ is a polygon of a constant complexity. A line segment intersects a polygon, if and only if either the red and blue line segments intersect each other in the interior, or the line segment is contained in the polygon. See Figure 5.6.

Without loss of generality, we take red line segments and blue trapezoids on Γ. All sets of blue line segments intersecting red edge wrench sets can be reported in a similar way. Now we construct the convex hulls of all pairs of blue trapezoids. There are $O(n^2)$ blue polygons, and $O(m)$ red line segments.

We build a segment intersection search structure on $O(m)$ red line segments in $O(m^2\log^2 m)$ time, and query with each of the $O(n^2)$ blue boundary line segments. The query time is $O(\log^4 m + k)$, so the time complexity of segment intersection search problem is $O(n^2\log^4 m + K)$. To report all segments contained in a blue polygon, we triangulate the polygon, and report all midpoints of the red segments contained in each triangle of the blue polygon. We build a triangle search structure on m midpoints of the red line segments in $O(m^2\log m)$ time, and query with each of the $O(n^2)$ blue query triangles. The query time is $O(\log^3 m + k)$, so the time complexity of the triangle intersection search problem is $O(n^2\log^3 m + K)$. The total time complexity to report all intersecting pairs of a red line segment and the convex hull of two blue edge wrench sets is thus $O(n^2\log^4 m + K)$.

Theorem 5.6 *Given a polygon with m concave vertices and n edges, all K triples of one concave vertex and two edges that yield a force-closure grasp with three frictional point fingers can be computed in time $O(n^2\log^4 m + K)$.*

5.2.4 Two concave vertices and one edge

We wish to report all triples of two concave vertices v and v' and one edge e that yield a force-closure grasp with three frictional point fingers. As a consequence of Lemma 5.1, two concave

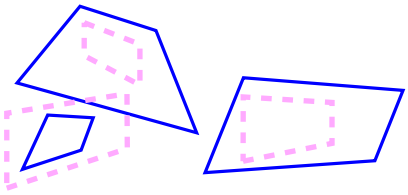


Figure 5.7: A set of red trapezoids (in dashed lines) intersecting blue quadrilaterals (in solid lines).

vertices and an edge allow a force-closure grasp, if and only if the red (blue) quadrilateral intersects the blue (red) trapezoid in the interior on Γ. An edge wrench set on Γ is a trapezoid, and the convex hull of two vertex wrench sets $s(v)$ and $s(v')$ on Γ is a quadrilateral. See Figure 5.7. Our problem is thus to report all red and blue intersecting pairs of a trapezoid and a quadrilateral. Here we focus on reporting red trapezoids intersecting blue quadrilaterals. All sets of blue trapezoids intersecting red quadrilaterals can be reported similarly. A red trapezoid intersects a blue quadrilateral in the interior, if and only if either the red and blue line segments intersect each other in the interior, or the red trapezoid is contained in the blue quadrilateral, or the blue quadrilateral is contained in the red trapezoid.

There are $O(m^2)$ blue quadrilaterals and $O(n)$ red trapezoids. For the line segment intersection search problem, we use either a red-blue segment intersection algorithm or a segment intersection search structure on n red line segments, depending on the relative size of m compared to n. To identify all blue quadrilaterals contained in a red trapezoid, and the red trapezoids in a blue quadrilateral, we use some variants of the triangle search structure on $O(m)$ midpoints of blue line segments, and on $O(n)$ points of red trapezoids with time-space trade-off. All possible combinations of intersection algorithms produce two different time complexities, depending on the relative size of m compared to n: (i) $m \leq n^{1/6}$, (ii) $n^{1/6} < m \leq n^{1/2}$, and (iii) $n^{1/2} < m \leq n$.

When $m \leq n^{1/6}$, we search for all K intersecting pairs of a red trapezoid and a blue quadrilateral in a brute-force manner. This takes $O(m^2 n)$ time.

When $n^{1/6} < m \leq n^{1/2}$, we search for all K intersecting pairs of a red trapezoid and a blue quadrilateral in total time $O(n^{4/3} \log^3 n + K)$ as follows. We first perform a red-blue line segment intersection algorithm on $O(n + m^2)$ red and blue line segments in $O(n^{4/3} \log^{1/3} n + K)$ time. To identify red trapezoids contained in blue quadrilaterals, we store the $O(n)$ red points in a triangle search structure using $M = O(n^{4/3})$ space; the preprocessing time is $O(n^{4/3} \log^{\varepsilon} n)$, and the query time is $O(n^{1/3} \log^3 n + k)$ for each of the $O(m^2)$ queries. The total time complexity for this case is $O(n^{4/3} \log^3 n + K)$. To identify blue quadrilaterals contained in red trapezoids, we store the $O(m)$ blue points in a triangle search structure with $O(m^2 \log m)$ preprocessing time. The query time is $O(\log^3 m + k)$ for each of the $O(n)$ queries. We take all pairs of blue segments among the k reported segments, and put them in set A. Then all blue quadrilaterals contained in the query red trapezoid belong to A. The total time complexity of this case is $O(n \log^3 m + K)$.

When $n^{1/2} < m \leq n$, the total time complexity is $O(n^2 \log^2 n + m^2 \log^4 n + K) = O(n^2 \log^4 n + K)$. We store the n red line segments in a segment intersection structure in $O(n^2 \log^2 n)$ time and query in $O(\log^4 n + k)$ time with each of the $O(m^2)$ query line segments. The total time complexity is $O(n^2 \log^2 n + m^2 \log^4 n + K)$. To identify all blue quadrilaterals contained in red trapezoids, we store $O(m)$ blue points in a triangle search structure with $O(m^2 \log m)$ preprocessing time. The query time is $O(\log^3 m + k)$ for each of the $O(n)$ queries. Hence the time complexity is $O(m^2 \log m + K)$. To identify all red trapezoids contained in blue quadrilaterals, we store n red points in a triangle search structure in $O(n^2 \log n)$ time. The query time is $O(\log^3 n + k)$ for each of the $O(m^2)$ queries, hence the time complexity is $O(n^2 \log n + m^2 \log^3 n + K)$. The following

table shows the data structures and algorithms used in each case and their time complexities. The number of queries and the data size are asymptotic.

Range of m	Name of the algorithm	(Query,Data)	Time complexity
$m \leq n^{1/6}$	naive segment-segment intersection search	(m, n)	$O(m^2 n)$
	naive triangle search structure	(m, n)	$O(m^2 n)$
	naive triangle search structure	(m, n)	$O(m^2 n)$
$n^{1/6} < m \leq n^{1/2}$	segment-segment intersection algorithm	(n, m)	$O(n^{4/3} \log^{1/3} n + K)$
	triangle search structure	(m, n)	$O(n^{4/3} \log^3 n + K)$
	triangle search structure	(n, m^2)	$O(n \log^3 m + K)$
$n^{1/2} \leq m \leq n$	segment-segment query structure	(m^2, n)	$O(n^2 \log^2 n + m^2 \log^4 n + K)$
	triangle search structure	(n, m)	$O(m^2 \log m + K)$
	triangle search structure	(m^2, n)	$O(n^2 \log n + m^2 \log^3 n + K)$

Theorem 5.7 *Given a polygon with m concave vertices and n edges, all K triples of two concave vertices and one edge that yield a force-closure grasp with three frictional point fingers can be computed in total time $O(n^2 \log^4 n + K)$. More precisely, the time complexities are:*

(i) $O(m^2 n)$ *when* $m \leq n^{1/6}$

(ii) $O(n^{4/3} \log^3 n + K)$ *when* $n^{1/6} < m \leq n^{1/2}$, *and*

(iii) $O(n^2 \log^2 n + m^2 \log^4 n + K) = O(n^2 \log^4 n + K)$ *when* $n^{1/2} < m \leq n$.

5.3 Computing all force-closure grasps of planar semi-algebraic sets

Let P' denote a planar semi-algebraic set with n boundary arcs and m concave vertices. The boundary arcs are all real algebraic arcs. In this section, we wish to report all force-closure grasps on the following combinations: (i) pairs of a concave vertex and an arc, (ii) triples of two concave vertices and an arc, and (iii) triples of a concave vertex and two arcs.

5.3.1 One concave vertex and one arc

We wish to report all pairs of a concave vertex and an arc that yield a force-closure grasp with two frictional point fingers. A frictional finger at a concave vertex v induces a line segment $s(v)$ on Γ, and a finger on an arc a induces a semi-algebraic set $r(a)$. As a consequence of Lemma 5.1, a concave vertex and an arc allow a force-closure grasp, if and only if the red (blue) line segment $s(v)$ intersects the blue (red) semi-algebraic set $r(a)$ in the interior. Here we focus on the case of red line segments intersecting blue semi-algebraic sets in the interior. All sets of blue line segments intersecting red semi-algebraic sets can be reported in the same way. A red line segment intersects a blue semi-algebraic set in the interior, if and only if one of the following holds: (i) the red line segment intersects the blue boundary arcs[8] in the interior, or (ii) the red segment lies in the blue semi-algebraic set.

There are $O(n)$ blue semi-algebraic sets on Γ, thus $O(n)$ arcs on Γ. There are $O(m)$ red line segments from the concave vertices, thus $O(m)$ red midpoints. When m is small relative to n, more precisely, when $m \leq n^{1/2}$, we take a brute-force approach, which takes $O(nm)$ time.

When $n^{1/2} < m \leq n$, we use the segment-arc intersection algorithm to search for all red and blue segment-arc intersections. This runs in $O(n^{3/2+\varepsilon}+K)$ time. To report all red segments lying in

[8]The boundary of $r(a)$ may have line segments. Since a line is a degenerate case of an algebraic curve, this case can handle segment-segment intersections as well. In addition, this does not affect the time complexity.

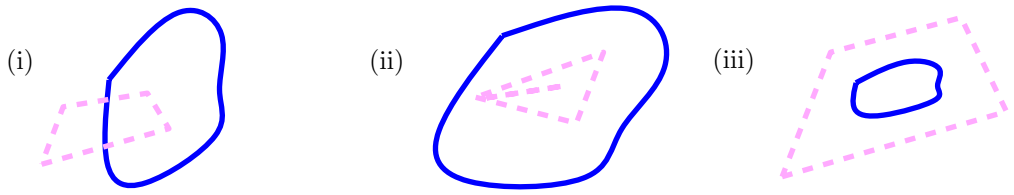


Figure 5.8: A set of red quadrilaterals (in dashed lines) intersecting blue semi-algebraic sets (in solid lines).

blue semi-algebraic sets (case (ii)), we store $O(m)$ red midpoints in a semi-algebraic range search structure in $O(m \log m)$ time. We query the structure with each of the $O(n)$ query semi-algebraic sets, which takes $O(m^{1/2+\varepsilon} + k)$ time. The time complexity for case (ii) is $O(nm^{1/2+\varepsilon} + K)$.[9]

The following table shows the data structures and algorithms used in each case and their time complexities. As in the previous section, the number of queries and the data size are asymptotic.

Range of m	Name of the algorithm	(Query,Data)	Time complexity
$m \leq n^{1/2}$	brute-force approach	(m, n)	$O(nm)$
$n^{1/2} < m \leq n$	segment-arc intersection algorithm	(m, n)	$O(n^{3/2+\varepsilon} + K)$
	semi-algebraic range search structure	(n, m)	$O(nm^{1/2+\varepsilon} + K)$

Theorem 5.8 *Given a planar semi-algebraic set with n boundary arcs of constant degree and m concave vertices, all K pairs of a concave vertex and an arc that yield force-closure grasps with two frictional point fingers can be enumerated in total time $O(n^{3/2+\varepsilon} + K)$. More precisely the time complexities are:*

1. *$O(nm)$ when $m \leq n^{1/2}$, and*
2. *$O(n^{3/2+\varepsilon} + K)$ when $n^{1/2} < m \leq n$.*

5.3.2 Two concave vertices and one arc

We wish to report all triples of two concave vertices v and v', and one arc a that yield a force-closure grasp with three frictional point fingers. Three fingers at v, v' and a induce two line segments $s(v)$, $s(v')$ and a semi-algebraic set $r(a)$ on Γ. As a result of Lemma 5.1, two concave vertices and one arc allow a force-closure grasp, if and only if the red quadrilateral[10] intersects the blue semi-algebraic set in the interior on Γ. Here we focus on the case of red quadrilaterals and blue semi-algebraic sets. All intersecting pairs of a blue quadrilateral and a red semi-algebraic set can be reported similarly. A red quadrilateral intersects a blue semi-algebraic set, if and only if one of the following three holds: (i) red line segment intersects the blue boundary arc in the interior, or (ii) a red point is in the blue semi-algebraic set, or (iii) a blue endpoint is inside the red quadrilateral. See Figure 5.8.

There are $O(n)$ blue semi-algebraic sets, thus $O(n)$ blue boundary arcs and representative points—we pick an endpoint from each of the semi-algebraic sets. There are $O(m^2)$ red quadrilaterals and $O(m)$ midpoints of the red segments. When m is small relative to n, more precisely, when $m \leq n^{1/2}$, we use brute-force algorithm, which takes $O(nm^2)$ time.

[9] Note that this query structure finds all the red line segments contained in a query semi-algebraic set, as well as some segments intersecting the query semi-algebraic set. The latter will be reported once again. In total the desired output can be reported at most twice, which does not affect the asymptotic time complexity. In the following sections as well, the output will be reported at most three times.

[10] The convex hull of $s(v)$ and $s(v')$ is in general, a quadrilateral. It can be a triangle, which we consider as a degenerate quadrilateral.

Figure 5.9: A set of red line segments $s(v)$ (in dashed lines) intersecting $r(a, a')$ (in solid lines).

When $n^{1/2} < m \leq n$, we do the following. We first store $O(n)$ blue arcs in a segment-arc query structure in $O(n^{2+\varepsilon})$ time. Each of the $O(m^2)$ queries takes $O(\log n + k)$ time, therefore, the total time complexity for case (i) is $O(n^{2+\varepsilon} + K)$. Then we store $O(m)$ red points in a semi-algebraic range search structure in $O(m \log m)$ time. With each of the $O(n)$ semi-algebraic sets, we query the structure in $O(m^{1/2+\varepsilon} + k)$ time. The total time complexity for case (ii) is thus $O(nm^{1/2+\varepsilon} + K)$. Finally we report all blue semi-algebraic sets contained in red quadrilaterals (case (iii)) as follows. We store $O(n)$ blue points in a triangle search structure in $O(n^2 \log n)$ time. We triangulate all red quadrilaterals; each quadrilateral has two triangles. We query the structure with each of the $O(m^2)$ triangles in $O(\log^3 n + k)$ time. The total time complexity of case (iii) is thus $O(n^2 \log n + m^2 \log^3 n + K)$.

The following table shows the algorithms used in each case and their time complexities. The number of queries and the data size are asymptotic.

Range of m	Name of the algorithm	(Query,Data)	Time complexity
$m \leq n^{1/2}$	brute-force manner	(n, m^2)	$O(nm^2)$
$n^{1/2} < m \leq n$	segment-arc query structure	(m, n)	$O(n^{2+\varepsilon} + K)$
	semi-algebraic range search structure	(n, m)	$O(nm^{1/2+\varepsilon} + K)$
	triangle search structure	(n, m^2)	$O(n^2 \log^3 n + K)$

Theorem 5.9 *Given a planar semi-algebraic set with n boundary arcs and m concave vertices, all K triples of two concave vertices and an arc that yield force-closure grasps with three frictional point fingers can be enumerated in total time $O(n^{2+\varepsilon} + K)$. More precisely the time complexities are:*

(i) $O(nm^2)$ *when* $m \leq n^{1/2}$, *and*

(ii) $O(n^{2+\varepsilon} + K)$ *when* $n^{1/2} < m \leq n$.

5.3.3 One concave vertex and two arcs

We wish to report all triples of a concave vertex v and two arcs a and a' that yield a force-closure grasp with three frictional point fingers. A finger at v induces a line segment $s(v)$, and two fingers sliding on a and a' independently induce a semi-algebraic set $r(a, a')$ on Γ—see Section 5.1.3. As a consequence of Lemma 5.1, a concave vertex and two arcs allow a force-closure grasp, if and only if the red line segment $s(v)$ intersects $r(a, a')$ in the interior on Γ. Here we focus on the case of red line segments and blue semi-algebraic sets. All sets of blue segments intersecting red semi-algebraic sets can be reported similarly. A red line segment intersects a blue semi-algebraic set, if and only if one of the following three holds: (i) the red and blue boundary line segments intersect each other in the interior, or (ii) the red line segment intersects the blue arcs in the interior, or (iii) the red midpoint is in the blue semi-algebraic set. See Figure 5.9.

There are $O(n)$ blue arcs and endpoints from $O(n)$ arc wrench sets, and $O(n^2)$ blue line segments and semi-algebraic sets from all pairs of arc wrench sets. There are also $O(m)$ red line

segments and midpoints. When m is small relative to n, more precisely, when $m \leq n^{1/2}$, we use brute-force algorithm, which takes $O(nm)$ time.

When $n^{1/2} < m \leq n$, we do the following. We first build a segment-segment query structure on $O(m)$ red line segments to report all red and blue intersections between line segments (case (i)). The preprocessing time is $O(m^2 \log^2 m)$, and the query time is $O(\log^4 m + k)$ for each of the $O(n^2)$ blue query line segments. The total time complexity for case (i) is $O(n^2 \log^4 m + K)$. We use the segment-arc intersection algorithm to report all intersecting pairs of a red line segment and a blue arc. This runs in $O(n^{3/2+\varepsilon} + K)$ time. Finally we report all red segments contained in blue semi-algebraic sets (case (iii)) as follows. We store $O(m)$ red midpoints in a semi-algebraic range search structure in $O(m \log m)$ time, and query it with each of the $O(n^2)$ semi-algebraic sets in $O(m^{1/2+\varepsilon} + k)$ time. The time complexity of case (iii) is thus $O(n^2 m^{1/2+\varepsilon} + K)$. The following table shows the algorithms used in each case and their time complexities. The number of queries and the data size are asymptotic.

Range of m	Name of the algorithm	(Query,Data)	Time complexity
$m \leq n^{1/2}$	brute-force approach	(m, n)	$O(nm)$
$n^{1/2} < m \leq n$	segment-segment query structure	(n^2, m)	$O(n^2 \log^4 m + K)$
	segment-arc intersection algorithm	$(0, n + m)$	$O(n^{3/2+\varepsilon} + K)$
	semi-algebraic range search structure	(n, m)	$O(n^2 m^{1/2+\varepsilon} + K)$

Theorem 5.10 *Given a planar semi-algebraic set with n boundary arcs and m concave vertices, all K triples of one concave vertex and two arcs that yield force-closure grasps with three frictional point fingers can be enumerated in total time $O(n^2 m^{1/2+\varepsilon} + K)$.*

5.4 Conclusion

In this chapter, we proposed efficient output-sensitive algorithms to enumerate all combinations of edges and concave vertices of polygons, and all combinations of arcs and concave vertices of semi-algebraic sets that yield force-closure grasps. We extended the methods in Chapter 3 and and 4 to compute all force-closure grasps. We investigated the shapes of the wrench sets induced by a finger, when there is friction between the part and the finger.

Note that two frictional fingers on two arcs, and three frictional fingers on three arcs of a semi-algebraic set can achieve force closure. Reporting all K arc pairs that yield force-closure grasps with two frictional fingers is equivalent to the problem of reporting all red and blue intersecting pairs of arc wrench sets, which are semi-algebraic sets. As far as we know, one of the most efficient algorithms that we could use to solve this problem is the line sweeping algorithm by Basch et al. [7]. The time complexity of this algorithm is $O(\lambda_{t+2}(n + K) \log^3 n)$, where $\lambda_{t+2}(n + K)$ is an almost linear function of $n + K$ in our setting. More detailed information can be found in Section 4.4. This algorithm, however, works well only when K is sufficiently small. It remains open to find an efficient output-sensitive algorithm for computing all arc pairs with force-closure grasps, where the K term is additive to the other terms involving n or m.

Chapter 6

Computing All Second-Order Immobility Grasps of Polygons

Many planar objects can be immobilized with three frictionless point fingers. This chapter[1] is about constructing all second-order immobility grasps of polygons with two and three frictionless point fingers. We call a configuration of frictionless point fingers that achieves second-order immobility a *second-order immobility grasp*. Polygons except some with parallel edges can be immobilized with three fingers—four are necessary to immobilize the exceptions.

Czyzowicz, Stojmenovic and Urrutia [33] showed that there exists at least one second-order immobility grasp for polygons without parallel edges. They also provided a necessary and sufficient condition for three normal lines of a polygon to achieve second-order immobility. See Section 2.2. We need at least three normal lines to immobilize a planar object, because the directions must positively span the object plane.

A finger at a concave vertex induces at least two normal lines. Taking advantage of concave vertices, we can reduce the number of fingers to achieve second-order immobility. In this chapter, we propose output-sensitve algorithms to report all edge triples and all pairs of a concave vertex and an edge that allow a second-order immobility with three and two point fingers, respectively. We first introduce notations and CSU condition in Section 6.1. Then we propose efficient output-sensitive algorithm to enumerate all edge triples (Section 6.2) and all pairs of an edge and a concave vertex (Section 6.3) that allow a second-order immobility grasp with three and two frictionless point fingers respectively. We discuss extensions and related issues in Section 6.4.

6.1 Preliminaries

In this section, we introduce some notations and definitions. Let the edges of the simple polygon P be oriented counter-clockwise around P, that is, P lies locally to the left of each edge. See Figure 6.1 (a). We denote the line orthogonal to an edge e through the start and end point of e by $s_0(e)$ and $s_1(e)$, respectively. Let $\hat{s}(e)$ be the relatively open infinite slab bounded by $s_0(e)$ and $s_1(e)$, that is, the union of all lines that are orthogonal to and intersect the interior of e (see Figure 6.4). Let $l(e)$ be the supporting line of e, and let $H(e)$ be the open half plane bounded by $l(e)$ lying locally to the left of e, that is, locally containing P (see Figure 6.2). When the

[1]This chapter is based on "On computing all immobilizing grasps of a simple polygon with few contacts" [23] by J.-S. Cheong, Herman Haverkort and Frank van der Stappen in ISAAC (2003), and "On computing all immobilizing grasps of a simple polygon with few contacts" [24] by J.-S. Cheong, Herman Haverkort and Frank van der Stappen in Algorithmica (2006).

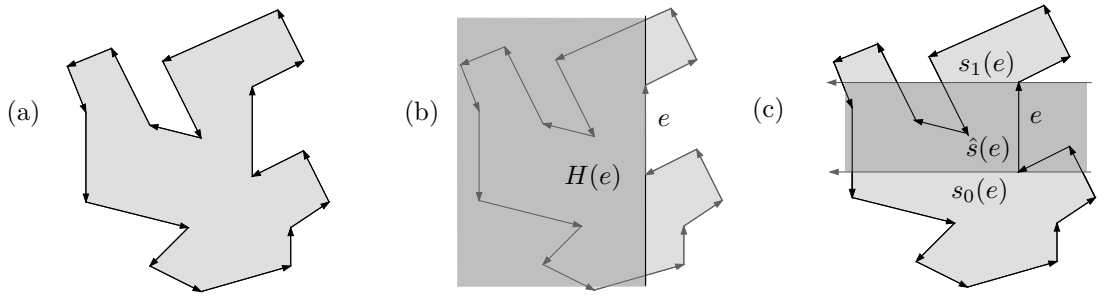


Figure 6.1: (a) Edges of P have directions, such that the interior of P lies on the left. (b) $H(e)$ is the open half plane bounded by $l(e)$ locally containing P. (c) $\hat{s}(e)$ is the union of all normal lines of the interior of e.

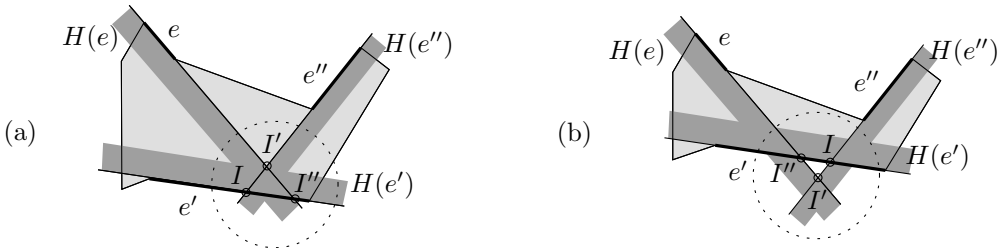


Figure 6.2: The edges e, e', e'' in (a) are a triangular triple, while those in (b) are not.

intersection of $H(e), H(e')$ and $H(e'')$ forms a (finite) triangle, then e, e', and e'' are said to be a *triangular triple*. (Compare Figure 6.2 (a) and (b).)

To identify all edge triples that allow second-order immobility grasps, we use the necessary and sufficient condition provided by Czyzowicz, Stojmenovic and Urrutia [33]. We will restate the CSU condition below again. The proof of this lemma can be found in Section 2.2.

Lemma 6.1 (Czyzowicz et al. [33]) *Three frictionless point fingers on the interior of three edges e, e', and e'' immobilize a polygon, if and only if the followings are true:*

(i) $\hat{s}(e) \cap \hat{s}(e') \cap \hat{s}(e'') \neq \emptyset$ *(common normal intersection condition),*

(ii) $H(e) \cap H(e') \cap H(e'')$ *is a triangle (triangular triple condition).*

6.2 Computing all second-order immobility grasps with three fingers

To find all edge triples that allow second-order immobility grasps, we take a similar approach as in [82]. We find all the edge triples that have a common normal intersection; among these, triangular triples will be filtered out. The sketch of the global approach is as follows. For each edge e of P, we build a data structure. It will be queried with each of the remaining $n-1$ edges e', to report all edges e'' such that the triple (e, e', e'') satisfies the conditions of Lemma 6.1.

From now on, we focus on building and searching the data structure for a fixed edge e. First we show the necessary and sufficient condition for the common normal intersection condition. We choose a coordinate system such that $l(e)$ is the y-axis, oriented in upward direction. We divide the remaining edges into two groups L(ower) and U(pper); when an edge forms an angle between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$ with the positive x-axis, it is in L, otherwise it is in U (see Figure 6.3 (b) and (c)). We omit all vertical edges, i.e. all edges parallel to e from L and U, since they could never make a triangular triple with e and a third edge. For $i \in \{0, 1\}$, we define l'_i and r'_i as the x-coordinates of

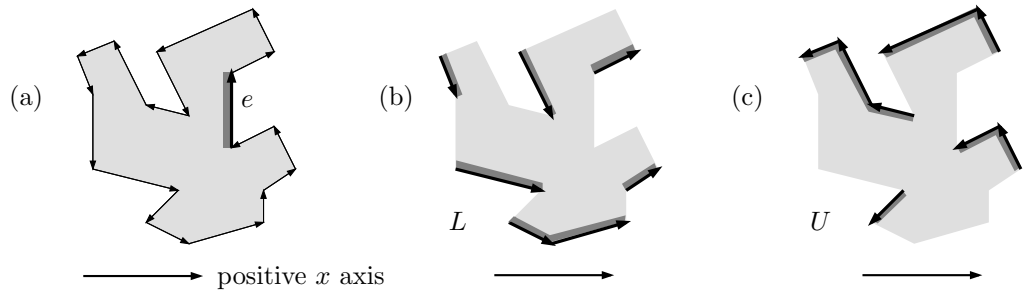


Figure 6.3: (a) A polygon with directed edges, oriented such that e is on the y-axis, pointing upward. (b) The edges in L, and (c) in U.

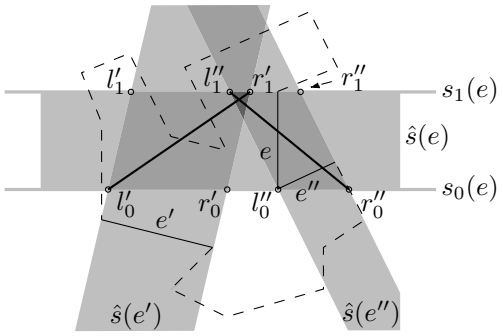


Figure 6.4: Notation for Lemma 6.2.

the left and right intersection points of $s_i(e)$ and the slab boundaries of $\hat{s}(e')$. We define l''_0, r''_0, l''_1 and r''_1 for edge e'' likewise—see Figure 6.4. The following is a necessary and sufficient condition for three edges to have a non-empty common normal intersection region.

Lemma 6.2 *Two slabs $\hat{s}(e')$ and $\hat{s}(e'')$ have a common intersection with $\hat{s}(e)$ if and only if one of the following is true:*

(i) $l'_0 < r''_0 \wedge l''_1 < r'_1$

(ii) $l''_0 < r'_0 \wedge l'_1 < r''_1$

Proof: We will first prove that if one of the conditions (i) or (ii) is met, there is a common intersection. After that we will prove that if neither of the conditions is fulfilled, there cannot be a common intersection.

The "if" direction: the two cases are identical except for e' and e'' switching roles, so without loss of generality, we restrict ourselves to the first case. Condition (i) implies that the line segments $\overline{l'_0 r'_1}$ and $\overline{r''_0 l''_1}$ intersect (see Figure 6.4 for an example). The first line segment lies completely inside $\hat{s}(e)$ and $\hat{s}(e')$; the second lies completely inside $\hat{s}(e)$ and $\hat{s}(e'')$. Hence, their intersection lies in all three slabs, which means that the intersection of $\hat{s}(e)$, $\hat{s}(e')$ and $\hat{s}(e'')$ is not empty.

The "only-if" direction: suppose neither condition (i) nor condition (ii) is true, i.e. the following *is* true:

$$(l'_0 \geq r''_0 \vee l''_1 \geq r'_1) \wedge (l''_0 \geq r'_0 \vee l'_1 \geq r''_1)$$

Because by definition, $l'_0 < r'_0$ and $l''_0 < r''_0$, we cannot simultaneously have $l'_0 \geq r''_0$ and $l''_0 \geq r'_0$. Likewise, we cannot simultaneously have $l'_1 \geq r''_1$ and $l''_1 \geq r'_1$. It follows that the proposition above is equivalent to:

$$(l'_0 \geq r''_0 \wedge l'_1 \geq r''_1) \vee (l''_0 \geq r'_0 \wedge l''_1 \geq r'_1)$$

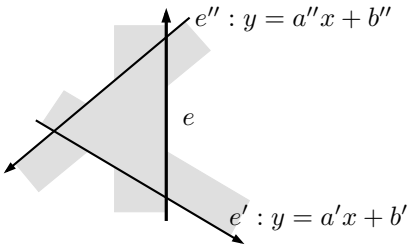


Figure 6.5: An illustration of Lemma 6.3.

In other words, the left boundary of one slab of $\hat{s}(e')$ and $\hat{s}(e'')$ lies to the right of the right boundary of the other slab, and the situation is the same both at the intersection with the lower boundary of $\hat{s}(e)$, and at the intersection with the upper boundary of $\hat{s}(e)$. It follows that the intersections of $\hat{s}(e')$ and $\hat{s}(e'')$ with $\hat{s}(e)$ are disjoint.

So if there is a common intersection, at least one of the conditions must be fulfilled, and if at least one of the conditions is fulfilled, there must be a common intersection. □

Now we move on to the condition equivalent to the triangular triple condition. Suppose $l(e')$ is the line defined by $y = a'x + b'$, and $l(e'')$ is the line defined by $y = a''x + b''$.

Lemma 6.3 *$H(e) \cap H(e') \cap H(e'')$ is a triangle if and only if one of the following is true:*

(i) $a' < a'' \wedge b' < b'' \wedge e' \in L \wedge e'' \in U$

(ii) $a'' < a' \wedge b'' < b' \wedge e' \in U \wedge e'' \in L$

Proof: Let I' be the intersection $(0, b'')$ of $l(e)$ and $l(e'')$; let I'' be the intersection $(0, b')$ of $l(e)$ and $l(e')$, and let I be the intersection (I_x, I_y) of $l(e')$ and $l(e'')$, where $I_x = (b'' - b')/(a' - a'')$ and $I_y = a'I_x + b' = a''I_x + b''$ (see Figure 6.2). Observe that $H(e) \cap H(e') \cap H(e'')$ is a triangle if and only if $I \in H(e)$, $I' \in H(e')$ and $I'' \in H(e'')$.

We will first prove that if one of the conditions (i) or (ii) holds, $H(e) \cap H(e') \cap H(e'')$ is a triangle, and then, that if the latter is a triangle, one of the conditions must be fulfilled.

The "if" direction: the two cases are identical except for e' and e'' switching roles, so without loss of generality, we restrict ourselves to the first case. Condition (i) (as well as (ii)) implies that $I_x < 0$, so $I \in H(e)$. Furthermore, $e' \in L$ means that $H(e')$ is the half plane above $l(e')$; since $b' < b''$, we have that $I' = (0, b'')$ lies above $l(e')$, and thus, inside $H(e')$. Likewise, from $e'' \in U$ and $b' < b''$ it follows that $I'' \in H(e'')$. Hence, $H(e) \cap H(e') \cap H(e'')$ is a triangle.

The "only-if" direction: suppose $H(e) \cap H(e') \cap H(e'')$ is a triangle, then $I = l(e') \cap l(e'') \in H(e)$, that is: $I_x = (b'' - b')/(a' - a'') < 0$. This implies that one of the following is true:

(1) $a' < a'' \wedge b' < b''$

(2) $a'' < a' \wedge b'' < b'$

In the first case, $I' = (0, b'')$ lies above $I'' = (0, b')$, so the triangle formed by the $l(e')$, $l(e'')$, and the y-axis $l(e)$, is bounded by $l(e')$ from below and by $l(e'')$ from above. From the fact that this triangle lies inside $H(e')$ and $H(e'')$, it follows that $e' \in L$ and $e'' \in U$, fulfilling condition (i) of the Lemma. In the same manner, we can derive that the second case implies that $e' \in U$ and $e'' \in L$, fulfilling condition (ii) of the Lemma. □

From Lemmas 6.1, 6.2 and 6.3 it follows that e, e' and e'' allow a three-point immobility grasp if and only if one of the following conditions is satisfied:

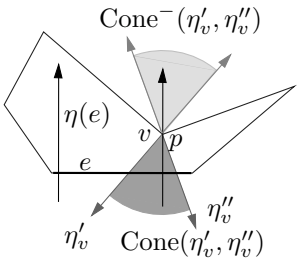


Figure 6.6: An illustration of Lemma 6.5.

(i) $l''_1 < r'_1 \wedge r''_0 > l'_0 \wedge a'' > a' \wedge b'' > b' \wedge e' \in L \wedge e'' \in U$

(ii) $l''_1 < r'_1 \wedge r''_0 > l'_0 \wedge a'' < a' \wedge b'' < b' \wedge e' \in U \wedge e'' \in L$

(iii) $l'_1 < r''_1 \wedge r'_0 > l''_0 \wedge a'' > a' \wedge b'' > b' \wedge e' \in L \wedge e'' \in U$

(iv) $l'_1 < r''_1 \wedge r'_0 > l''_0 \wedge a'' < a' \wedge b'' < b' \wedge e' \in U \wedge e'' \in L$

Since the roles of e' and e'' are interchangeable, we only need to search for triples satisfying condition (i) or (ii). We can do this with two four-dimensional orthogonal range trees [9] as follows. In the first tree, store every edge $e'' \in U$ as a four-dimensional point (l''_1, r''_0, a'', b''). Query this tree with every edge $e' \in L$ for all points in $\langle -\infty, r'_1\rangle \times \langle l'_0, \infty\rangle \times \langle a', \infty\rangle \times \langle b', \infty\rangle$. In the second tree, store every edge $e'' \in L$ as a four-dimensional point (l''_1, r''_0, a'', b''). Query this tree with every edge $e' \in U$ for all points in $\langle -\infty, r'_1\rangle \times \langle l'_0, \infty\rangle \times \langle -\infty, a'\rangle \times \langle -\infty, b'\rangle$. Every edge e'' reported forms a triple with e and e' such that three point fingers on e, e' and e'' will immobilize the polygon.

Now we analyze the time complexity of this algorithm. A four-dimensional orthogonal range tree can be built in $O(n \log^3 n)$ time using $O(n \log^3 n)$ space, and can be queried in $O(\log^4 n + k)$ time (see Chapter 5.4 in [9]). This can be improved to $O(\log^3 n + k)$ query time, with the same building time, using fractional cascading (see Chapter 5.6 in [9]).

We query each tree with $O(n)$ edges e', for a total building and query time of $O(n \log^3 n + k)$ per tree, and we do this for every edge e, so that the complete search takes $O(n^2 \log^3 n + k)$ time.

Theorem 6.4 *Given a polygon with n edges, all K edge triples (e, e', e'') such that the polygon can be immobilized by three frictionless point fingers on the interiors of e, e' and e'', can be computed in time $O(n^2 \log^3 n + K)$.*

6.3 Computing all second-order immobility grasps with two fingers

If we exploit concave vertices, two fingers can achieve second-order immobility for a simple polygon: one at a concave vertex v and the other in the interior of an edge e. When a polygon has n edges and m concave vertices, all such pairs can be reported in time $O(mn)$ by simply checking all vertex-edge pairs. Obviously we want a more efficient algorithm. We could adapt the algorithm in Section 6.2 to report only triples of edges where two edges are in fact reduced to points that coincide on a concave vertex. But this would cost even more than $O(mn)$ time. Therefore we develop a specialized algorithm based on Lemma 6.5, which we will introduce later. This lemma is equivalent to the condition described in Lemma 6.1.

First, we introduce some notations used in this section. Let e' and e'' be the edges incident to v. Let η'_v be the inward normal to e', and let η''_v be the inward normal to e''. Let $\mathrm{Cone}(\eta'_v, \eta''_v)$ be $\{\lambda'\eta'_v + \lambda''\eta''_v | \lambda', \lambda'' > 0\}$, that is, the set of all positive linear combinations of η'_v and η''_v. In the

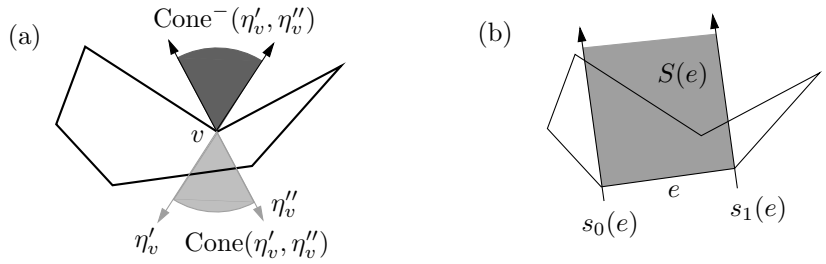


Figure 6.7: (a) Notations of normals of edges and concave vertices. (b) Vertex v is in the simplex $S(e)$ of edge e.

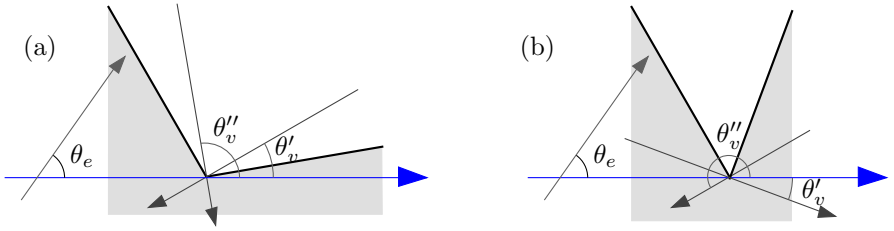


Figure 6.8: An illustration of the observation of angles.

same way, let Cone$^-(\eta'_v, \eta''_v)$ be the set of all positive linear combinations of $-\eta'_v$ and $-\eta''_v$ (see Figure 6.7(a)). For each edge e, let η_e be the inward normal to e, and let the open simplex $S(e)$ be $\hat{s}(e) \cap H(e)$ (see Figure 6.7 (b)).

Lemma 6.5 *Placing two point fingers at a concave vertex v and on an edge e immobilizes a polygon if and only if:*

(i) $\eta_e \in$ *Cone*$^-(\eta'_v, \eta''_v)$*, and*

(ii) $p \in S(e)$.

Proof: Let e' and e'' be the adjacent edges to v, shrunk onto the vertex v, so that $\hat{s}(e')$ is the line orthogonal to e' through v, and $\hat{s}(e'')$ is the line orthogonal to e'' through v. We will first show that any three edges e, e' and e'' satisfying the above statement must satisfy Lemma 6.1. Since $p = \hat{s}(e') \cap \hat{s}(e'') \in S(e) \subset \hat{s}(e)$, we must have $\hat{s}(e) \cap \hat{s}(e') \cap \hat{s}(e'') \neq \emptyset$. Furthermore, since $p \in S(e) \subset H(e)$, the intersection $H(e) \cap H(e') \cap H(e'') \neq \emptyset$. In fact, $H(e) \cap H(e') \cap H(e'')$ is a triangle, because $\eta_e \in$ Cone$^-(\eta'_v, \eta''_v)$, i.e., the normals of e, e' and e'' span the plane positively.

Let us now show that any three edges e, e' and e'', such that e' and e'' are both adjacent to and shrunk onto a common concave vertex v, and e, e' and e'' satisfy Lemma 6.1, must also satisfy the two conditions above. The common normal intersection condition assures that $p \in \hat{s}(e)$. The triangular triple condition, that $H(e) \cap H(e') \cap H(e'') \neq \emptyset$, implies that the normals of the edges span the plane positively, which proves that $\eta_e \in$ Cone$^-(\eta'_v, \eta''_v)$. □

For any edge e and any concave vertex v, let θ_e, θ'_v and θ''_v be the angles that η_e, $-\eta'_v$ and $-\eta''_v$, respectively, make with the positive x-axis. Let θ'_v be the smaller angle of θ'_v and θ''_v, i.e. $\theta'_v < \theta''_v$. Observe that $\eta_e \in$ Cone$^-(\eta'_v, \eta''_v)$ if and only if one of the following is true:

(i) $\theta''_v - \theta'_v < \pi \wedge \theta_e \in \langle \theta'_v, \theta''_v \rangle$

(ii) $\theta''_v - \theta'_v > \pi \wedge \theta_e \in [-\pi, \theta'_v\rangle \cup \langle \theta''_v, \pi]$.

For a given edge e, we find all concave vertices that have a two-point immobilizing grasp with e. For this, we store the concave vertices in a data structure that stores pairs of the form (I_v, p),

where I_v is a one-dimensional interval ($\langle\theta'_v, \theta''_v\rangle$) and v is a point in the plane. Each vertex v with $\theta''_v - \theta'_v < \pi$ is stored once, as a pair $(\langle\theta'_v, \theta''_v\rangle, p)$. Each vertex v with $\theta''_v - \theta'_v > \pi$ is stored twice: once as a pair $(\langle-\infty, \theta'_v\rangle, p)$ and once as $(\langle\theta''_v, \infty\rangle, p)$. We query this data structure with each edge e of P, to report all vertices v stored as a pair (I_v, p) such that $\theta_e \in I_v$ and $p \in S(e)$.

The data structure we use is a two-level data structure. The top level is a segment tree [9] on the intervals I_v. Let X be the set of all begin and end points of intervals I_v to be stored in the tree, in order of increasing value. A segment tree is a balanced binary tree on the intervals between consecutive values from X: each leaf is associated with one such interval. Each internal node p is associated with an interval $I(p)$, which is the union of the intervals of its descendants. With each node p, we associate a data structure $\mathcal{T}(p)$ that stores all pairs (I_v, v) such that I_p contains $I(p)$ but not $I(\text{parent}(p))$. In our case, the data structures $\mathcal{T}(v)$ will be triangle search structures on the points v in the pairs (I_v, v). Again, we use the triangle search structure by Matoušek [52], using $O(m^\alpha)$ space to store $O(m)$ points, for a certain constant d. We will explain later how α is chosen, but in any case, we will choose it such that $1 < \alpha \leq 2$.

Let us first analyse the time needed to construct the data structure. A simplex range searching structure can be built in time $O(m^d \log^\delta m)$, where m is the number of points stored, m^d is the amount of storage used for them, and δ is any small positive constant [52]. A node v at depth i in a segment tree stores intervals I_v that completely contain $I(v)$, but not $I(\text{parent}(v))$, which means that all intervals I_v stored in v must have an endpoint in $I(\text{brother}(v))$. Since the segment tree is balanced, there are at most $2m/2^i$ such intervals. Thus, at each depth i in the segment tree, we have at most 2^i nodes storing at most $2m/2^i$ intervals each. The time needed to build the complete tree thus becomes $O(m \log m)$ (for the segment tree itself) plus, for the associated triangle search structures, $O(\sum_{i=0}^{\log m} 2^i O((m/2^i)^d \log^\delta(m/2^i)))$. Since $d > 1$, the larger triangle search structures dominate, making a total construction time of $O(m^d \log^\delta m)$.

A query with an edge e for matching v in this multi-level data structure proceeds as follows. We walk down the segment tree, finding all $O(\log m)$ nodes v (one at each depth) such that $I(v)$ contains θ_e. Together, these nodes contain all pairs (I_v, p) such that I_v contains θ_e. For each of these nodes, we search the associated triangle search structure, and report the answers. The query time in a simplex range searching structure on m points with m^d storage is $O(m(\log^3 m^{d-1})/\sqrt{m^d} + k)$. The total time for a query in our data structure is therefore

$$O\left(\sum_{i=0}^{\lceil \log m \rceil}\left(1 + O\left(\frac{m}{2^i}\frac{\log^3\left(\frac{m}{2^i}\right)^{d-1}}{\sqrt{\left(\frac{m}{2^i}\right)^d}}\right)\right) + k\right)$$

If $d = 2$, this is $O(\log^4 m + k)$, otherwise it is $O(m(\log^3 m)/\sqrt{m^d} + k)$.

Since we perform n queries, the time for building and querying the data structure adds up to $O(m^d \log^\delta m + n \log^4 m + K)$ (for $d = 2$) or $O(m^d \log^\delta m + nm(\log^3 m)/\sqrt{m^d} + K)$ (for $1 < d < 2$).

Let's now choose d. If $m/\log^{3/2} m \leq \sqrt{n}$, we choose $d = 2$, and the algorithm runs in $O(n \log^4 m + K)$ time. Otherwise, we have $\sqrt{n} < m/\log^{3/2} m$, so $n^{2/3} \log^2 m < m^{4/3}$, and thus $(mn)^{2/3} \log^2 m < m^2$. Furthermore we have $n > m$, so certainly $n^{2/3} \log^2 m > m^{1/3}$, and thus $(mn)^{2/3} \log^2 m > m$. Hence we can choose d such that $1 < d < 2$ and $m^d = (mn)^{2/3} \log^2 m$, resulting in a total running time of $O(m^d \log^\delta m + nm(\log^3 m)/\sqrt{m^d} + K) = O((mn)^{2/3} \log^{2+\delta} m + K)$.

Theorem 6.6 *Given a polygon with n edges and m concave vertices, all K pairs of an edge e and a concave vertex v such that the polygon can be immobilized by two frictionless point fingers on e*

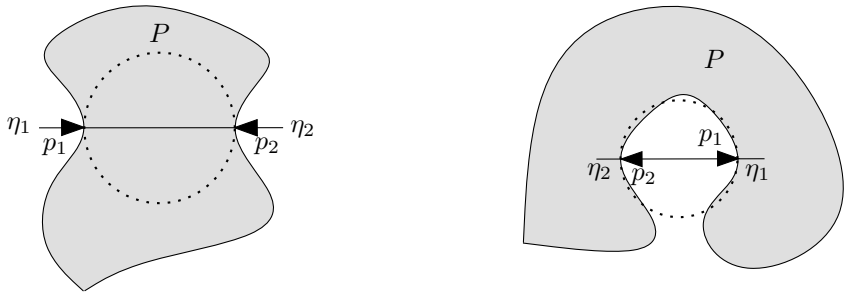


Figure 6.9: Second-order immobility grasps with two frictionless point fingers on semi-algebraic sets.

and at v, can be computed in time $O(n \log^4 m + (mn)^{2/3} \log^{2+\delta} m + K)$.

6.4 Conclusion

We proposed the first efficient output-sensitive algorithms to report all sets of edges and concave vertices of a simple polygon that yield second-order immobility grasps. The algorithms solve geometric problems in the object plane, which are based on the CSU condition [33].

It is open to efficiently compute all second-order immobility grasps of an arbitrary planar object. Three frictionless point fingers can often immobilize a planar object, but when the object is non-convex, two fingers can also immobilize it. We believe that two fingers can achieve second-order immobility, if and only if the following hold: (i) the normal lines coincide in the opposite direction, (ii) there is a circle centered at each contact with radius of an arbitrary small number ε, such that the intersection of the object and this circle is contained in the circle with the two contacts as diameter, and (iii) the corresponding boundaries are concave. See Figure 6.9.

It is also open to efficiently enumerate all second-order immobility grasps of a three-dimensional object. Ponce et al. [61] identified a necessary and sufficient condition for four fingers to hold a three-dimensional object in equilibrium. From this, one may find a necessary and sufficient condition for four fingers to hold a three-dimensional object in second-order immobility.

Chapter 7

Computing All Independent Form-Closure Grasp Regions of Polygons

This chapter is about efficiently reporting all independent form-closure grasp regions of a specified size ε. An independent form-closure grasp region is a set of edge patches (of length ε) and concave vertices, such that the fingers at the vertices and anywhere in the edge patches achieve form closure. Independent form-closure grasp regions are convenient and realistic, because in practice, the fingers cannot be positioned with infinite accuracy. Nguyen [58] studied independent grasp regions for the first time. He showed how to construct maximal independent regions on a given set of faces of a polygon and a polyhedron, such that any placement of two to seven frictional fingers in these regions results in a form-closure grasp. Most form-closure grasps tolerate small misposition of the point fingers, but the permitted amount of positioning error differs from finger to finger, and it can be extremely small. The independent form-closure grasp regions reported in this chapter guarantee form closure with a positioning error of $\varepsilon/2$.

In this chapter, we propose output-sensitive algorithms to report all combinations of concave vertices and edge patches that form independent form-closure grasp regions. The edge patches have a specified length ε, and each set yields a form-closure grasp with at most four frictionless point fingers. We study this problem for a polygon and a rectilinear polygon. For both polygons and rectilinear polygons, the combinations include: (i) edge patch quadruples, (ii) triples of one concave vertex and two edge patches, and (iii) triples of two concave vertices and one edge patch.

This chapter is structured as follows. In Section 7.1, we describe our approach, the shapes of the wrench sets and the intersection algorithms that we use. In Section 7.2, we propose efficient and output-sensitive algorithms to report all combinations of edge patches and concave vertices of a polygon that form independent form-closure grasp regions with three or four frictionless point fingers. When a polygon is rectilinear, we can compute all independent form-closure grasp regions more efficiently. Section 7.3 presents how we do this efficiently in an output-sensitive manner.

7.1 Preliminaries

We predivide the edges into segment pieces of a given length ε. (See Figure 7.1.) We call these segments *edge patches*. Note that a concave vertex does not have any patch, because positioning errors at concave vertices are less likely. We sketch our approach and introduce a projection scheme in Section 7.1.1. When a finger slides along an edge, or a finger is placed at a concave vertex, the corresponding wrench points form some shapes in wrench space. These will be described in Section 7.1.2. Finally, the data structures and algorithms to identify all independent regions will

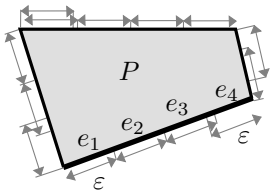


Figure 7.1: The edges are divided into segments of length ε.

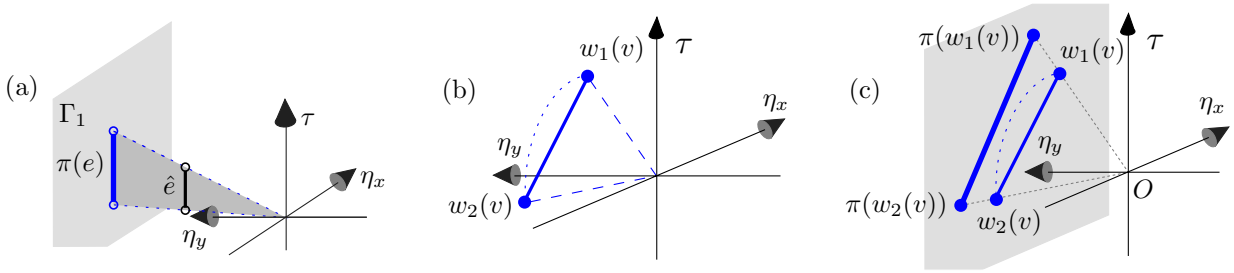


Figure 7.2: (a) The edge wrench patch induced by a finger on an edge patch, and its projection. (b) A concave vertex wrench set, and (c) its projection.

be introduced in Section 7.1.3.

7.1.1 Our approach

We base our approach on the ideas from Chapter 3. Remember that a combination of concave vertices and edges yields a form-closure grasp with at most four fingers if and only if they have four wrench points whose convex hull contains the origin in the interior (Theorem 2.1). For an independent form-closure grasp region, it will be modified as follows: a combination of concave vertices and edge patches form an independent form-closure grasp region with at most four fingers if and only if the convex hull of *any* combination of points from an edge wrench patch and the vertex wrench sets contains the origin strictly in its interior. To identify such wrench patches, we project them on screen Γ, and identify all red and blue wrench patches, such that they make a red and blue intersection for any point from each edge wrench patch (Lemma 2.7). We use the projection scheme and screen Γ defined in Section 2.4.1. We say that the convex hull of red sets cross the convex hull of blue sets, when the convex hull of red points from every red set intersects that of blue points from every blue set in the interior, no matter which point is chosen from each set.

7.1.2 Edge wrench patches and vertex wrench sets

The shapes of edge wrench patches and vertex wrench sets are as described in 3.1.1 of Chapter 3. The wrench of a finger at position p on an edge patch e is $(\eta, \tau)^T = (\eta_x, \eta_y, p \times \eta)^T$. See Section 3.1.1. We assume that η is a unit vector. Then the set of wrench points induced by a finger sliding along e forms a vertical line segment $\hat{e}$ in wrench space. We call this edge wrench set $\hat{e}$ of an edge patch e the *edge wrench patch of* e. The projection of $\hat{e}$ is also a vertical line segment $\pi(\hat{e})$ on Γ—see Figure 7.2 (a). Abusing the notation slightly, we call $\pi(\hat{e})$ the *edge wrench patch of* e as well. We do not include the endpoints in the edge wrench patch for simplicity, although they are included if the endpoints of the edge patch lie in the interior of an edge.

A finger at a vertex induces a set of wrench points, which form a line segment on Γ. Figure 7.2 (b) and (c) show a vertex wrench set in wrench space, and its projection on Γ. We let $s(v)$ denote the projected vertex wrench set of v, and call it the *vertex wrench set of* v as well.

7.1.3 Intersection search algorithms

In this chapter we need to perform two kinds of queries to report all red and blue intersections. To answer these queries, we use the following two: a segment intersection search structure and an extended triangle search structure.

The segment intersection search structure is explained in Section 2.4.2. We can have trade-off between time and space for this structure as explained in Section 6.3 of Chapter 6. The extended triangle search structure can report all k line segments contained in a query triangle. This is an order-6 tree as described in Section 2.4.2. It stores q segments in $O(q^2 \log^4 q)$, and the query time is $O(\log^6 q + k)$.

7.2 Computing all independent form-closure grasp regions of a polygon

This section is about reporting all independent form-closure grasp regions of a polygon. More precisely, we propose output-sensitive algorithms to enumerate all sets of: (i) four edge patches, (ii) one concave vertex and two edge patches, and (iii) two concave vertices and one edge patch, such that four or three frictionless point fingers on these sets yield a form-closure grasp. The case of all pairs of concave vertices is not included, because vertices do not have patches, thus they cannot be an independent form-closure grasp region. All pairs of concave vertices with form-closure grasps can be reported with the algorithm proposed in Section 3.3.3. Throughout this chapter, we let n be the number of edge patches and m be the number of concave vertices of a polygon P. We also let $p \preceq \ell$ to denote the relationship of a point p and a line ℓ where p is on or below ℓ. Likewise, we let $p \succeq \ell$ denote the relationship of p and ℓ where p is on or above ℓ.

7.2.1 Four edge patches

We wish to report all edge patch quadruples that form independent form-closure grasp regions. An edge patch quadruple forms an independent form-closure grasp region, if and only if the corresponding red and blue trapezoids cross each other—see Section 7.1.1. Fingers on edges e_1 and e_2, they induce two wrench points w_1 and w_2. When the fingers slide along e_1 and e_2, these points w_1 and w_2 slide along $\hat{e}_1$ and $\hat{e}_2$. The set of line segments connecting w_1 and w_2 for any $w_1 \in \hat{e}_1$ and $w_2 \in \hat{e}_2$ equals the convex hull of $\hat{e}_1$ and $\hat{e}_2$. The projection of these line segments equals the convex hull of $\pi(\hat{e}_1)$ and $\pi(\hat{e}_2)$. Since $\pi(\hat{e}_1)$ and $\pi(\hat{e}_2)$ are vertical line segments, the convex hull of $\pi(\hat{e}_1)$ and $\pi(\hat{e}_2)$ is a trapezoid. The problem is thus formulated as follows: given all red and blue trapezoids, report all pairs of red and blue trapezoids that cross each other on Γ. Two trapezoids cross each other, when they intersect each other, and the vertical sides of the trapezoids are disjoint with the intersection. See Figure 7.3.

Let $r_u r_d$ and $r'_u r'_d$ be the red vertical segments,[1] and $b_u b_d$ and $b'_u b'_d$ be the blue vertical segments on Γ. Let r_u, r'_u, b_u and b'_u be the upper endpoints, and r_d, r'_d, b_d and b'_d be the lower endpoints of the segments. The convex hulls of $r_u r_d$ and $r'_u r'_d$, and $b_u b_d$ and $b'_u b'_d$ are red and blue trapezoids respectively. We assume that $r_u r_d$ is on the left side of the line containing $r'_u r'_d$, and $b_u b_d$ is on the left side of the line containing $b'_u b'_d$. Suppose that red trapezoid $r_u r_d r'_d r'_u$ crosses blue trapezoid

[1] Note that the vertical segments are projected edge wrench patches.

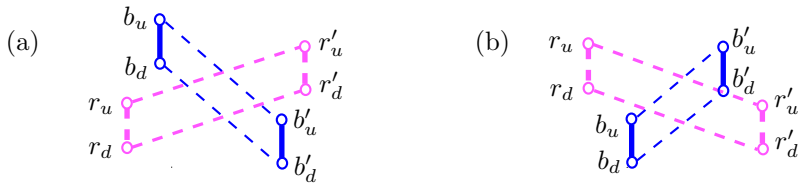


Figure 7.3: The cases of red and blue trapezoids that cross each other.

$b_u b_d b'_d b'_u$. Note that the endpoints r_u, r'_u, b_u, b'_u, r_d, r'_d, b_d and b'_d are not included in the edge wrench patches, since a finger cannot be at a vertex. Hence these endpoints are allowed to be on the boundary line segments $\overline{r_u r'_u}$, $\overline{r_d r'_d}$, $\overline{b_u b'_u}$ and $\overline{b_d b'_d}$. We let $\ell(x, x')$ be the supporting line of the line segment $\overline{x, x'}$. The following lemma is a necessary and sufficient condition for $r_u r_d r'_u r'_d$ to cross $b_u b_d b'_u b'_d$. Figure 7.3 illustrates Lemma 7.1.

Lemma 7.1 *Two trapezoids $r_u r_d r'_u r'_d$ and $b_u b_d b'_u b'_d$ cross each other, if and only if one of the following holds:*

(i) $r_u \preceq \ell(b_d b'_d)$, $r'_d \succeq \ell(b_u b'_u)$, $b_d \succeq \ell(r_u r'_u)$, *and* $b'_u \preceq \ell(r_d r'_d)$;

(ii) $r_d \succeq \ell(b_u b'_u)$, $r'_u \preceq \ell(b_d b'_d)$, $b_u \preceq \ell(r_d r'_d)$, *and* $b'_d \succeq \ell(r_d r'_d)$.

We use a segment intersection search structure to identify all red and blue trapezoids that satisfy Lemma 7.1. To identify pairs that satisfy the first condition of Lemma 7.1, we store $(r_u, r'_d, \ell(r_u r'_u)^*, \ell(r_d r'_d)^*)$ of all red trapezoids at each level of a segment intersection search structure, and query with $\ell(b_d b'_d)$, $\ell(b_u b'_u)$, b^*_d and b'^*_u of a blue trapezoid, for each level of the structure. Note that $\ell(r_u r'_u)^*$ and $\ell(r_d r'_d)^*$ are the dual points of segments $\ell(r_u r'_u)$ and $\ell(r_d r'_d)$ respectively. Likewise, b^*_d and b'^*_u are the dual lines of points b_d and b'_u respectively. See Section 2.4.2. Those that satisfy the second condition of Lemma 7.1 can be found similarly.

There are $O(n^2)$ red and blue trapezoids, so we will use the segment search structure with trade-off. We use $O(n^{8/3})$ storage, then the preprocessing time is $O(n^{8/3} \log^{\delta} n)$, where δ is an arbitrarily small number. The query time is $O(n^{2/3} \log^{O(1)} n + k)$—to report k segments intersecting each of $O(n^2)$ query segments. The following theorem summarizes the result.

Theorem 7.2 *Given a polygon with n edge patches, all K edge patch quadruples that form an independent form-closure grasp region for four frictionless point fingers can be computed in time $O(n^{8/3} \log^{O(1)} n + K)$.*

7.2.2 One concave vertex and two edge patches

We wish to report all triples of one concave vertex and two edge patches that form independent form-closure grasp regions for three frictionless point fingers. Note that two fingers on two edges induce a line segment on Γ, the endpoints of which are from two vertical segments (two projected edge wrench patches). A set of two edge patches and a concave vertex forms an independent form-closure grasp region if and only if the corresponding red (blue) line segment cross blue (red) trapezoid, i.e. if and only if any blue (red) segment connecting two points from the two vertical sides intersects the red (blue) segment in the interior. A line segment crosses a trapezoid, if they have non-empty intersection, and the vertical sides of the trapezoid and the endpoints of the line segment are disjoint with the intersection. Our problem is thus to report all red and blue line segment and trapezoid that cross each other.

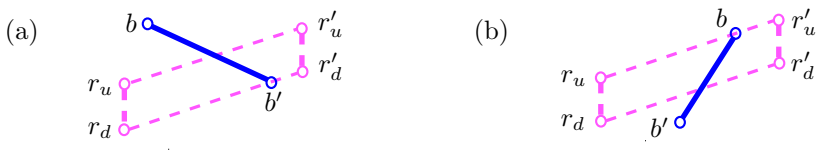


Figure 7.4: The cases of a blue line segment crossing a red trapezoid.

Without loss of generality, we take a blue line segment $\overline{bb'}$ and a red trapezoid $r_u r_d r'_d r'_u$. As in the previous section, b is the left endpoint and b' is the right endpoint. Also r_u, r_d, r'_u and r'_d are defined as in the previous section. The following lemma is a necessary and sufficient condition for $\overline{bb'}$ crossing $r_u r_d r'_d r'_u$. Figure 7.3 illustrates Lemma 7.3.

Lemma 7.3 *A blue line segment $\overline{bb'}$ crosses a red trapezoid $r_u r_d r'_d r'_u$, if and only if one of the following holds:*

(i) $b \succeq \ell(r_u r'_u)$, $b' \preceq \ell(r_d r'_d)$, $r_u \preceq \ell(bb')$, *and* $r'_d \succeq \ell(bb')$;

(ii) $b \preceq \ell(r_d r'_d)$, $b' \succeq \ell(r_u r'_u)$, $r_d \succeq \ell(bb')$, *and* $r'_u \preceq \ell(bb')$.

We use a segment intersection search structure to identify the red and blue line segment and trapezoid that satisfy the two conditions in Lemma 7.3. To identify all pairs that satisfy the first condition of Lemma 7.3, we store b, b', $\ell(bb')^*$ and $\ell(bb')^*$ of all blue line segments at each level of a segment intersection search structure,[2] and query with the half planes bounded by $\ell(r_u r'_u)$, $\ell(r_d r'_d)$, r^*_u and r'^*_d of a red trapezoid, for each level of the structure.

There are $O(n^2)$ red and blue trapezoids, and $O(m)$ blue and red line segments. We store $O(m)$ line segments in a segment search structure. The preprocessing time is $O(m^2 \log^2 m)$, and the query time is $O(\log^4 m + k)$—to report k segments intersecting each of $O(n^2)$ query trapezoids. The following theorem summarizes the result.

Theorem 7.4 *Given a polygon with m concave vertices and n edge patches, all K combinations of one concave vertex and two edge patches that form an independent form-closure grasp region for three frictionless point fingers can be computed in time $O(n^2 \log^4 m + K)$.*

7.2.3 Two concave vertices and one edge patch

We wish to report all triples of two concave vertices v and v' and one edge patch e that form independent form-closure grasp regions for three frictionless point fingers. A set (v, v', e) forms an independent form-closure grasp region, if and only if the corresponding red or blue quadrilateral induced by v and v' contains the blue or red line segment induced by e. The reason is that two fingers at v and v' always induce the points in the convex hull of the two wrench line segments $s(v)$ and $s(v')$. If the convex hull of $s(v)$ and $s(v')$ contains the whole edge wrench patch, no matter where a finger is placed on e, the corresponding wrench point will be contained in the quadrilateral, thus result in a red and blue intersection. This is illustrated in Figure 7.3.

Our problem is thus to report all red (blue) quadrilaterals containing blue (red) line segments (edge wrench patches). We can use two solutions, depending on the relative size of m compared to n. When $m < \sqrt{n}$, we check all triples of two concave vertices and one edge patch. It takes $O(m^2 n)$ time.

[2]Recall that a segment intersection search structure has four levels.

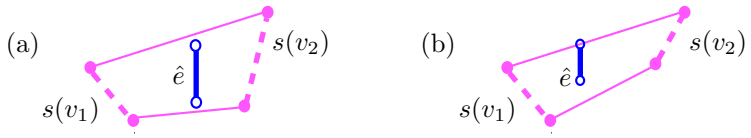


Figure 7.5: A red quadrilateral contains a blue vertical line segment.

When $m \geq \sqrt{n}$, we do the following. Without loss of generality, we take a red quadrilateral $r_u r_d r'_d r'_u$, and a blue vertical line segment $\overline{bb'}$. There are $O(m^2)$ red quadrilaterals and $O(n)$ blue (vertical) line segments. On $O(n)$ blue segments, we build a two-level triangle search structure, which is an order 6 tree. We store (b, b, b, b', b', b') at each level. The preprocessing time is $O(n^2 \log^4 n)$—see Section 3.1.2. A quadrilateral can be decomposed into two disjoint triangles $\triangle_1$ and $\triangle_2$, by cutting along a diagonal. The endpoints b and b' can be in $\triangle_1$, or in $\triangle_2$, or one in $\triangle_1$ and another in $\triangle_2$. Let σ_1, σ_2 and σ_3 be the half planes, such that each of them is bounded by the supporting line of each boundary segment of $\triangle_1$, and that $\triangle_1$ lies in the half planes. We also let σ_4, σ_5 and σ_6 be the half planes, such that each of them is bounded by the supporting line of each boundary segment of $\triangle_2$, and that $\triangle_2$ lies in the half planes. For each of $O(m^2)$ query quadrilaterals, we perform the following four queries: $(\sigma_1, \sigma_2, \sigma_3, \sigma_1, \sigma_2, \sigma_3)$, $(\sigma_4, \sigma_5, \sigma_6, \sigma_4, \sigma_5, \sigma_6)$, $(\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6)$, and $(\sigma_4, \sigma_5, \sigma_6, \sigma_1, \sigma_2, \sigma_3)$. Each query can be performed in $O(\log^6 n + k)$ time to report k blue segments. We can build the structure on red segments, and query with blue triangles, in the same way, with the same time bound. The following theorem summarizes the result.

Theorem 7.5 *Given a polygon with m concave vertices and n edge patches, all K combinations of two concave vertices and one edge patch that form an independent form-closure grasp region for three frictionless point fingers can be computed in time:*

1. $O(m^2 n)$ *time when* $m < \sqrt{n}$, *and*
2. $O(n^2 \log^4 n + m^2 \log^6 n + K) = O(n^2 \log^6 n + K)$ *time when* $m \geq \sqrt{n}$.

7.3 Computing all independent form-closure grasp regions of a rectilinear polygon

This section is about reporting all independent form-closure grasp regions of a rectilinear polygon. More precisely, we propose output-sensitive algorithms to enumerate all sets of: (i) four edge patches, (ii) one concave vertex and two edge patches, and (iii) two concave vertices and one edge patch, such that four or three frictionless point fingers on these sets yield a form-closure grasp.

As mentioned in Section 7.1, the edges are divided into edge patches of length ε. We use the same projection scheme, screen and notations described in Section 3.3. We divide the edge patches and concave vertices into four families as in Section 3.3. The families of edge patches are E, N, W and S, and those of concave vertices are EN, WN, ES and WS. Note that n is the number of edge patches and m is the number of concave vertices of a rectilinear polygon P.

We use the projection scheme described in Section 2.4.1, and the screen described in Section 3.3. The problem of reporting all independent form-closure grasp regions is easier for rectilinear polygons than for arbitrary polygons, because the edge wrench patches and the vertex wrench sets are more regularly positioned. More precisely, the edge wrench patches on Γ lie on three

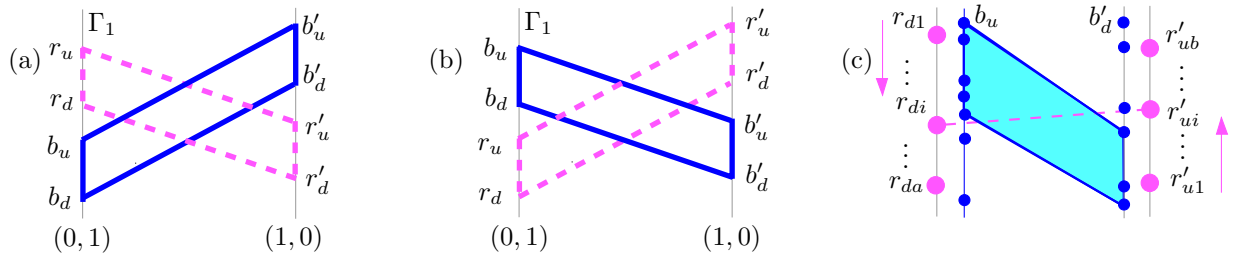


Figure 7.6: (a) (b) An illustration of Lemma 7.6. (c) An illustration of the algorithm.

lines: $(1,0)$, $(0,1)$ and $(-1,0)$.[3] See Section 3.3 for more details. When we say that two trapezoids (or a trapezoid and a line segment) cross each other, it means that they intersect each other, and the vertical sides of the trapezoid and the endpoints of the line segment are disjoint with the intersection.

7.3.1 Four edge patches

Four edge patches form an independent form-closure grasp region, if and only if the corresponding red and blue trapezoids cross each other. The following lemma states a necessary and sufficient condition.

Lemma 7.6 *Two trapezoids $r_u r_d r'_u r'_d$ and $b_u b_d b'_u b'_d$ cross each other, if and only if one of the following two holds:*

(i) $r_d > b_u$ *and* $r'_u < b'_d$*;*

(ii) $r_u < b_d$ *and* $r'_d > b'_u$*.*

Proof: It is straightforward to see the "if" direction, so we show the "only if" direction. Since $r_u r_d r'_u r'_d$ and $b_u b_d b'_u b'_d$ cross each other, any blue segment $\overline{bb'}$ intersects any red line segment $\overline{rr'}$ in the interior. Because $\overline{bb'}$ intersects $\overline{rr'}$ in the interior, $\overline{bb'}$ and $\overline{rr'}$ must satisfy either (1) $r > b$ and $r' < b'$, or (2) $r < b$ and $r' > b'$. Note that $b_d \le b \le b_u$, $b'_d \le b' \le b'_u$, $r_d \le r \le r_u$ and $r'_d \le r' \le r'_u$. Combining these with the two conditions (1) and (2) gives condition (i) and (ii) stated in the lemma. □

We use an approach similar to that in Section 3.3. We have $O(n)$ red and blue edge wrench patches. We build sorted lists of b_u, b_d, b'_u and b'_d in $O(n \log n)$ time. We also sort r_u and r_d from top to bottom, then r'_u and r'_d from bottom to top. See Figure 7.6. Here we show how to report all quadrilaterals that satisfy the first condition in Lemma 7.1. Let $r_{d1}, r_{d2}, \cdots, r_{da}$ and $r'_{u1}, r'_{u2}, \cdots, r'_{ub}$ be the sorted lists of r_d's and r'_u's. We take r_{d1}, and identify all blue $\pi(\hat{e})$ such that $r_d > b_u$ in $O(\log n + k)$ time, and put them in set A. We take r'_{u1}, and identify all blue $\pi(\hat{e})$ such that $r'_u < b'_d$ in $O(\log n + k)$ time, and put them in set A'. We report the Cartesian product $A \times A'$. For a given r_{d1} we repeat this process for each of r'_{ui} ($i = 1, 2, \cdots, b$). When we go from r'_{ui} to $r'_{u(i+1)}$, we do not have to perform a binary search in the lists; we check the neighbors in the list until we find one that satisfies the condition. We repeat the whole process for each of r_{di} ($i = 1, 2, \cdots, b$). There are $O(n)$ choices for r_{di}, and for each of them, the query time takes $O(\log n + k)$, thus the total time complexity is $O(n \log n + K)$.

[3] The lines $(1,0)$, $(0,1)$ and $(-1,0)$ are in fact $(1,0,\tau)$, $(0,1,\tau)$ and $(-1,0,\tau)$ lines, which lie on Γ from the construction of Γ.

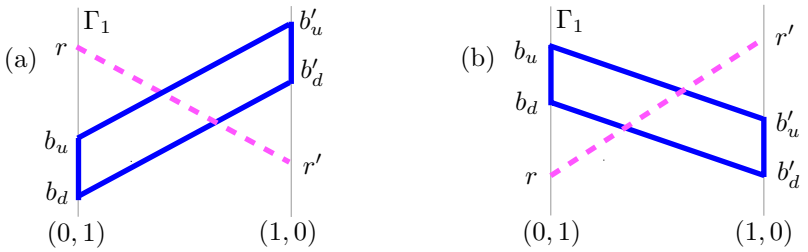


Figure 7.7: An illustration of Lemma 7.8.

Theorem 7.7 *All K edge patch quadruples of a rectilinear polygon that form independent form-closure grasp regions with four frictionless point fingers can be enumerated in $O(n \log n + K)$ time.*

7.3.2 One concave vertex and two edge patches

Two edge patches and one concave vertex form an independent form-closure grasp region with three frictionless point fingers, if and only if the red (blue) trapezoid crosses the blue (red) line segment. The following lemma states a necessary and sufficient condition for a red line segment to cross a blue trapezoid. Lemma 7.8 can be easily changed to check whether a blue line segment crosses a red trapezoid. The proof for Lemma 7.6 can be easily modified to prove Lemma 7.8, so we omit the proof.

Lemma 7.8 *A red line segment $\overline{rr'}$ intersects a blue trapezoid $b_u b'_u b'_d b_d$ in the interior independently, if and only if one of the following two holds:*

(i) $r \geq b_u$ *and* $r' \leq b'_d$*;*

(ii) $r \leq b_d$ *and* $r' \geq b'_u$*.*

We use the same approach used in the previous section 7.3.1. One difference is that the query is a red line segment instead of a trapezoid. There are $O(n)$ blue edge wrench patches, and $O(m)$ red query line segments, so the time complexity for this case is $O(n \log n + m \log n + K) = O(n \log n + K)$.

Theorem 7.9 *All K triples of one concave vertex and two edge patches of a rectilinear polygon that form independent form-closure grasp regions with three frictionless point fingers can be enumerated in $O(n \log n + K)$ time.*

7.3.3 Two concave vertices and one edge patch

We wish to report all triples of two concave vertices and an edge patch of a rectilinear polygon that form an independent form-closure grasp region with three frictionless point fingers. Such a triple of two concave vertices and an edge patch belongs to one of the two cases: when the two concave vertices induce three different normal directions, and when they induce four different normal directions. See Section 3.3.4.

We first look at the first case when the vertices induce three different normal directions. We take the convex hulls of red points and blue points, which form red and blue triangles. Note that the triangle pairs that we will consider have both of the vertical sides on one of the three lines: $(0,1)$, $(1,0)$ or $(0,-1)$—see Section 3.3.4. If the projected red and blue triangles cross each other, the

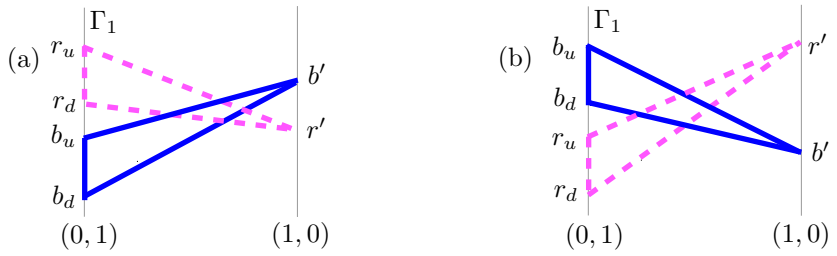


Figure 7.8: An illustration of Lemma 7.10 on Γ_1.

triple forms an independent form-closure grasp region. The following lemma provides a necessary and sufficient condition for a pair of red and blue triangles to cross each other. The proof for Lemma 7.6 can be easily modified to prove Lemma 7.10, so we omit the proof.

Lemma 7.10 *A blue triangle $bb'_ub'_d$ crosses a red triangle $rr'_ur'_d$, if and only if one of the following two holds:*

(i) $b_u \leq r_d$ *and* $b' > r'$*;*

(ii) $b_d \geq r_u$ *and* $b' < r'$*.*

Here we describe how we report all pairs of red and blue triangles that satisfy Lemma 7.10. There are $O(n)$ edge wrench patches and $O(m)$ concave vertex wrench sets. Without loss of generality, assume that the edge wrench patches are blue. Then there are $O(nm)$ blue triangles and $O(m^2)$ red triangles. We build binary search trees on the upper and lower endpoints (b_u and b_d) of the blue edge wrench patches in $O(n \log n)$ time. We take all $O(m^2)$ vertex wrench set pairs with three red endpoints r_u, r_d and r' (r_u and r_d on $(0,1)$ line and $r_d < r_u$) and one blue point b' on $(1,0)$ line. For each of these $O(m^2)$ vertex wrench set pairs, we find the following: (i) if $b' > r'$, find all blue edge wrench patches b_ub_d, such that $b_u \leq r_d$, (ii) if $b' < r'$, find all blue edge wrench patches b_ub_d, such that $b_d \geq r_u$. All k such blue edge wrench patches can be reported in $O(\log n + k)$ time. Therefore the total time complexity of this case is $O(n \log n + m^2 \log n + K)$. This approach can easily be modified to search for the pairs of red and blue triangles, whose vertical sides are on the right. It has the same time complexity—$O(n \log n + m^2 \log n + K)$.

Now we look at the second case when the vertices induce four different normal directions. Without loss of generality, we take a triple of two concave vertices and an edge patch (v_{EN}, v_{WS}, e_N). The projections of the edge wrench patch and the vertex wrench sets on Γ_1 will be as follows. A blue endpoint b of $\pi(\hat{v}_{EN})$, the blue segment b_ub_d ($b_d < b_u$) of $\pi(\hat{e}_N)$ and a red endpoint r of $\pi(\hat{v}_{WS})$ are on the line $(0,1)$. The other blue endpoint b' of $\pi(\hat{v}_{EN})$ and the other red endpoint r' of $\pi(\hat{v}_{WS})$ are on the line $(1,0)$. If the projected red segment and blue triangle cross each other, the triple forms an independent form-closure grasp region. The following lemma provides a necessary and sufficient condition for a pair of red segment and blue triangle to cross each other. Lemma 7.11 can easily be modified to report all blue segment and red triangle that cross each other. Figure 7.9 illustrates Lemma 7.11. The proof for Lemma 7.6 can be easily modified to prove Lemma 7.11, so we omit the proof.

Lemma 7.11 *A red segment rr' crosses the convex hull of bb' and b_ub_d, if and only if one of the following two holds:*

(i) $b < r$ *and* $b' > r'$ *and* $b_u < r$*;*

(ii) $b > r$ *and* $b' < r'$ *and* $b_d > r$*.*

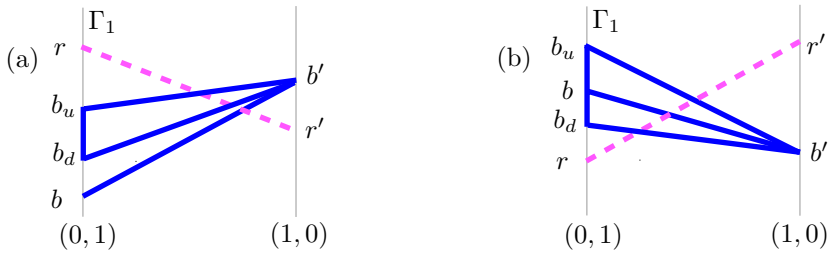


Figure 7.9: An illustration of Lemma 7.11.

We describe how we report all pairs of red segment and blue triangle that satisfy Lemma 7.11. Note that there are $O(n)$ blue vertical segments $b_u b_d$ and $O(m)$ red and blue segments. We build a two-level orthogonal search tree on the endpoints of blue segments (projected vertex wrench sets) in $O(m \log m)$ time. We also build binary search trees on the upper endpoints (b_u) and the lower endpoints (b_d) of all blue (projected) edge wrench patches in $O(n \log n)$ time. For each segment rr' of all $O(m)$ projected vertex wrench sets, we find the following: (i) find all k_1 blue segments bb', such that $b < r$ and $b' > r'$, and find all k_2 blue vertical segments $b_u b_d$, such that $b_u < r$; (ii) find all k_1 blue segments bb', such that $b > r$ and $b' < r'$, and find all k_2 blue vertical segments $b_u b_d$, such that $b_d > r$. All reported $k_1 k_2$ pairs of $b_u b_d$ and bb' satisfy Lemma 7.11, and they can be reported in $O(\log^2 m + \log n + k_1 k_2)$ time. Therefore the total time complexity of this case is $O(n \log n + m(\log^2 m + \log n) + K) = O(n \log^2 n + K)$.

The following summarizes the result.

Theorem 7.12 *All K triples of two concave vertices and an edge patch of a rectilinear polygon that form independent form-closure grasp regions with three frictionless point fingers can be enumerated in $O(n \log^2 n + m^2 \log n + K)$ time.*

7.4 Conclusion

In this chapter, we proposed efficient output-sensitive algorithms to report all combinations of edge patches and concave vertices that form independent form-closure grasp regions for at most four frictionless point fingers, involving prespecified edge patches. By projecting the wrench sets, this becomes red and blue intersection search problem on planes. In particular, the problem for rectilinear polygons boils down to orthogonal range search problems.

When the edge patches are not given beforehand, one may need to know all combinations of edge patches and concave vertices, which contain at least one independent form-closure grasp region. Efficiently reporting such combinations is open.

The approach presented in this chapter can be extended to any planar object, as long as there are efficient algorithms or query data structures that can report all wrench sets contained in a given half-plane. A first challenge would be to compute all independent form-closure grasp regions of a planar semi-algebraic sets. To achieve this, we need an algorithm or a data structure that can efficiently report all arcs contained in a given half-plane.

Chapter 8

Computing All Form-Closure Grasps of a Rectilinear Polyhedron

Many researchers provided ways of computing one form-closure grasp or many form-closure grasps of three-dimensional objects [13, 35, 39, 53, 54, 56, 58, 85, 88, 90]. But, there is no algorithm to enumerate all form-closure grasps of a three-dimensional object efficiently, as in the case of fixturing a two-dimensional object. This is not surprising, because the problem itself is more complicated than the two-dimensional fixturing problem; it has high dimensionality—a directed line in a three-dimensional space has six degrees of freedom instead of three.

In this chapter, we propose algorithms to compute all form-closure grasps with at most seven frictionless point fingers for a rectilinear polyhedron, which is the first attempt to compute *all* grasps for a three-dimensional object. We use a form-closure condition in wrench space, which is similar to Theorem 2.1 for two-dimensional objects. As a matter of fact, there is no easier, intuitive or graphical form-closure condition expressed in terms of the geometry of the normal lines in the three-dimensional object space, as far as we know. For this reason, we use the form-closure condition in wrench space. In this chapter, we propose a form-closure condition which is composed of two conditions on three-dimensional vectors. Intuitively, this comes from the following view of fixturing: a three-dimensional object can be immobilized (fixtured) by holding it such that it cannot move horizontally, and again holding it such that it cannot move vertically. One can see it as squeezing the object vertically (along z direction), and putting the object in form-closure with respect to the horizontal plane (parallel to the xy plane)—achieving two-dimensional form-closure on the projection of the object onto the xy plane. A rectilinear polyhedron is an obvious example where we can apply the horizontal and vertical immobilization in a straightforward way.

This chapter is structured as follows. We first introduce form-closure conditions in wrench space and also in two subspaces of wrench space in Section 8.1. We also present notations and detailed information required by the algorithm such as the shapes of wrenches, projection schemes and the intersection algorithms that we use. In Section 8.2, we propose algorithms to report all combinations of faces, concave edges and concave vertices of a rectilinear polyhedron that allow form-closure grasps with at most seven frictionless point fingers. All algorithms except for one case presented in this chapter are sensitive to K' and K, where K' and K are the sizes of the intermediate output and the final output. The algorithm for the exceptional case is sensitive to K' only.

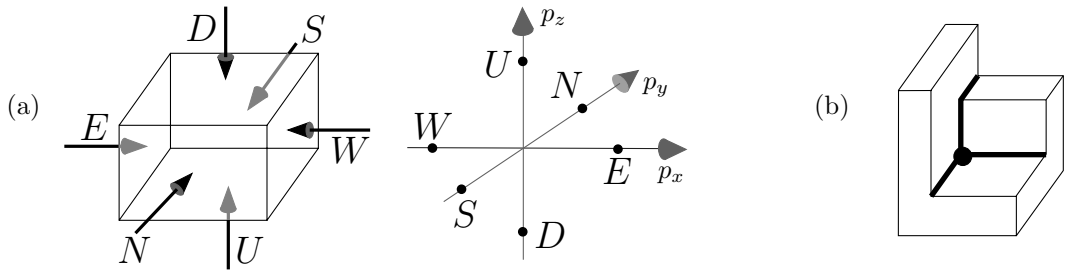


Figure 8.1: (a) Families of faces E, W, N, S, U and D. (b) The concave edges are in thick line segments, and a big dot represents a concave vertex, which is from WND here.

8.1 Preliminaries

Here we introduce the notations, conditions, and other information that we need in this chapter. We divide the faces, concave edges and concave vertices into some families according to their normal directions. In Section 8.1.1, we introduce these families. We also define the wrench sets of faces, concave edges and concave vertices, and their h and ν components. In Section 8.1.2, we present two form-closure conditions: one in six-dimensional wrench space, and the other in two subspaces of wrench space. One can check whether a given set of faces, concave edges and concave vertices achieve form-closure by solving two three-dimensional problems. If we project three-dimensional vectors on a plane, the three-dimensional subproblems turn out to be two-dimensional intersection problems, as described in the previous chapters, e.g. Chapter 3. The projection scheme is presented in this section as well. In Section 8.1.3, we discuss the algorithms to tackle intersection search problems.

8.1.1 Families of faces, concave edges and concave vertices

Let P be a rectilinear polyhedron with n faces. We triangulate the faces of P, and each triangle of the $O(n)$ triangulated faces will be considered as a face. We place P in an xyz coordinate system, such that the origin lies in the interior of P. In Section 3.3, we divided the faces of a rectilinear polygon into four families according to the normal directions, namely, E, W, N and S. For a rectilinear polyhedron, we divide the faces into six families according to the normal directions, namely, E, W, N, S, U and D. The faces from E, W, N and S are vertical (i.e. perpendicular to the xy plane), and those from U and D are horizontal (i.e. parallel to the xy plane). For simplicity, we normalize the normal direction vector η of faces—η has a unit length.

According to which faces a concave edge is adjacent to, the concave edges are divided into twelve families, namely, EN, ES, WN, WS, EU, WU, NU, SU, ED, WD, ND and SD. If a concave edge is adjacent to two faces from E and N, it belongs to family EN. Figure 8.1 shows concave edges from ND, WD and WN. Likewise, the concave vertices belong to one of the following eight families, according to which faces they are adjacent to: ENU, ESU, WNU, WSU, END, ESD, WND and WSD. If a concave vertex is adjacent to three faces from W, N and D, it belongs to family WND. Figure 8.1 shows a concave vertex from WND.

When a finger pushes an object on a face, force is applied along the inward normal line, which we call *line of force*. This force makes the object translate and/or rotate, depending on where the line of force is. A line of force plays a key role to describe the instantaneous motion that the force causes. A *wrench* is a six-dimensional description of a directed line in three-dimensional space, and it is defined as $(\eta, p \times \eta)$, where η is a direction vector, and p is the position vector of a point

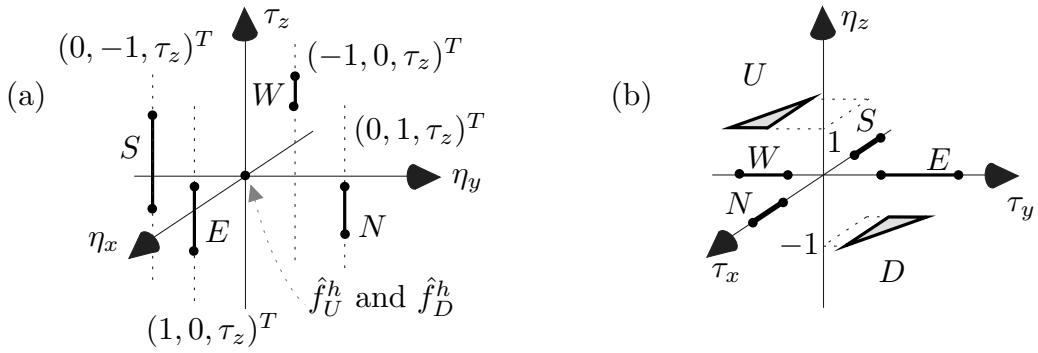


Figure 8.2: (a) $\hat{f}^h$'s and (b) $\hat{f}^\nu$'s from each family, when the position vectors have positive numbers only.

on the line. The magnitude of a wrench $(\eta, p \times \eta)^T$ represents the magnitude of force. As one can see, a directed line has a set of wrench vectors.[1] We normalize the direction vector η, i.e. the directed line can be represented as a unique wrench point in this chapter. However, to satisfy the form closure condition, only the direction of the wrench matters, not the magnitude.[2]

We place a finger at position $p = (p_x, p_y, p_z)^T$ on a face, with the inward normal direction η. The inward normal line can be represented as a wrench point $(\eta, p \times \eta)^T = (\eta_x, \eta_y, \eta_z, \tau_x, \tau_y, \tau_z)^T$. Let $\hat{f}$ be the set of wrench points of a finger when it moves in the interior of a face f. More precisely,

$$\hat{f} = \{(\eta, \tau)^T\} = \{(\eta, p \times \eta)^T \mid p \in int(f)\},$$

where $int(f)$ denotes the set of interior points of f. We call $\hat{f}$ the *face wrench set of* f. Let $\hat{f}_E$, $\hat{f}_W$, $\hat{f}_N$, $\hat{f}_S$, $\hat{f}_U$, $\hat{f}_D$ be the wrench sets of a finger on faces from E, W, N, S, U and D respectively. When a finger moves on faces from E and W, only p_x is fixed. Similarly, when a finger moves on faces from N and S, only p_y is fixed, and on those from U and D, only p_z is fixed. Throughout this chapter, let $(p_{xe}, p_{ye}, p_{ze})^T$, $(p_{xw}, p_{yw}, p_{zw})^T$, $(p_{xn}, p_{yn}, p_{zn})^T$, $(p_{xs}, p_{ys}, p_{zs})^T$, $(p_{xu}, p_{yu}, p_{zu})^T$ and $(p_{xd}, p_{yd}, p_{zd})^T$ be the position of an interior point of faces $f_E \in E$, $f_W \in W$, $f_N \in N$, $f_S \in S$, $f_U \in U$ and $f_D \in D$, respectively. The following shows the face wrench set from each family.

$$\hat{f}_E = \{(1,0,0,0,p_{ze},-p_{ye})^T \mid p_{ye} \text{ and } p_{ze} \text{ are variables in } f \in E\},$$

$$\hat{f}_W = \{(-1,0,0,0,-p_{zw},p_{yw})^T \mid p_{yw} \text{ and } p_{zw} \text{ are variables in } f \in W\},$$

$$\hat{f}_N = \{(0,1,0,-p_{zn},0,p_{xn})^T \mid p_{xn} \text{ and } p_{zn} \text{ are variables in } f \in N\},$$

$$\hat{f}_S = \{(0,-1,0,p_{zs},0,-p_{xs})^T \mid p_{xs} \text{ and } p_{zs} \text{ are variables in } f \in S\},$$

$$\hat{f}_U = \{(0,0,1,p_{yu},-p_{xu},0)^T \mid p_{xu} \text{ and } p_{yu} \text{ are variables in } f \in U\},$$

$$\hat{f}_D = \{(0,0,-1,-p_{yd},p_{xd},0)^T \mid p_{xd} \text{ and } p_{yd} \text{ are variables in } f \in D\}.$$

We define η, h and ν components to be $(\eta_x, \eta_y, \eta_z)^T$, $(\eta_x, \eta_y, \tau_z)^T$ and $(\eta_z, \tau_x, \tau_y)^T$, respectively. We let $\hat{f}^h$, $\hat{e}^h$ and $\hat{v}^h$ be the h components, and $\hat{f}^\nu$, $\hat{e}^\nu$ and $\hat{v}^\nu$ be the ν components of $\hat{f}$, $\hat{e}$ and $\hat{v}$ respectively. Observe that $\hat{f}^h$ describes the projection of the normal line of a face f on the xy plane. More precisely, $\hat{f}^h = (\eta_x, \eta_y, (p_x, p_y)^T \times (\eta_x, \eta_y)^T)^T$. The remaining components make $\hat{f}^\nu$. We call the spaces of η, h and ν of all wrenches $\mathcal{N}$, $\mathcal{H}$ and $\mathcal{V}$ *spaces*.

[1] Note that η's with different lengths correspond to different wrench points.

[2] This is for the same reason for the form closure of a planar object, explained in Section 2.4.1. It will be further explained in Section 8.1.2.

We first look at the shapes of $\hat{f}^h$ and $\hat{f}^\nu$ from each family. We place the plane $\eta_x\eta_y$ horizontally in $\mathcal{H}$ space, and the plane $\tau_x\tau_y$ horizontally in $\mathcal{V}$ space. In $\mathcal{H}$ space, $\hat{f}^h_E$, $\hat{f}^h_W$, $\hat{f}^h_N$ and $\hat{f}^h_S$ are vertical line segments parallel to τ_z axis; as a finger moves in the interior of a face, $(\eta_x, \eta_y)^T$ remains the same, but τ_z changes. Moreover, they lie on the lines $(1, 0, \tau_z)^T$, $(-1, 0, \tau_z)^T$, $(0, 1, \tau_z)^T$, and $(0, -1, \tau_z)^T$, where the line $(1, 0, \tau_z)^T$ is defined as $\{(1, 0, 0)^T + \lambda(0, 0, 1)^T \mid \lambda \in \mathbb{R}\}$, the line $(-1, 0, \tau_z)^T$ as $\{(-1, 0, 0)^T + \lambda(0, 0, 1)^T \mid \lambda \in \mathbb{R}\}$, the line $(0, 1, \tau_z)^T$ as $\{(0, 1, 0)^T + \lambda(0, 0, 1)^T \mid \lambda \in \mathbb{R}\}$, and the line $(0, -1, \tau_z)^T$ as $\{(0, -1, 0)^T + \lambda(0, 0, 1)^T \mid \lambda \in \mathbb{R}\}$. The h components $\hat{f}^h_U$ and $\hat{f}^h_D$ lie at the origin of $\mathcal{H}$ space, because the inward normal directions have zeros for η_x and η_y, which makes τ_z also zero. See Figure 8.2 (a). The ν components $\hat{f}^\nu_E$ and $\hat{f}^\nu_W$ are of the form $\{(0, 0, 0)^T + p_z(0, 0, \pm 1)^T \mid p_z \text{ is in an interval of } \mathbb{R}\}$; they are line segments on τ_y axis. The ν components $\hat{f}^\nu_N$ and $\hat{f}^\nu_S$ are of the form $\{(0, 0, 0)^T + p_z(0, \mp 1, 0)^T \mid p_z \text{ is in an interval of } \mathbb{R}\}$; they are line segments on τ_x axis. The ν components $\hat{f}^\nu_U$ and $\hat{f}^\nu_D$ are of the form $(\pm 1, \pm p_y, \mp p_x)^T$; they are the faces of the polyhedron, rotated by 90 degrees, and placed on the planes $\eta_z = 1$ and $\eta_z = -1$ respectively. See Figure 8.2 (b). The followings are formal expressions of $\hat{f}^h$ and $\hat{f}^\nu$ from each family.

$$\hat{f}^h_E = \{(1, 0, -p_{ye})^T \mid p_{ye} \text{ is a variable in } f \in E\},$$
$$\hat{f}^h_W = \{(-1, 0, p_{yw})^T \mid p_{yw} \text{ is a variable in } f \in W\},$$
$$\hat{f}^h_N = \{(0, 1, p_{xn})^T \mid p_{xn} \text{ is a variable in } f \in N\},$$
$$\hat{f}^h_S = \{(0, -1, -p_{xs})^T \mid p_{xs} \text{ is a variable in } f \in S\},$$
$$\hat{f}^h_U = (0, 0, 0)^T,$$
$$\hat{f}^h_D = (0, 0, 0)^T.$$
$$\hat{f}^\nu_E = \{(0, 0, p_{ze})^T \mid p_{ze} \text{ is a variable in } f \in E\},$$
$$\hat{f}^\nu_W = \{(0, 0, -p_{zw})^T \mid p_{zw} \text{ is a variable in } f \in W\},$$
$$\hat{f}^\nu_N = \{(0, -p_{zn}, 0)^T \mid p_{zn} \text{ is a variable in } f \in N\},$$
$$\hat{f}^\nu_S = \{(0, p_{zs}, 0)^T \mid p_{zs} \text{ is a variable in } f \in S\},$$
$$\hat{f}^\nu_U = \{(1, p_{yu}, -p_{xu})^T \mid p_{xu} \text{ and } p_{yu} \text{ are variables in } f \in U\},$$
$$\hat{f}^\nu_D = \{(-1, -p_{yd}, p_{xd})^T \mid p_{xd} \text{ and } p_{yd} \text{ are variables in } f \in D\}.$$

We define $\hat{e}$ to be the set of wrench points of a finger when it moves in the interior of a concave edge e. We call $\hat{e}$ the *edge wrench set of* e. Let η' and η'' be the normal directions of the faces incident to edge e, and let $int(e)$ denote the interior points of e. Then $\hat{e}$ is defined as follows.[3]

$$\hat{e} = \{(\eta, p_e \times \eta)^T \mid p_e \in int(e), \eta = \kappa_1\eta' + \kappa_2\eta'', 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1\}$$

The h and ν components $\hat{e}^h$ and $\hat{e}^\nu$ are similarly defined as $\hat{f}^h$ and $\hat{f}^\nu$. Before we define $\hat{e}$, $\hat{e}^h$ and $\hat{e}^\nu$ more formally, we need to introduce notations for the points on the boundary of $\hat{e}$. Without loss of generality, we take a concave edge e_{EN} from EN. A finger at position p on e_{EN} induces a set of lines of force, which is bounded by two face normals of the two incident faces from E and N. The wrench set induced by the finger at p is also bounded by two wrench points. We let $\hat{e}_{EN,E}$

[3]Note that η's in $\hat{e}$ are not unit vectors, when $\eta \neq \eta'$ and $\eta \neq \eta''$. But this does not matter, because only the direction matters, not the magnitude, to check whether a given set of wrench vectors positively spans wrench space. We chose these sets because it is easier to handle when the vectors are on the line segment connecting η' and η'' than when they are on a curve between η' and η''.

and $\hat{e}_{EN,N}$ denote the sets of boundary points of $\hat{e}_{EN}$, such that they correspond to the normals of the incident faces from E and from N respectively, for all position p in the interior of e_{EN}. We call $\hat{e}_{EN,E}$ and $\hat{e}_{EN,N}$ the boundary sets of e_{EN}. The following shows formal definition of $\hat{e}_{EN,E}$ and $\hat{e}_{EN,N}$ for $\hat{e}_{EN}$, and the corresponding sets for $\hat{e}_{SU}$ and $\hat{e}_{WD}$. Those from other families are defined similarly.

$$\begin{aligned}
\hat{e}_{EN,E} &= \{(\eta_E, p_e \times \eta_E)^T \mid \eta_E = (1,0,0)^T, p_e \in int(e_{EN})\} \\
&= \{(1,0,0,0,p_z,-p_y)^T \mid p_e = (p_x,p_y,p_z)^T \in int(e_{EN})\} \\
\hat{e}_{EN,N} &= \{(\eta_N, p_e \times \eta_N)^T \mid \eta_N = (0,1,0)^T, p_e \in int(e_{EN})\} \\
&= \{(0,1,0,-p_z,0,p_x)^T \mid p_e = (p_x,p_y,p_z)^T \in int(e_{EN})\} \\
\hat{e}_{WD,W} &= \{(\eta_W, p_e \times \eta_W)^T \mid \eta_W = (-1,0,0)^T, p_e \in int(e_{WD})\} \\
&= \{(-1,0,0,0,-p_z,p_y)^T \mid p_e = (p_x,p_y,p_z)^T \in int(e_{WD})\} \\
\hat{e}_{WD,D} &= \{(\eta_D, p_e \times \eta_D)^T \mid \eta_D = (0,0,-1)^T, p_e \in int(e_{WD})\} \\
&= \{(0,0,-1,-p_y,p_x,0)^T \mid p_e = (p_x,p_y,p_z)^T \in int(e_{WD})\} \\
\hat{e}_{SU,S} &= \{(\eta_S, p_e \times \eta_S)^T \mid \eta_S = (0,-1,0)^T, p_e \in int(e_{SU})\} \\
&= \{(0,-1,0,p_z,0,-p_x)^T \mid p_e = (p_x,p_y,p_z)^T \in int(e_{SU})\} \\
\hat{e}_{SU,U} &= \{(\eta_U, p_e \times \eta_U)^T \mid \eta_U = (0,0,1)^T, p_e \in int(e_{SU})\} \\
&= \{(0,0,1,p_y,-p_x,0)^T \mid p_e = (p_x,p_y,p_z)^T \in int(e_{SU})\}
\end{aligned}$$

The h and ν components $\hat{e}^h_{EN,E}$, $\hat{e}^h_{EN,N}$, $\hat{e}^\nu_{EN,E}$, $\hat{e}^\nu_{EN,N}$, $\hat{e}^\nu_{WU,W}$, $\hat{e}^\nu_{WU,U}$, $\hat{e}^\nu_{SD,S}$ and $\hat{e}^\nu_{SD,D}$ are defined as follows. Those for the other families can also be defined similarly.

$$\hat{e}^h_{EN,E} = \{(1,0,-p_y)^T \mid p_e = (p_x,p_y,p_z)^T \in int(e_{EN})\}$$

$$\hat{e}^h_{EN,N} = \{(0,1,p_x)^T \mid p_e = (p_x,p_y,p_z)^T \in int(e_{EN})\}$$

$$\hat{e}^h_{WU,W} = \{(-1,0,p_y)^T \mid p_e = (p_x,p_y,p_z)^T \in int(e_{WU})\}$$

$$\hat{e}^h_{WU,U} = \{(0,0,0)^T \mid p_e = (p_x,p_y,p_z)^T \in int(e_{WU})\}$$

$$\hat{e}^h_{SD,S} = \{(0,-1,-p_x)^T \mid p_e = (p_x,p_y,p_z)^T \in int(e_{SD})\}$$

$$\hat{e}^h_{SD,D} = \{(0,0,0)^T \mid p_e = (p_x,p_y,p_z)^T \in int(e_{SD})\}$$

$$\hat{e}^\nu_{EN,E} = \{(0,0,p_z)^T \mid p_e = (p_x,p_y,p_z)^T \in int(e_{EN})\}$$

$$\hat{e}^\nu_{EN,N} = \{(0,-p_z,0)^T \mid p_e = (p_x,p_y,p_z)^T \in int(e_{EN})\}$$

$$\hat{e}^\nu_{WU,W} = \{(0,0,-p_z)^T \mid p_e = (p_x,p_y,p_z)^T \in int(e_{WU})\}$$

$$\hat{e}^\nu_{WU,U} = \{(1,p_y,-p_x)^T \mid p_e = (p_x,p_y,p_z)^T \in int(e_{WU})\}$$

$$\hat{e}^\nu_{SD,S} = \{(0,p_{zs},0)^T \mid p_e = (p_x,p_y,p_z)^T \in int(e_{SD})\}$$

$$\hat{e}^\nu_{SD,D} = \{(-1,-p_y,p_x)^T \mid p_e = (p_x,p_y,p_z)^T \in int(e_{SD})\}$$

The edge wrench set $\hat{e}$ of e from each family can now be reformulated with the boundary sets as follows.

$$\begin{aligned}
\hat{e}_{EN} &= \{\kappa_1 w_E + \kappa_2 w_N \mid w_E \in \hat{e}_{EN,E}, w_N \in \hat{e}_{EN,N}, 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1\} \\
&= \{(\kappa_1, \kappa_2, 0, -\kappa_2 p_z, \kappa_1 p_z, -\kappa_1 p_y + \kappa_2 p_x)^T \mid 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1, \\
&\quad p_e = (p_x,p_y,p_z)^T \in int(e_{EN})\}
\end{aligned}$$

$$\begin{aligned}
\hat{e}_{WN} = &\ \{\kappa_1 w_W + \kappa_2 w_N \mid w_W \in \hat{e}_{WN,W}, w_N \in \hat{e}_{WN,N}, 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1\} \\
= &\ \{(-\kappa_1, \kappa_2, 0, -\kappa_2 p_z, -\kappa_1 p_z, \kappa_1 p_y + \kappa_2 p_x)^T \mid 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1, \\
& p_e = (p_x, p_y, p_z)^T \in int(e_{WN})\}
\end{aligned}$$

$$\begin{aligned}
\hat{e}_{ES} = &\ \{\kappa_1 w_E + \kappa_2 w_S \mid w_E \in \hat{e}_{ES,E}, w_S \in \hat{e}_{ES,S}, 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1\} \\
= &\ \{(\kappa_1, -\kappa_2, 0, \kappa_2 p_z, \kappa_1 p_z, -\kappa_1 p_y - \kappa_2 p_x)^T \mid 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1, \\
& p_e = (p_x, p_y, p_z)^T \in int(e_{ES})\}
\end{aligned}$$

$$\begin{aligned}
\hat{e}_{WS} = &\ \{\kappa_1 w_W + \kappa_2 w_S \mid w_W \in \hat{e}_{WS,W}, w_S \in \hat{e}_{WS,S}, 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1\} \\
= &\ \{(-\kappa_1, -\kappa_2, 0, \kappa_2 p_z, -\kappa_1 p_z, \kappa_1 p_y - \kappa_2 p_x)^T \mid 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1, \\
& p_e = (p_x, p_y, p_z)^T \in int(e_{WS})\}
\end{aligned}$$

$$\begin{aligned}
\hat{e}_{EU} = &\ \{\kappa_1 w_E + \kappa_2 w_U \mid w_E \in \hat{e}_{EU,E}, w_U \in \hat{e}_{EU,U}, 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1\} \\
= &\ \{(\kappa_1, 0, \kappa_2, \kappa_2 p_y, \kappa_1 p_z - \kappa_2 p_x, -\kappa_1 p_y)^T \mid 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1, \\
& p_e = (p_x, p_y, p_z)^T \in int(e_{EU})\}
\end{aligned}$$

$$\begin{aligned}
\hat{e}_{WU} = &\ \{\kappa_1 w_W + \kappa_2 w_U \mid w_W \in \hat{e}_{WU,W}, w_U \in \hat{e}_{WU,U}, 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1\} \\
= &\ \{(-\kappa_1, 0, \kappa_2, \kappa_2 p_y, -\kappa_1 p_z - \kappa_2 p_x, \kappa_1 p_y)^T \mid 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1, \\
& p_e = (p_x, p_y, p_z)^T \in int(e_{WU})\}
\end{aligned}$$

$$\begin{aligned}
\hat{e}_{NU} = &\ \{\kappa_1 w_N + \kappa_2 w_U \mid w_N \in \hat{e}_{NU,N}, w_U \in \hat{e}_{NU,U}, 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1\} \\
= &\ \{(0, \kappa_1, \kappa_2, -\kappa_1 p_z + \kappa_2 p_y, -\kappa_2 p_x, \kappa_1 p_x)^T \mid 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1, \\
& p_e = (p_x, p_y, p_z)^T \in int(e_{NU})\}
\end{aligned}$$

$$\begin{aligned}
\hat{e}_{SU} = &\ \{\kappa_1 w_S + \kappa_2 w_U \mid w_S \in \hat{e}_{SU,S}, w_U \in \hat{e}_{SU,U}, 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1\} \\
= &\ \{(0, -\kappa_1, \kappa_2, \kappa_1 p_z + \kappa_2 p_y, -\kappa_2 p_x, -\kappa_1 p_x)^T \mid 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1, \\
& p_e = (p_x, p_y, p_z)^T \in int(e_{SU})\}
\end{aligned}$$

$$\begin{aligned}
\hat{e}_{ED} = &\ \{\kappa_1 w_E + \kappa_2 w_D \mid w_E \in \hat{e}_{ED,E}, w_D \in \hat{e}_{ED,D}, 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1\} \\
= &\ \{(\kappa_1, 0, -\kappa_2, -\kappa_2 p_y, \kappa_1 p_z + \kappa_2 p_x, -\kappa_1 p_y)^T \mid 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1, \\
& p_e = (p_x, p_y, p_z)^T \in int(e_{ED})\}
\end{aligned}$$

$$\begin{aligned}
\hat{e}_{WD} = &\ \{\kappa_1 w_W + \kappa_2 w_D \mid w_W \in \hat{e}_{WD,W}, w_D \in \hat{e}_{WD,D}, 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1\} \\
= &\ \{(-\kappa_1, 0, -\kappa_2, -\kappa_2 p_y, -\kappa_1 p_z + \kappa_2 p_x, \kappa_1 p_y)^T \mid 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1, \\
& p_e = (p_x, p_y, p_z)^T \in int(e_{WD})\}
\end{aligned}$$

$$\begin{aligned}
\hat{e}_{ND} = &\ \{\kappa_1 w_N + \kappa_2 w_D \mid w_N \in \hat{e}_{ND,N}, w_D \in \hat{e}_{ND,D}, 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1\} \\
= &\ \{(0, \kappa_1, -\kappa_2, -\kappa_1 p_z - \kappa_2 p_y, \kappa_2 p_x, \kappa_1 p_x)^T \mid 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1, \\
& p_e = (p_x, p_y, p_z)^T \in int(e_{ND})\}
\end{aligned}$$

$$\begin{aligned}
\hat{e}_{SD} = &\ \{\kappa_1 w_S + \kappa_2 w_D \mid w_S \in \hat{e}_{SD,S}, w_D \in \hat{e}_{SD,D}, 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1\} \\
= &\ \{(0, -\kappa_1, -\kappa_2, \kappa_1 p_z - \kappa_2 p_y, \kappa_2 p_x, -\kappa_1 p_x)^T \mid 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1, \\
& p_e = (p_x, p_y, p_z)^T \in int(e_{SD})\}
\end{aligned}$$

The h and ν components $\hat{e}^h$ and $\hat{e}^\nu$ of an edge e can be reformulated with the h and ν components of the boundary sets as follows. Here we reformulate only $\hat{e}^h_{EN}$, $\hat{e}^h_{EU}$, $\hat{e}^\nu_{EN}$ and $\hat{e}^\nu_{EU}$. Those from other families can be reformulatd similarly.

$$\begin{aligned}
\hat{e}^h_{EN} = &\ \{\kappa_1 h_E + \kappa_2 h_N \mid h_E \in \hat{e}^h_{EN,E}, h_N \in \hat{e}^h_{EN,N}, 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1\} \\
= &\ \{(\kappa_1, \kappa_2, -\kappa_1 p_y + \kappa_2 p_x)^T \mid 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1, p_e = (p_x, p_y, p_z)^T \in int(e_{EN})\}
\end{aligned}$$

$$\begin{aligned}
\hat{e}^h_{EU} = &\ \{\kappa_1 h_E + \kappa_2 h_U \mid h_E \in \hat{e}^h_{EU,E}, h_U \in \hat{e}^h_{EU,U}, 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1\} \\
= &\ \{(\kappa_1, 0, -\kappa_1 p_y)^T \mid 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1, p_e = (p_x, p_y, p_z)^T \in int(e_{EU})\}
\end{aligned}$$

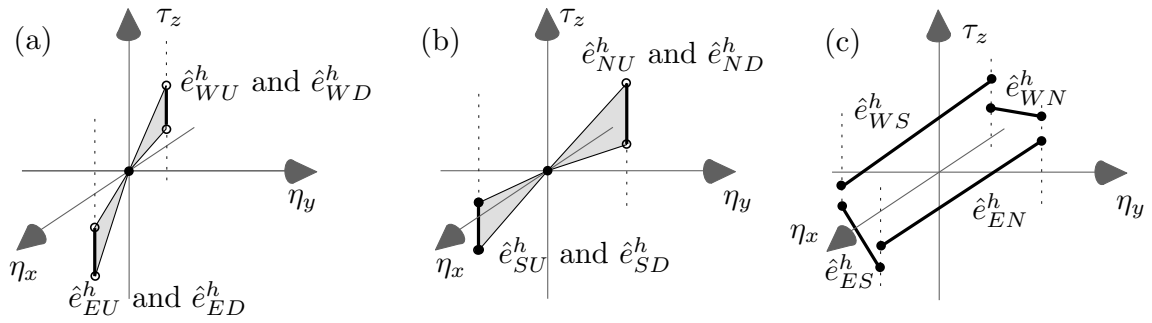


Figure 8.3: The shapes of $\hat{e}^h$ from each family.

$$\begin{aligned}\hat{e}^\nu_{EN} &= \{\kappa_1\nu_E + \kappa_2\nu_N \mid \nu_E \in \hat{e}^\nu_{EN,E}, \nu_N \in \hat{e}^\nu_{EN,N}, 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1\} \\ &= \{(0, -\kappa_2 p_z, \kappa_1 p_z)^T \mid 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1, p_e = (p_x, p_y, p_z)^T \in int(e_{EN})\} \\ \hat{e}^\nu_{EU} &= \{\kappa_1\nu_E + \kappa_2\nu_U \mid \nu_E \in \hat{e}^\nu_{EU,E}, \nu_U \in \hat{e}^\nu_{EU,U}, 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1\} \\ &= \{(\kappa_2, \kappa_2 p_y, \kappa_1 p_z - \kappa_2 p_x)^T \mid 0 \le \kappa_1, \kappa_2 \le 1, \kappa_1 + \kappa_2 = 1, p_e = (p_x, p_y, p_z)^T \in int(e_{EU})\}\end{aligned}$$

The shape of $\hat{e}^h$ from each family is as follows. The sets $\hat{e}^h_{EN}$, $\hat{e}^h_{WN}$, $\hat{e}^h_{ES}$ and $\hat{e}^h_{WS}$ are line segments with one endpoint lying on the lines $(\pm 1, 0, \tau_z)^T$, and the other endpoint on the lines $(0, \pm 1, \tau_z)^T$. The sets $\hat{e}^h_{EU}$, $\hat{e}^h_{WU}$, $\hat{e}^h_{NU}$, $\hat{e}^h_{SU}$, $\hat{e}^h_{ED}$, $\hat{e}^h_{WD}$, $\hat{e}^h_{ND}$ and $\hat{e}^h_{SD}$ are triangles that are incident to the origin in $\mathcal{H}$ space, because $\hat{f}^h_U$ and $\hat{f}^h_D$ are $(0, 0, 0)^T$. In fact, $\hat{e}^h_{EU}$, $\hat{e}^h_{ED}$ are triangles formed by the origin of $\mathcal{H}$ space and a line segment on the line $(1, 0, \tau_z)^T$. The sets $\hat{e}^h_{WU}$ and $\hat{e}^h_{WD}$ are triangles formed by the origin of $\mathcal{H}$ space and a line segment on the line $(-1, 0, \tau_z)^T$. The sets $\hat{e}^h_{NU}$ and $\hat{e}^h_{ND}$ are triangles formed by the origin and a line segment on the line $(0, 1, \tau_z)^T$, and those of $\hat{e}^h_{SU}$ and $\hat{e}^h_{SD}$ are also triangles formed by the origin and a line segment on the line $(0, -1, \tau_z)^T$. The endpoints of $\hat{e}^h_{EN}$, $\hat{e}^h_{WN}$, $\hat{e}^h_{ES}$ and $\hat{e}^h_{WS}$ are on the lines $(1, 0, \tau_z)^T$ (for E), $(-1, 0, \tau_z)^T$ (for W), $(0, 1, \tau_z)^T$ (for N) and $(0, -1, \tau_z)^T$ (for S). Figure 8.3 shows an edge wrench set from each family, when all position vectors have positive numbers only. Their shapes do not change much even when the position vectors have negative numbers as well as positive ones.

Now we define $\hat{v}$ to be the set of wrench points of a finger at a concave vertex v. We call $\hat{v}$ the *vertex wrench set of* v. Notice that a concave vertex is always incident to three faces: one from U or D, one from N or S, and one from E or W. Let η_1, η_2 and η_3 be the normal directions of the three incident faces of v, and let p_v be the position vector of v. Then $\hat{v}$ is defined as follows.[4]:

$$\hat{v} = \{(\eta, p_v \times \eta)^T \mid \eta = \mu_1\eta_1 + \mu_2\eta_2 + \mu_3\eta_3, 0 \le \mu_1, \mu_2, \mu_3 \le 1, \mu_1 + \mu_2 + \mu_3 = 1\}.$$

As in the case of edge wrench sets, we can reformulate vertex wrench sets in terms of the boundary sets. Without loss of generality, we take a concave vertex v_{ENN} from ENU. A finger at v_{ENU} induces a set of lines of force, which forms the convex hull of the three face normals of E, N and U. The convex hull of the wrench points for these three face normals is the wrench set induced by the finger at v_{ENU}. We let $\hat{v}_{ENU,E}$, $\hat{v}_{ENU,N}$ and $\hat{v}_{ENU,U}$ denote the wrench points for these three face normals, and call them the boundary vertices of $\hat{v}_{ENU}$. The following shows formal definitions of the boundary vertices, and their h and ν components for $\hat{v}_{ENU}$. Those from other families are defined similarly.

$$\begin{aligned}\hat{v}_{ENU,E} &= \{(\eta_E, p_e \times \eta_E)^T \mid \eta_E = (1, 0, 0)^T, p_v \text{ at } v_{ENU}\} \\ &= \{(1, 0, 0, 0, p_z, -p_y)^T \mid p_v = (p_x, p_y, p_z)^T\}\end{aligned}$$

[4]As in the case of edge wrench set, η's in $\hat{v}$ are not unit vectors, when $\eta \neq \eta_1$, $\eta \neq \eta_2$ and $\eta \neq \eta_3$. This does not matter, because only the direction matters, not the magnitude, to check whether a given set of wrench vectors positively spans wrench space. We chose these sets because it is easier to handle when the vectors are in the convex hull of η's ($\sum_i \mu_i = 1$) than when they are in a curved surface between η's ($\sum_i \mu_i^2 = 1$).

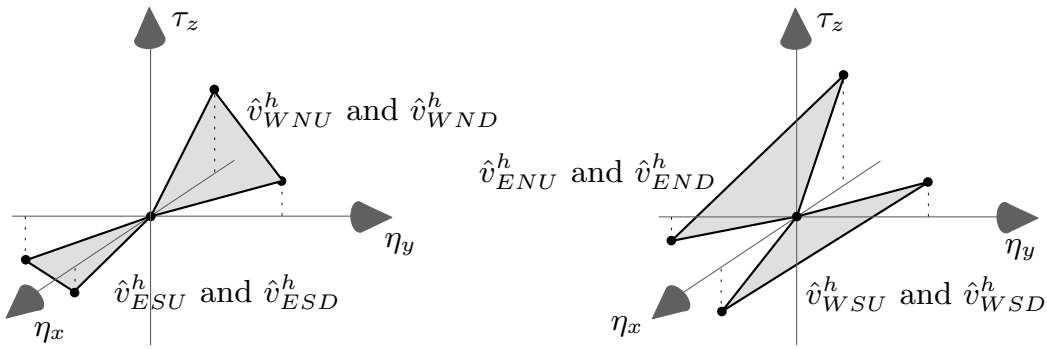


Figure 8.4: The shapes of $\hat{v}^h$ from each family, when the position vectors have positive numbers only.

$$\begin{aligned}\hat{v}_{ENU,N} = &\ \{(\eta_N, p_e \times \eta_N)^T \mid \eta_N = (0,1,0)^T, p_v \text{ at } v_{ENU}\} \\ = &\ \{(0,1,0,-p_z,0,p_x)^T \mid p_v = (p_x,p_y,p_z)^T\}\end{aligned}$$

$$\begin{aligned}\hat{v}_{ENU,U} = &\ \{(\eta_U, p_e \times \eta_U)^T \mid \eta_U = (0,0,1)^T, p_v \text{ at } v_{ENU}\} \\ = &\ \{(0,0,1,p_y,-p_x,0)^T \mid p_v = (p_x,p_y,p_z)^T\}\end{aligned}$$

$$\hat{v}^h_{ENU,E} = \{(1,0,-p_y)^T \mid p_v = (p_x,p_y,p_z)^T\}$$

$$\hat{v}^h_{ENU,N} = \{(0,1,p_x)^T \mid p_v = (p_x,p_y,p_z)^T\}$$

$$\hat{v}^h_{ENU,U} = \{(0,0,0)^T \mid p_v = (p_x,p_y,p_z)^T\}\}$$

$$\hat{v}^\nu_{ENU,E} = \{(0,0,p_z)^T \mid p_v = (p_x,p_y,p_z)^T\}$$

$$\hat{v}^\nu_{ENU,N} = \{(0,-p_z,0)^T \mid p_v = (p_x,p_y,p_z)^T\}$$

$$\hat{v}^\nu_{ENU,U} = \{(1,p_y,-p_x)^T \mid p_v = (p_x,p_y,p_z)^T\}\}$$

The sets $\hat{v}_{ENU}$, $\hat{v}^h_{ENU}$ and $\hat{v}^\nu_{ENU}$ can be reformulated with the boundary sets as follows. Those from other families can be reformulated similarly.

$$\begin{aligned}\hat{v}_{ENU} = &\ \{\mu_1 w_E + \mu_2 w_N + \mu_3 w_U \mid w_E = \hat{v}_{ENU,E}, w_N = \hat{v}_{ENU,N}, w_U = \hat{v}_{ENU,U}, \\ &\ 0 \le \mu_1,\mu_2,\mu_3 \le 1, \mu_1+\mu_2+\mu_3 = 1\} \\ = &\ \{(\mu_1,\mu_2,\mu_3,-\mu_2 p_z + \mu_3 p_y, \mu_1 p_z - \mu_3 p_x, -\mu_1 p_y + \mu_2 p_x)^T \mid \\ &\ 0 \le \mu_1,\mu_2,\mu_3 \le 1, \mu_1+\mu_2+\mu_3 = 1, p_v = (p_x,p_y,p_z)^T\}\end{aligned}$$

$$\begin{aligned}\hat{v}^h_{ENU} = &\ \{\mu_1 h_E + \mu_2 h_N + \mu_3 h_U \mid h_E = \hat{v}^h_{ENU,E}, h_N = \hat{v}^h_{ENU,N} h_U = \hat{v}^h_{ENU,U}, \\ &\ 0 \le \mu_1,\mu_2,\mu_3 \le 1, \mu_1+\mu_2+\mu_3 = 1\} \\ = &\ \{(\mu_1,\mu_2,-\mu_1 p_y + \mu_2 p_x)^T \mid \\ &\ 0 \le \mu_1,\mu_2,\mu_3 \le 1, \mu_1+\mu_2+\mu_3 = 1, p_v = (p_x,p_y,p_z)^T\}\end{aligned}$$

$$\begin{aligned}\hat{v}^\nu_{ENU} = &\ \{\mu_1 \nu_E + \mu_2 \nu_N + \mu_3 \nu_U \mid \nu_E = \hat{v}^\nu_{ENU,E}, \nu_N = \hat{v}^\nu_{ENU,N}, \nu_U = \hat{v}^\nu_{ENU,U}, \\ &\ 0 \le \mu_1,\mu_2,\mu_3 \le 1, \mu_1+\mu_2+\mu_3 = 1\} \\ = &\ \{(\mu_3,-\mu_2 p_z + \mu_3 p_y, \mu_1 p_z - \mu_3 p_x)^T \mid \\ &\ 0 \le \mu_1,\mu_2,\mu_3 \le 1, \mu_1+\mu_2+\mu_3 = 1, p_v = (p_x,p_y,p_z)^T\}\end{aligned}$$

Observe that $\hat{v}^h_{ENU}$ forms a triangle in $\mathcal{H}$ space. Since $\hat{v}^h_{ENU,U}$ is $(0,0,0)^T$, a vertex of the triangle is the origin in $\mathcal{H}$ space. Moreover, another vertex $\hat{v}^h_{ENU,E}$ is on the line $(1,0,\tau_z)^T$, and the third vertex $\hat{v}^h_{ENU,N}$ is on the line $(0,1,\tau_z)^T$. Those from other families are similar. See Figure 8.4.

8.1.2 The form closure condition and the projection scheme

The following theorem states a condition for a three-dimensional object to be in form closure, which is analogous to Theorem 2.1.

Theorem 8.1 *Given a set of κ (≥ 7) wrenches $w_1, w_2, \cdots, w_\kappa$ on a three-dimensional object P, the following three conditions are equivalent:*

(i) P is in form closure.

(ii) Any wrench w_F can be written as $-w_F = \lambda_1 w_1 + \cdots + \lambda_\kappa w_\kappa$, with $\lambda_i \geq 0$.

(iii) The origin O lies in the interior *of the convex hull of $w_1, w_2, \cdots, w_\kappa$.*

Theorem 8.1 basically states that P is in form closure if and only if the κ wrenches positively span wrench space. If we let w_F be a zero vector, Theorem 8.1 (ii) becomes an algebraic formulation of Theorem 8.1 (iii). In particular, when $\kappa = 7$, all λ_i's must be positive to make a zero vector, i.e. $\sum_{i=1}^{7} \lambda_i w_i = \vec{0}$ for $\lambda_i > 0$. Hence Theorem 8.1 (ii) and (iii) are algebraic and geometric formulations that the κ wrench vectors positively span six-dimensional wrench space. This is why κ is at least seven—the dimension plus one. Since a wrench is a six-dimensional description of a directed line in three-dimensional space, Theorem 8.1 (ii) and (iii) imply that any directed line in three-dimensional space can be represented by a linear combination of a given set of directed lines, whose wrenches positively span wrench space. The following lemma is another algebraic formulation of $\sum_{i=1}^{7} \lambda_i w_i = \vec{0}$ ($\lambda_i > 0$) for a rectilinear polyhedron. This is for the case of seven fingers, two of which lie on a face either from U or D. This lemma can be applied to the other cases when two fingers lie on a face from E, W N or S as well; rotate the polyhedron such that these families become U or D.

Lemma 8.2 *Given a set of seven wrenches $w_1, \cdots, w_7$ of a rectilinear polyhedron P, let w_1, w_2, w_3 and w_4 be the wrenches for vertical faces (from E, W, N and S), and w_5, w_6, w_7 be those for horizontal faces (from U and D). Let h_i and ν_i be the h and the ν components of w_i. Then the seven wrenches $w_1, \cdots, w_7$ achieve form closure if and only if they satisfy the following two conditions:*

1. *there exist $\alpha_i > 0$ such that $\sum_{i=1}^{4} \alpha_i h_i = \vec{0}$, and*
2. *there exist $\beta_i > 0$ such that $\sum_{i=1}^{4} \alpha_i \nu_i + \sum_{i=5}^{7} \beta_i \nu_i = \vec{0}$.*

Proof: Since $h_i \neq \vec{0}$ ($i = 1, 2, 3, 4$), there exist $\alpha_i > 0$ such that $\sum_{i=1}^{4} \alpha_i h_i = \vec{0}$. There also exist $\beta_i > 0$ such that $\sum_{i=1}^{4} \alpha_i \nu_i + \sum_{i=5}^{7} \beta_i \nu_i = \vec{0}$. Since $h_5 = h_6 = h_7 = \vec{0}$, it is straightforward to see that there exist $\lambda_i > 0$ such that $\sum_{i=1}^{7} \lambda_i w_i = \vec{0}$. (Set $\lambda_i = \alpha_i$ for $i = 1, 2, 3, 4$, and $\lambda_i = \beta_i$ for $i = 5, 6, 7$). □

The conditions in Lemma 8.2 can be verified by projecting the vectors on some screens. We define the screens Γ_h in $\mathcal{H}$ space and Γ_ν in $\mathcal{V}$ space as follows.

$$\begin{aligned} \Gamma_h := \Gamma_{h1} \cup \Gamma_{h2} &= \{(\eta_x, \eta_y, \tau_z)^T \mid \eta_x + \eta_y - 1 = 0, -1 - \varepsilon < \eta_x < 1 + \varepsilon, \tau_z \in \mathbb{R}\} \\ &\cup \{(\eta_x, \eta_y, \tau_z)^T \mid \eta_x - \eta_y - 1 = 0, -1 - \varepsilon < \eta_x < 1 + \varepsilon, \tau_z \in \mathbb{R}\} \end{aligned}$$

$$\begin{aligned} \Gamma_\nu := \Gamma_{\nu 1} \cup \Gamma_{\nu 2} &= \{(\eta_z, \tau_x, \tau_y)^T \mid \tau_x + \tau_y - 1 = 0, \eta_z \in \mathbb{R}\} \\ &\cup \{(\eta_z, \tau_x, \tau_y)^T \mid \tau_x - \tau_y - 1 = 0, \eta_z \in \mathbb{R}\} \end{aligned}$$

In the definition of Γ_h, ε is an arbitrarily small positive number. Figure 8.5 shows a top view of Γ_h and Γ_ν. The planes of Γ_h are extended by ε so that the lines $(1, 0, \tau_z)^T$ and $(0, 1, \tau_z)^T$ are

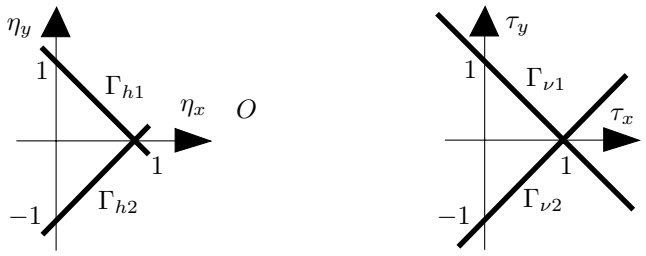


Figure 8.5: A top view of the screens Γ_h and Γ_ν.

completely contained in Γ_{h1}, and the lines $(1, 0, \tau_z)^T$ and $(0, -1, \tau_z)^T$ are completely contained in Γ_{h2}.

Remember that we place the planes $\eta_x\eta_y$ and $\tau_x\tau_y$ horizontally in $\mathcal{H}$ and $\mathcal{V}$ space respectively. The intersections of Γ_h and $\tau_z = 0$, and Γ_ν and $\eta_z = 0$ are horizontal lines on Γ_h and Γ_ν; we call them $\tau_z = 0$ line and $\eta_z = 0$ line respectively. We let $(0, 1)$, $(1, 0)$ and $(0, -1)$ lines on Γ_h denote the lines $(0, 1, \tau_z)^T$, $(1, 0, \tau_z)^T$ and $(0, -1, \tau_z)^T$ on Γ_h respectively. Likewise, we let $(0, 1)$, $(1, 0)$ and $(0, -1)$ lines on Γ_ν denote the lines $(\eta_z, 0, 1)^T$, $(\eta_z, 1, 0)^T$ and $(\eta_z, 0, -1)^T$ on Γ_ν respectively.

The projection scheme is the same as in Chapter 3. We project a vector $\omega \neq O$ onto a plane Γ as follows[5] : Consider the line ℓ through ω and the origin O, and let $\pi(\omega) := \ell \cap \Gamma$. The origin does not have any projection on Γ in our projection scheme. A non-vertical line intersecting a plane has two cases: (i) O lies between ω and $\pi(\omega)$, and (ii) O lies on the left or on the right of ω and $\pi(\omega)$ on ℓ. To visually distinguish these two kinds of projections, we color $\pi(\omega)$ *red* when O lies between ω and $\pi(\omega)$, and we color $\pi(\omega)$ *blue* when O lies on the left or on the right of ω and $\pi(\omega)$ on ℓ.

To check whether a given set of vectors positively span three-dimensional space, we use the following lemma.[6]

Lemma 8.3 *Given a set of κ (≥ 4) vectors in $\mathbb{R}^3$ $\omega_1, \omega_2, \cdots, \omega_\kappa$, the convex hull of $\omega_1, \omega_2, \cdots, \omega_\kappa$ contains the origin in the interior, if and only if the (blue) convex hull of the blue $\pi(\omega_i)$ intersects the (red) convex hull of the red $\pi(\omega_j)$ in the interior on the screen.*

Proof: Without loss of generality, let the projections of $\omega_1, \cdots, \omega_j$ be blue, and let the projections of $\omega_{j+1}, \cdots, \omega_\kappa$ be red. We show the "if" direction first. Let p_1 and p_2 be the red and blue points in the interior of the red and blue intersection region. From the construction, $p_1 = -\mu p_2$ for $\mu > 0$, and the interior points p_1 and p_2 can be represented as follows: $p_1 = \sum_{i=1}^{j} \lambda_i\omega_i$ and $p_2 = \sum_{i=(j+1)}^{\kappa} \lambda_i\omega_i$, where $\lambda_i \geq 0$. Observe that there are at least four non-zero λ_i's, because p_1 and p_2 are in the *interior* of the red and blue intersection region, and we need at least four points to have p_1 and p_2 in the interior of the red convex hull and the blue convex hull. When we combine these three equations, we get $\sum_{i=1}^{\kappa} \lambda_i\omega_i = \vec{0}$, where $\lambda_i \geq 0$ and at least four λ_i's are non-zero. (When $\kappa = 4$, all λ_i's are positive.) This means that the convex hull of the κ vectors contains the origin, according to Theorem 8.1.

The "only if" direction: From the assumption, there exist $\lambda_i \geq 0$ such that there are at least four non-zero λ_i's, and $\sum_{i=1}^{j} \lambda_i\omega_i = -\sum_{i=j+1}^{\kappa} \lambda_i\omega_i$. Observe that $\sum_{i=1}^{j} \lambda_i\omega_i$ is in the convex hull of j points $\omega_1, \cdots, \omega_j$, and $\sum_{i=j+1}^{\kappa} \lambda_i\omega_i$ is in the convex hull of $\omega_{j+1}, \cdots, \omega_\kappa$. The equation $\sum_{i=1}^{j} \lambda_i\omega_i = -\sum_{i=j+1}^{\kappa} \lambda_i\omega_i$ implies that their projections coincides on the plane, and their colors differ because the origin lies between the two points $\sum_{i=1}^{j} \lambda_i\omega_i$ and $\sum_{i=j+1}^{\kappa} \lambda_i\omega_i$. □

[5]This projection scheme is to find a set of points whose convex hull contains the origin in the interior. It is thus meaningless that the origin is in the point set. This justifies that we do not define the projection of the origin.

[6]This is another formulation of Lemma 2.7.

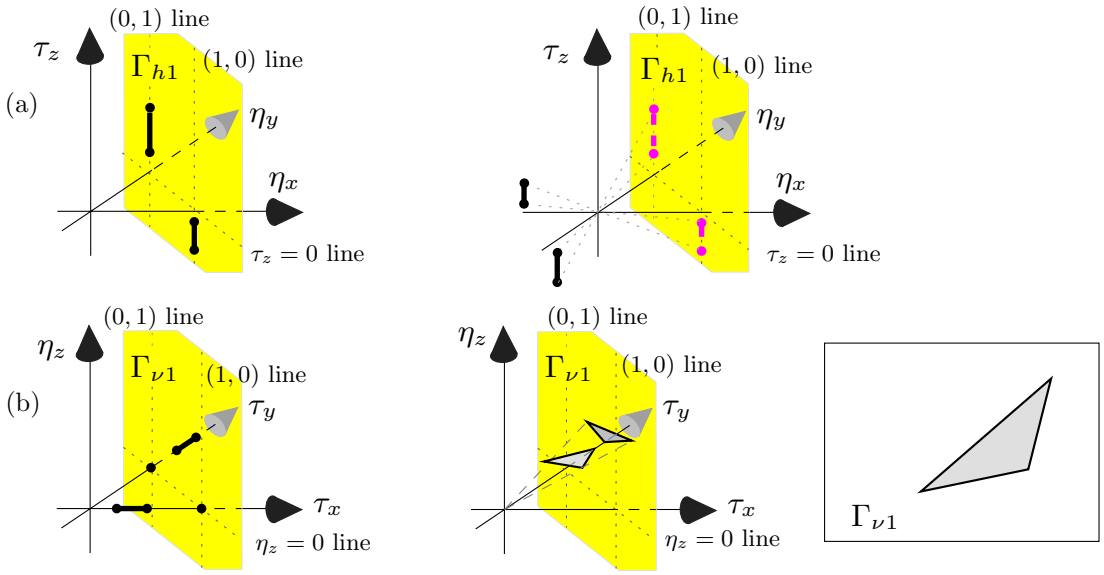


Figure 8.6: (a) The projections of $\hat{f}^h$ and (b) $\hat{f}^\nu$ from each family.

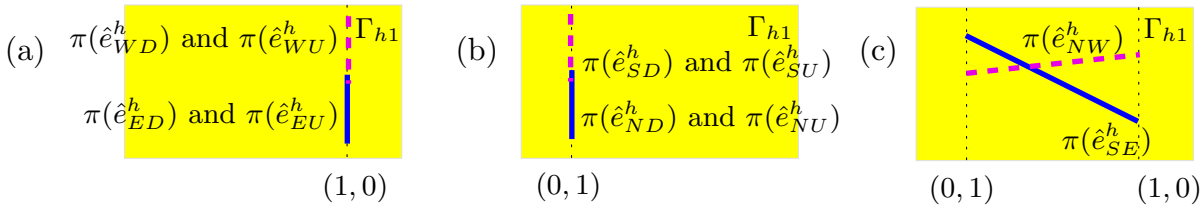


Figure 8.7: The projections of $\hat{e}^h$ from each family.

Let us describe the shapes of $\pi(\hat{f}^h)$, $\pi(\hat{e}^h)$ and $\pi(\hat{v}^h)$ on Γ_h, and $\pi(\hat{f}^\nu)$ on Γ_ν. The shapes of $\hat{f}^h_E$, $\hat{f}^h_W$, $\hat{f}^h_N$ and $\hat{f}^h_S$ are vertical line segments in $\mathcal{H}$ space, thus the projections are also vertical line segments on Γ_h. Note that $\pi(\hat{f}^h_U)$ and $\pi(\hat{f}^h_D)$ do not appear on Γ_h, because they are at the origin in $\mathcal{H}$ space. (The projection of the origin is not defined.) The sets $\hat{f}^\nu_E$, $\hat{f}^\nu_W$, $\hat{f}^\nu_N$ and $\hat{f}^\nu_S$ are line segments on τ_x and τ_y axes, thus their projections are points on Γ_ν. More precisely, they are $(0,1,0)^T$ (from N and S), or $(0,0,\pm 1)$ (from E and W). The projections of $\hat{f}^\nu_U$ and $\hat{f}^\nu_D$ are triangles on Γ_ν. See Figure 8.6.

The shapes of $\hat{e}^h_{EU}$, $\hat{e}^h_{WU}$, $\hat{e}^h_{NU}$, $\hat{e}^h_{SU}$, $\hat{e}^h_{ED}$, $\hat{e}^h_{WD}$, $\hat{e}^h_{ND}$ and $\hat{e}^h_{SD}$ are triangles with the origin as one of their vertices, and the other two on $(\pm 1, 0, \tau_z)^T$ line and on $(0, \pm 1, \tau_z)^T$ line. The projections of these triangles are vertical line segments on Γ_h, especially on $(1,0)$ line and $(0,\pm 1)$ line. See Figure 8.7 (a) and (b). The h components $\hat{e}^h_{EN}$, $\hat{e}^h_{WN}$, $\hat{e}^h_{ES}$ and $\hat{e}^h_{WS}$ are line segments whose endpoints lie on the lines $(\pm 1, 0, \tau_z)^T$ and $(0, \pm 1, \tau_z)^T$, so their projections are line segments whose endpoints lie on $(1,0)$ and $(0,\pm 1)$ lines on Γ_h. See Figure 8.7 (c).

Remember that $\hat{v}^h$ is also a triangle in $\mathcal{H}$ space with the origin as a vertex, and with the other two vertices on the lines $(\pm 1, 0, \tau_z)^T$ or $(0, \pm 1, \tau_z)^T$. Thus the projection is a line segment one endpoint of which is on $(1,0)$ line, and the other endpoint of which is on $(0,\pm 1)$ line on Γ_ν. See Figure 8.8.

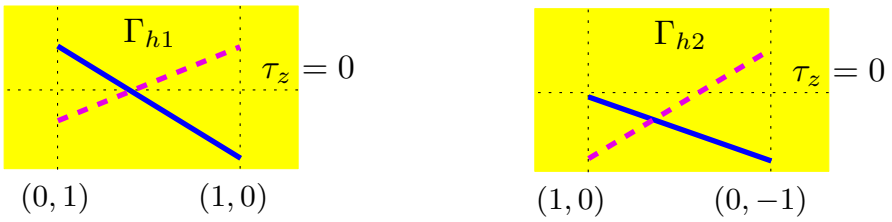


Figure 8.8: The projections of $\hat{v}^h$ from each family.

8.1.3 Intersection search algorithms

To compute all red and blue intersections, we use the following data structures: *segment intersection search structure*, *triangle search structure* and *half-plane search structure*. Section 2.4.2 has more detailed explanations about segment intersection search structure and triangle search structure. Throughout this chapter, we let k denote the output size of one query, and let K denote the total output size.

Sometimes, we wish to report points in a half plane or in the intersection region of two half planes. To report points in a half plane, we use the half-plane search structure by Chazel et al. [19]. The building time of the structure on q points is $O(q \log q)$, and the query time is $O(\log q + k)$.

To report points in the intersection region of two half planes, we use an order 2 tree. This stores q points in an order 2 tree in $O(q^2)$ time[7], as explained in Section 2.4.2: each point is stored twice, one in the first level, another in the second level. We query the tree with one half plane on one level, and with the other half plane on the other level, in $O(\log^2 q + k)$ time.

8.2 Computing all form-closure grasps of a rectilinear polyhedron

This section proposes algorithms to report all combinations of faces, concave edges and concave vertices of P that allow form-closure grasps with at most seven frictionless point fingers. We let $\mathcal{A}_{total}$ denote all such sets. Let C denote the combination of faces, concave edges and concave vertices that we consider. The combination C in Section 8.2.1 is seven faces. In Section 8.2.2, C is a combination of faces and concave edges; more precisely, we consider one edge and five faces, two edges and three faces, and three edges and one face. In Section 8.2.3, C is a combination of faces and concave vertices; more precisely, we consider one vertex and four faces, and two vertices and one face. In Section 8.2.2, C is one vertex, one edge and two faces. Note that each of these combinations involves seven face normals; seven is necessary to span six-dimensional wrench space positively, thus to achieve form-closure of a three-dimensional object.

Each of the algorithms proposed in the following sections reports a subset $\mathcal{A}$ of $\mathcal{A}_{total}$, such that each of $\mathcal{A}$ involves two face normals of a family of U or D. In other words, each set of $\mathcal{A}$ of combination C has a form-closure grasp, and it involves two face normals of a family of U or D. The reason is that the algorithms are based on Lemma 8.2, which imposes the additional condition about two face normals of a family of U or D. Note that the face normals can be induced by the fingers on edges or at vertices.

We can report $\mathcal{A}_{total}$ by reporting $\mathcal{A}$ of a polyhedron P and appropriately rotated P. The form-closure grasps involving two face normals of a family of E, W, N or S can be reported by the corresponding algorithm applied to the rotated polyhedron P. When the set induces two face normals of a family of E or W, we rotate P such that E becomes D. When the set involves two

[7] We set $t = \log q$; t is the parameter in Section 2.4.2.

face normals of a family of N or S, we rotate P such that N becomes D. The rotations that we use are the following two:

1. Rot_1: $N \to N$, $S \to S$, $E \to D$, $W \to U$, $U \to E$ and $D \to W$,
2. Rot_2: $N \to D$, $S \to U$, $E \to E$, $W \to W$, $U \to N$ and $D \to S$.

Now we give the outline of the algorithms. We report $\mathcal{A}$ by filtering all candidates of combination C twice. Let combination C' be a subset of C, such that a set of C' are necessary and sufficient to have a set of points that satisfy the first condition of Lemma 8.2. We also let $\mathcal{A}'$ denote all sets of combination C', such that each of these has a set of points that satisfy the first condition of Lemma 8.2. In phase I, the algorithm finds $\mathcal{A}'$. The time complexities of the algorithms in phase I are sensitive to K', which is the cardinality of $\mathcal{A}'$. In phase II, the algorithm computes the remaining faces that has a set of points satisfying the second condition of Lemma 8.2 together with one of $\mathcal{A}'$. These faces with $\mathcal{A}'$ form $\mathcal{A}$. Phase II of all algorithms except one[8] have time complexities sensitive to K, which is the cardinality of $\mathcal{A}$.

8.2.1 Seven faces

We wish to report all sets of seven faces that yield form-closure grasps with seven frictionless point fingers. Let a rectilinear polyhedron P have n (triangulated) faces. We pick any form-closure grasps on seven faces. If this set involves two face normals of a family of U or D, it is in $\mathcal{A}$. If this set involves two face normals of a family of E, W, N or S, we rotate P with Rot_1 or Rot_2 accordingly. It is straightforward to see that the rotated set is in $\mathcal{A}$, i.e. it has two face normals of a family of U or D.

In phase I, we report $\mathcal{A}'$. Four faces are necessary and sufficient to have a set of points that satisfy the first condition of Lemma 8.2; C' is four faces. Note that $\hat{f}^h$ of a vertical face f corresponds to an edge wrench set of a polygon. Hence in phase I, the approach in Section 3.3.1 can report $\mathcal{A}'$ in $O(n \log n + K')$ time; P has $O(n)$ faces.

We pick a face quadruple from $\mathcal{A}'$. This quadruple (f_1, f_2, f_3, f_4) has a range of coefficients $\alpha_i > 0$ for each of their $h_1 \in \hat{f}_1^h$, $h_2 \in \hat{f}_2^h$, $h_3 \in \hat{f}_3^h$ and $h_4 \in \hat{f}_4^h$, such that $\sum_{i=1}^4 \alpha_i h_i = \vec{0}$. The set $\sum_{i=1}^4 \alpha_i \nu_i$ with $\nu_i \in \hat{f}_i^\nu$ forms a polygon Δ of the following form:

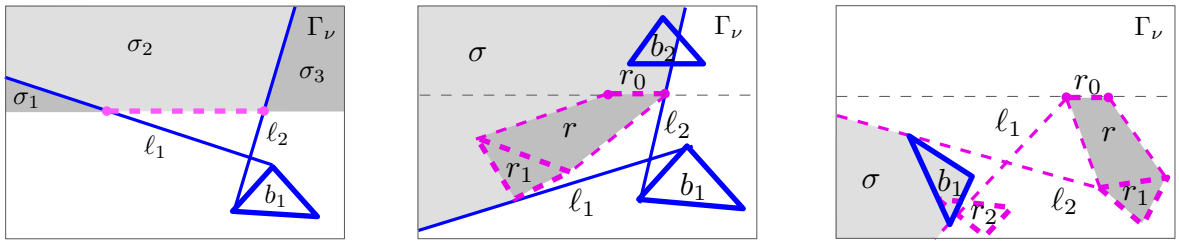
$$\Delta := \{\alpha_1\nu_1 + \alpha_2\nu_2 + \alpha_3\nu_3 + \alpha_4\nu_4 \mid \nu_1 \in \hat{f}_1^\nu, \nu_2 \in \hat{f}_2^\nu, \nu_3 \in \hat{f}_3^\nu, \nu_4 \in \hat{f}_4^\nu\}.$$

Plugging in the vectors show in Section 8.1.1 produces a vector $(0, -\alpha_1 p_{z1} + \alpha_3 p_{z3}, \alpha_2 p_{z2} - \alpha_4 p_{z4})^T$, where $p_i = (p_{xi}, p_{yi}, p_{zi})^T$ is the position vector of a finger on f_i. We get $\alpha_1\nu_1 + \alpha_2\nu_2 + \alpha_3\nu_3 + \alpha_4\nu_4$, because this is an equation for a point in the convex hull of ν_1, ν_2, ν_3 and ν_4. A set of fixed position p_i ($i = 1, 2, 3, 4$) determines the values of α_1, α_2, α_3 and α_4,[9] and the position vectors are independent, thus Δ forms a polygon on the plane $\eta_z = 0$ in $\mathcal{V}$ space. The projection of Δ is a part of the (horizontal) line segment connecting two points $(0, 1, 0)^T$ and $(0, 0, \pm 1)^T$ on Γ_ν. We first look at the case where $\pi(\Delta)$ is red. We let r_0 denote the red $\pi(\Delta)$ for convenience. The case where $\pi(\Delta)$ is blue can be handled in the same way as in the case when $\pi(\Delta)$ is red.

In phase II, we wish to report all blue triangle triples from $\pi(\hat{f}_U^\nu)$ and $\pi(\hat{f}_D^\nu)$ that has a set of points satisfying the second condition of Lemma 8.2 together with each set of $\mathcal{A}'$. Three points from three triangles of $\hat{f}_U^\nu$ and $\hat{f}_D^\nu$ positively span $\mathcal{V}$ space with a point of Δ, if and only if one of

[8] It is the algorithm for the case of three edges and a face. It is sensitive to K' only.

[9] The coefficients α_1, α_2, α_3, α_4 should be computed in $\mathcal{H}$ space originally. In our setting of Γ_h, however, $\hat{f}^h$'s *are* on screen Γ_h, thus the convex hull of two points from $\hat{f}^h$ is also on Γ_h. Therefore, we can compute these coefficients α_1, α_2, α_3, α_4 directly on Γ_h.



(i) (ii) (iii)

Figure 8.9: The tangent lines ℓ_1 and ℓ_2 of r_0 and b_1, and their regions in three cases (i), (ii) and (iii).

the following holds on Γ_ν: (i) the convex hull of three blue triangles intersects r_0 in the interior, (ii) the convex hull of two blue triangles intersects that of a red triangle and r_0 in the interior, and (iii) a blue triangle intersects the convex hull of two red triangles and r_0 in the interior. We enumerate all red and blue triangle triples that belong to one of these three cases one by one.

We enumerate all blue triangle triples whose convex hull intersects r_0 in the interior as follows. We first look at the cases when one blue triangle of $\pi(\hat{f}_D^\nu)$ and two blue triangles of $\pi(\hat{f}_U^\nu)$. The case when one blue triangle of $\pi(\hat{f}_U^\nu)$ and two blue triangles of $\pi(\hat{f}_D^\nu)$ can be handled in a similar way. We pick a blue triangle b_1 of $\pi(\hat{f}_D^\nu)$ for example, and compute two tangent lines ℓ_1 and ℓ_2 which separates r_0 and b_1; ℓ_1 touches the left endpoint of r_0, and ℓ_2 touches the right endpoint of r_0. Remember that the blue $\pi(\hat{f}_D^\nu)$ and r_0 do not intersect, because blue $\pi(\hat{f}_D^\nu)$ lies in the region of $\eta_z < 0$, and r_0 lies on the line $\eta_z = 0$. Also observe that all blue triangles $\pi(\hat{f}_U^\nu)$ lie in the region $\eta_z > 0$. Two lines ℓ_1 and ℓ_2 divide the region of $\eta_z > 0$ into three regions. We let σ_1, σ_2 and σ_3 denote the *interior* of these regions from left to right. See Figure 8.9 (i).

Let b_2 be a blue triangle of $\pi(\hat{f}_U^\nu)$ that intersects σ_2. Then (b_1, b_2, b_3) gives a face triple that positively span $\mathcal{V}$ space with the $\hat{f}^\nu$ quadruple for r_0, where b_3 is any blue triangle in $\pi(\hat{f}_U^\nu)$ or $\pi(\hat{f}_D^\nu)$—including b_1 and b_2 itself. Note that σ_2 is bounded by three half-planes: ℓ_1, ℓ_2 and the line $\eta_z = 0$. Since $\pi(\hat{f}_U^\nu)$ and $\pi(\hat{f}_D^\nu)$ lie on one side of the line $\eta_z = 0$, for each of $\pi(\hat{f}_U^\nu)$ and $\pi(\hat{f}_D^\nu)$, σ_2 can be considered to be the intersection of two half planes. Thus the blue triangles of $\pi(\hat{f}_U^\nu)$ intersecting σ_2 can be reported by an order 2 tree described in Section 8.1.3 and a segment intersection search structure. There are $O(n)$ blue triangles of $\pi(\hat{f}_U^\nu)$ and $\pi(\hat{f}_D^\nu)$, so the preprocessing times are $O(n^2)$ and $O(n^2 \log^2 n)$ respectively, and the query times are $O(\log^2 n + k)$ and $O(\log^4 n + k)$ respectively. Each of K' sets induces r_0, and there are $O(n)$ blue triangles of $\pi(\hat{f}_D^\nu)$ and $\pi(\hat{f}_U^\nu)$, thus such blue triangle triples can be reported in $O(n^2 \log^2 n + nK' \log^4 n + k)$ time.

If b_2 intersects σ_1, and b_3 intersects σ_3, the corresponding faces of (b_1, b_2, b_3) have three points that span $\mathcal{V}$ space positively with a point of Δ. A triangle intersects σ_1 (or σ_3), if and only if a vertex of the triangle lies in σ_1 (or σ_3). We store the vertices of the $O(n)$ blue triangles in a half-plane search structure [19] in $O(n \log n)$ time. The points lying in σ_1 and those in σ_3 can be identified in $O(\log n + k)$ time. Then we report every pair (b_2, b_3), where b_2 intersects σ_1 and b_3 intersects σ_3. Each of K' sets induces r_0, and there are $O(n)$ blue triangles of $\pi(\hat{f}_D^\nu)$ and $\pi(\hat{f}_U^\nu)$, thus all K blue triangle triples that intersect r_0 can be reported in $O(nK' \log n + K)$ time.

Now we look at case (ii). We wish to report all triples of a red triangle and two blue triangles, such that the convex hull of two blue triangles intersects that of r_0 and a red triangle in the interior. We pick one red triangle r_1 of $\pi(\hat{f}_U^\nu)$ or $\pi(\hat{f}_D^\nu)$, and compute the convex hull r of r_1 and r_0. We also pick a blue triangle b_1 from $\pi(\hat{f}_U^\nu)$ or $\pi(\hat{f}_D^\nu)$. If r and b_1 intersect each other in the interior,

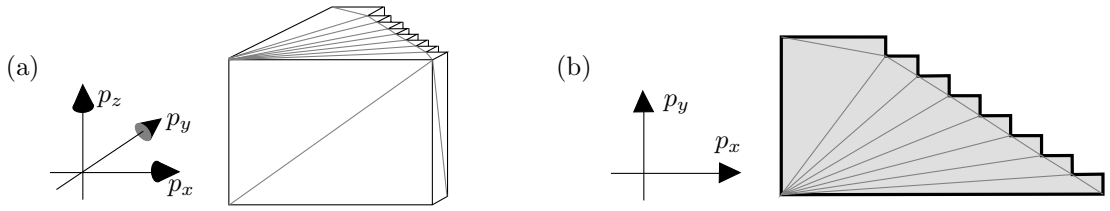


Figure 8.10: A polyhedron with $\Theta(n^2)$ face quadruples that positively span $\mathcal{H}$ space, and $\Theta(n^5)$ sets of seven faces that allow form-closure grasps with seven frictionless point fingers.

then we are done; two points on b_1 will positively span the space with three points from r_0 and r_1. Hence we focus on the case when r and b_1 do not intersect each other in the interior.

We compute the tangent lines ℓ_1 and ℓ_2 of b_1 and r such that ℓ_1 and ℓ_2 separate b_1 and r. Let σ denote the region containing the interior of r, which is bounded by ℓ_1, ℓ_2 and some edges of r. See Figure 8.9 (ii). Note that σ does not include its boundaries. Observe that blue triangles of $\pi(\hat{f}_D^\nu)$ (or $\pi(\hat{f}_U^\nu)$) intersect at most three boundary line segments of σ, because blue triangles of $\pi(\hat{f}_D^\nu)$ ($\pi(\hat{f}_U^\nu)$) lie below (above) the line $\eta_z = 0$. See Figure 8.9 (ii). Hence, we use the triangle search structure to report all k blue triangles intersecting σ. There are $O(n)$ blue triangles of $\pi(\hat{f}_U^\nu)$ and $\pi(\hat{f}_D^\nu)$, so the preprocessing time is $O(n^2 \log n)$, and the query time is $O(\log^3 n + k)$.

We repeat this process for each pair of red and blue triangles of $\pi(\hat{f}_U^\nu)$ and/or $\pi(\hat{f}_D^\nu)$. There are $O(n)$ red and blue triangles of $\pi(\hat{f}_U^\nu)$ and $\pi(\hat{f}_D^\nu)$, so there are $O(n^2)$ red and blue triangle pairs. The total time complexity of case (ii) is thus $O(n^2 K' \log^3 n + K)$.

Now we look at case (iii). We wish to report all triples of a blue triangle and two red triangles from $\pi(\hat{f}_U^\nu)$ and/or $\pi(\hat{f}_D^\nu)$, such that the convex hull of r_0 and the two red triangles intersects the blue triangle in the interior. We pick a red triangle r_1 from red $\pi(\hat{f}_U^\nu)$ or $\pi(\hat{f}_D^\nu)$, and compute the convex hull r of r_1 and r_0. We also pick a blue triangle b_1 from blue $\pi(\hat{f}_U^\nu)$ or $\pi(\hat{f}_D^\nu)$. Then we compute the tangent lines ℓ_1 and ℓ_2 of b_1 and r, such that ℓ_1 and ℓ_2 separate b_1 and r. Let σ denote the region bounded by ℓ_1, ℓ_2, and some edges of b_1[10], such that σ contains the interior of b_1. See Figure 8.9 (iii). Here as well, σ does not include its boundaries. Note that red triangles of $\pi(\hat{f}_U^\nu)$ lie in the region of $\eta_z < 0$, and red triangles of $\pi(\hat{f}_D^\nu)$ in the region of $\eta_z > 0$. This implies that red triangles of $\pi(\hat{f}_U^\nu)$ or $\pi(\hat{f}_D^\nu)$ intersecting σ lie in the region bounded by at most three lines. We use a triangle search structure to store the vertices of $O(n)$ red triangles of $\pi(\hat{f}_U^\nu)$ and $\pi(\hat{f}_D^\nu)$—the building time is $O(n^2 \log n)$. We also use a segment intersection search structure to store the edges of $O(n)$ red triangles—the construction time is $O(n^2 \log^2 n)$. We report all k red points in σ in $O(\log^3 n + k)$ time, and all k red segments intersecting σ in $O(\log^4 n + k)$ time.

We repeat this process for each of blue triangles of $\pi(\hat{f}_U^\nu)$ and $\pi(\hat{f}_D^\nu)$ as well. There are $O(n)$ blue triangles of $\pi(\hat{f}_U^\nu)$ and $\pi(\hat{f}_D^\nu)$, and each of $\mathcal{A}'$ induces r_0—remember that $K' = |\mathcal{A}'|$. Therefore, all K triples of a blue triangle and two red triangles of case (iii) can be reported in $O(n^2 K' \log^4 n + K)$ time.

Theorem 8.4 *All K sets of six to seven faces of P that yield form-closure grasps with seven frictionless point fingers can be enumerated in $O(n^2 K' \log^4 n + K)$ time, where $K' = |\mathcal{A}|$.*

Note that K' is $O(n^4)$, and K is $O(n^7)$. The lower bound of K' is $\Omega(n^2)$, and the lower bound of K is $\Omega(n^5)$—K' is $\Omega(n^2)$ and three fingers are on three faces from U and D. (The proof for the lower bound for K' can be found in Lemma 4.6 in [91].) A polyhedron can have

[10] At most two edges of b_1 bound σ.

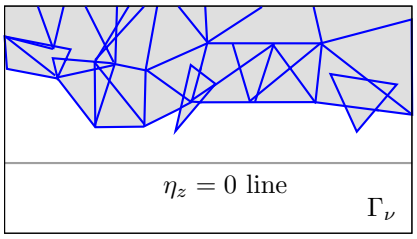


Figure 8.11: The blue projection of $\hat{f}_U^\nu$. They can overlap because the projections of the faces from U on the planes $\eta_z = \pm 1$ can overlap. The blue projection of $\hat{f}_D^\nu$ is the same, except that they lie in the region of $\eta_z < 0$.

one rectilinear face in U and another rectilinear face in D, but the number of vertices of these faces will be $O(n)$, thus the polyhedron has $O(n)$ triangulated faces in U and D. A polyhedron with $K' = |\mathcal{A}'| = \Omega(n^2)$ has $\Omega(n^5)$ sets of seven faces that allow form-closure grasps with seven fingers—see in Figure 8.10. The following lemma justifies the approach of computing all face quadruples in $\mathcal{A}$, and then computing the remaining face triples for each of $\mathcal{A}'$.

Lemma 8.5 $K = \Omega(n^2 K')$.

Proof: We first show that each of $\mathcal{A}'$ has at least one face triple from $\pi(\hat{f}_U^\nu)$ and $\pi(\hat{f}_D^\nu)$, such that they together have a set of points satisfying the second condition of Lemma 8.2. Let (f_1, f_2, f_3, f_4) be a quadruple of $\mathcal{A}'$. It is enough to show that $\pi(\Delta)$ for (f_1, f_2, f_3, f_4) has at least one triple of red or blue triangles that has a red and blue intersection with $\pi(\Delta)$. Without loss of generality, assume that $\pi(\Delta)$ is red. We show that there exists a triple of blue triangles whose convex hull intersects $\pi(\Delta)$ in the interior.

First we look at the shapes of the blue triangles of $\hat{f}_U^\nu$ and $\hat{f}_D^\nu$. If we project the faces from U (D) on the xy plane, the union of these projections form a rectilinear *simple* polygon. Remember that the origin of the object space is *in* the simple polygon. We rotate the projections from U (D) by 90 degrees, and place them on the plane $\eta_z = 1$ ($\eta_z = -1$) in $\mathcal{V}$ space. Note that η_z axis still penetrates these simple polygons on the planes $\eta_z = \pm 1$. This implies that on Γ_ν, any non-horizontal line will intersect the red and blue triangles of $\pi(\hat{f}_U^\nu)$ and $\pi(\hat{f}_D^\nu)$. Figure 8.11 shows an example of the union of blue triangles of $\pi(\hat{f}_U^\nu)$. The blue triangles of $\pi(\hat{f}_D^\nu)$ look similar in the region $\eta_z < 0$.[11] Therefore, there exists a blue triangle b_2 in $\pi(\hat{f}_U^\nu)$ that intersects the region σ_2, which is induced by $\pi(\Delta)$ and a blue triangle b_1 in $\pi(\hat{f}_D^\nu)$. In fact, there is a blue triangle in $\pi(\hat{f}_U^\nu)$ intersecting σ_2 for any blue triangle b_1 in $\pi(\hat{f}_D^\nu)$, thus we have $O(n)$ triples of blue triangles (b_1, b_2, b_3) for a given b_1. Therefore, $K = \Omega(n^2 K')$. This argument holds for all red/blue triangles of $\pi(\hat{f}_D^\nu)$. □

8.2.2 Combinations of faces and concave edges

Let t be the number of concave edges of a rectilinear polyhedron P. When we use concave edges, fewer fingers can induce seven face normals. In this section, we report all combinations of faces and concave edges, such that the fingers on each of these combinations allow form-closure grasps with at most six frictionless point fingers, and that each combination involves seven face normals. In particular, we look at the following three cases: (i) when C is one concave edge and five faces, (ii) when C is two concave edges and three faces, and (iii) when C is three concave edges and one face.

[11] Red triangles of $\pi(\hat{f}_D^\nu)$ or $\pi(\hat{f}_U^\nu)$ look similar to those in the region of $\eta_z > 0$ and $\eta_z < 0$, respectively.

	EN	WN	ES	WS	EU	ED	WU	WD	NU	ND	SU	SD
Rot_1	ND	NU	SD	SU	ED	WD	EU	WU	EN	WN	ES	WS
Rot_2	ED	WD	EU	WU	EN	ES	WN	WS	ND	SD	NU	SU

Table 8.1: The renamed edge families after rotation Rot_1 and after rotation Rot_2.

One concave edge and five faces

We wish to report all sets of a concave edge and five faces[12] that yield form-closure grasps with six frictionless point fingers; C is a concave edge and five faces. Remember that each set of $\mathcal{A}$ reported by the algorithms presented in this section yields a form-closure grasp with six fingers, and two fingers induce two face normals of a family of U or D. To identify all sets of a concave edge and five faces that allow a form-closure grasp $\mathcal{A}_{total}$, we apply the algorithms to P and rotated P with Rot_1 and Rot_2. The following lemma shows that the algorithms on P and rotated P with Rot_1 and Rot_2 can indeed report $\mathcal{A}_{total}$.

Lemma 8.6 *When we rotate a set of one concave edge and five faces that yields a form-closure grasp with Rot_1 or Rot_2, the rotated set belongs to one of the following two cases:*

(i) one vertical edge, two vertical faces[13] and three horizontal faces, and

(ii) one horizontal edge,[14] three vertical faces and two horizontal faces.

Proof: When a form-closure grasp on one concave edge and five faces induces two face normals of E (W), we rotate the set with Rot_1. When the grasp induces two face normals of N (S), we rotate the set with Rot_2. Then the two face normals will become those of D (U). Since a set of one concave edge and five faces induces seven face normals, each of the other families induces one face normal.

Note that an edge is either vertical or horizontal. When we rotate a vertical edge with Rot_1 or Rot_2, the vertical edge becomes horizontal—see Table 8.1. When we rotate a horizontal edge with Rot_1 or Rot_2, the horizontal edge will either become vertical or remain horizontal—see Table 8.1. When the rotated edge is vertical, three of the five faces will be horizontal, which is case (i). When the rotated edge is horizontal, the edge is incident to a horizontal face, thus two of the five faces will be horizontal, which is case (ii). □

First, we wish to report all sets of one vertical edge and five faces that yields a form-closure grasp. One vertical edge and two faces are necessary and sufficient to have a set of points that satisfy the first condition of Lemma 8.2; C' is one vertical edge and two faces. Note that $\hat{e}^h$ of a vertical edge e corresponds to a vertex wrench set of a polygon, and $\hat{f}^h$ of a face f to an edge wrench set of a polygon. See Figure 8.12 (a). Polyhedron P has $O(n)$ faces and t edges. Therefore, the algorithm proposed in Section 3.3.2 can report $\mathcal{A}'$ in $O(n \log n + K')$ time.

We pick a triple of a vertical concave edge and two faces from $\mathcal{A}'$. Without loss of generality, assume that the triple is (e_{EN}, f_W, f_S); e_{EN} is from EN, f_S is from S and f_W is from W. The triple induces three wrench sets $\hat{f}_W$, $\hat{f}_S$ and $\kappa_1\omega_E + \kappa_2\omega_N$, where $\omega_E \in \hat{e}_{EN,E}$, $\omega_N \in \hat{e}_{EN,N}$, $0 \leq \kappa_1, \kappa_2 \leq 1$ and $\kappa_1 + \kappa_2 = 1$. Let $h_E \in \hat{e}^h_{EN,E}$, $h_N \in \hat{e}^h_{EN,N}$, $h_W \in \hat{f}^h_W$ and $h_S \in \hat{f}^h_S$. Since $(e_{EN}, f_W, f_S) \in \mathcal{A}'$, there exist α_1, α_2, α_3, α_4, in the intervals of 0 and 1, such that $\alpha_1 h_E +$

[12] In fact, this set can have one concave edge and four faces, because two fingers are allowed to be on one face.
[13] Remember that vertical edges and faces are parallel to the τ_z axis.
[14] Remember that horizontal edges and faces are parallel to the $\tau_x\tau_y$ plane.

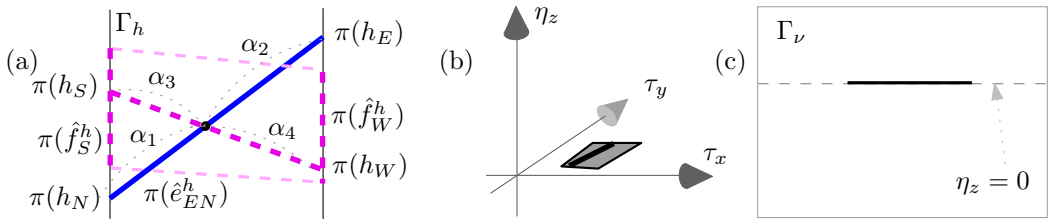


Figure 8.12: (a) The shape induced by $\hat{e}^h_{EN}$, $\hat{f}^h_W$ and $\hat{f}^h_S$ on Γ_h. (b) Δ in $\mathcal{V}$ space. (c) $\pi(\Delta)$ on Γ_ν.

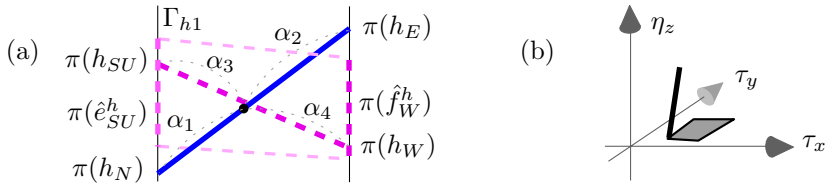


Figure 8.13: (a) The h components induced by one horizontal edge e_{SU} and three faces f_E, f_N and f_W on Γ_{h1}. (b) δ and Δ in $\mathcal{V}$ space.

$\alpha_2 h_N + \alpha_3 h_W + \alpha_4 h_S = \vec{0}$, where $\alpha_1 + \alpha_2 = 1$ and $\alpha_3 + \alpha_4 = 1$. In other words, the intersection point of two line segments $\pi(\hat{e}^h_{EN})$ and $\pi(h_W h_S)$ on Γ_h is the projection of $\alpha_1 h_E + \alpha_2 h_N$ and $-(\alpha_3 h_W + \alpha_4 h_S)$. See Figure 8.12 (a). We define Δ to be the set of points in $\mathcal{V}$ space induced by four fingers along the interior of e_{EN}, f_W and f_S. Then Δ has the following form:

$$\Delta := \{\alpha_1 \nu_E + \alpha_2 \nu_N + \alpha_3 \nu_W + \alpha_4 \nu_S \mid \nu_E \in \hat{e}^\nu_{EN,E}, \nu_N \in \hat{e}^\nu_{EN,N}, \nu_W \in \hat{f}^\nu_W, \nu_S \in \hat{f}^\nu_S\}$$

Replacing with the vectors given in Section 8.1.1 shows that Δ consists of vectors $(0, -\alpha_2 p_{zn} + \alpha_4 p_{zs}, \alpha_1 p_{ze} - \alpha_3 p_{zw})^T = (0, -\alpha_2 p_{zn} + \alpha_4 p_{zs}, \alpha_1 p_{zn} - \alpha_3 p_{zw})^T$, where p_{zn}, p_{zw} and p_{zs} are in some ranges. The equality holds because the finger on e_{EN} touches two incident faces from E and N.

When the fingers are at fixed positions, α_1, α_2, α_3 and α_4 are determined. Thus the three fixed fingers induce a (closed) line segment δ on the plane $\eta_z = 0$ in $\mathcal{V}$ space. When the three fingers move in the interior of e_{EN}, f_S and f_W independently, the segment δ also moves in a certain region on the plane $\eta_z = 0$. This region is Δ, and it is a polygon of a constant complexity on the plane $\eta_z = 0$. The projection $\pi(\Delta)$ is a line segment on the line $\eta_z = 0$ on Γ_ν. When the edge e is from other families such as ES, WN or WS, the shapes of Δ and $\pi(\Delta)$ are similar.

In phase II, we wish to report all triangle triples from $\pi(\hat{f}^\nu_U)$ and $\pi(\hat{f}^\nu_D)$ with a set of points satisfying the second condition of Lemma 8.2 with each set of $\mathcal{A}'$. The phase II of the algorithm in Section 8.2.1 can report all such triangle triples, thus $\mathcal{A}$. There are $O(n)$ red and blue triangles of $\pi(\hat{f}^\nu_U)$ and $\pi(\hat{f}^\nu_D)$, thus the total time complexity to report all K sets of one vertical edge and five faces that allow form-closure grasps is $O(n^2 K' \log^4 n + K)$.

Now we look at the case when C is a horizontal concave edge and five faces. One horizontal concave edge and three faces are necessary and sufficient to have a set of points that satisfy the first condition of Lemma 8.2; C' is a horizontal concave edge and three faces. Note that $\hat{e}^h$ of a horizontal edge e and $\hat{f}^h$ of a face f correspond to edge wrench sets of a polygon. See Figure 8.13 (a). There are $O(n)$ faces and t edges. Therefore, the algorithm proposed in Section 3.3.1 can report $\mathcal{A}'$ in $O(n \log n + K')$ time.

We pick a quadruple of a horizontal concave edge and three faces from $\mathcal{A}'$. Without loss of generality, assume that the quadruple is (e_{SU}, f_E, f_N, f_W); e_{SU} is from SU, f_E is from E, f_N is

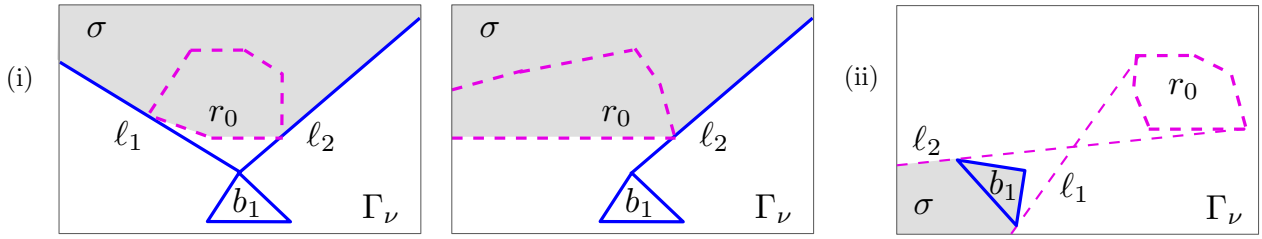


Figure 8.14: When r_0 is induced by a horizontal edge and three faces, the figures show the regions σ to report (i) blue triangle pairs, and (ii) one red triangle and one blue triangle that make a red and blue intersection with r_0.

from N and f_W is from W. The quadruple induces four wrench sets $\hat{f}_E$, $\hat{f}_N$, $\hat{f}_W$ and $\kappa_1\omega_S + \kappa_2\omega_U$, where $\omega_S \in \hat{e}_{SU,S}$, $\omega_U \in \hat{e}_{SU,U}$, $0 \leq \kappa_1, \kappa_2 \leq 1$ and $\kappa_1 + \kappa_2 = 1$. Let $h_E \in \hat{f}^h_E$, $h_N \in \hat{f}^h_N$, $h_W \in \hat{f}^h_W$, $h_{SU} \in \hat{e}^h_{SU}$, $h_S \in \hat{e}^h_{SU,S}$ and $h_U \in \hat{e}^h_{SU,U}$. Since $(e_{SU}, f_E, f_N, f_W) \in \mathcal{A}'$, there exist α_1, α_2, α_3, α_4, in the intervals of 0 and 1, such that $\alpha_1 h_E + \alpha_2 h_N + \alpha_3 h_W + \alpha_4(\kappa_1 h_S + \kappa_2 h_U) = \vec{0}$,[15] where $0 < \kappa_1 \leq 1$, $0 \leq \kappa_2 < 1$,[16] $\kappa_1 + \kappa_2 = 1$, $\alpha_1 + \alpha_2 = 1$ and $\alpha_3 + \alpha_4 = 1$. In other words, the intersection point of two line segments $\pi(h_E h_N)$ and $\pi(h_W h_{SU})$ on Γ_h is the projection of $\alpha_1 h_E + \alpha_2 h_N$ and $-(\alpha_3 h_W + \alpha_4(\kappa_1 h_S + \kappa_2 h_U))$. We define Δ to be the set of points in $\mathcal{V}$ space induced by four fingers along the interior of e_{SU}, f_E, f_N and f_W. The set Δ has the following form:

$$\begin{aligned}\Delta := \quad & \{\alpha_1\nu_E + \alpha_2\nu_N + \alpha_3\nu_W + \alpha_4(\kappa_1\nu_S + \kappa_2\nu_U) \mid \\ & \nu_E \in \hat{f}^\nu_E, \nu_N \in \hat{f}^\nu_N, \nu_W \in \hat{f}^\nu_W, \nu_S \in \hat{e}^\nu_{SU,S}, \nu_U \in \hat{e}^\nu_{SU,U}, \\ & 0 < \kappa_1 \leq 1, 0 \leq \kappa_2 < 1, \kappa_1 + \kappa_2 = 1\}\end{aligned}$$

The set Δ consists of vectors of the form $(\alpha_4\kappa_2, -\alpha_2 p_{zn} + \alpha_4\kappa_1 p_{zs} + \alpha_4\kappa_2 p_{yu}, \alpha_1 p_{ze} - \alpha_3 p_{zw} - \alpha_4\kappa_2 p_{xu})^T = (\alpha_4\kappa_2, -\alpha_2 p_{zn} + \alpha_4\kappa_1 p_{zs} + \alpha_4\kappa_2 p_{ys}, \alpha_1 p_{ze} - \alpha_3 p_{zw} - \alpha_4\kappa_2 p_{xs})^T$. Note that p_{ys}, p_{zs} are fixed numbers, and that p_{ze}, p_{zn}, p_{zw} and p_{xs} are in some ranges.

When the four fingers are at fixed positions, α_1, α_2, α_3 and α_4 are all determined. Thus the four fixed fingers induce a (closed) line segment δ, one endpoint of which lies on the plane $\eta_z = 0$ in $\mathcal{V}$ space. When the four fingers move in the interior of e_{SU}, f_N, f_E and f_W independently, the segment δ also moves in a certain region. This region is Δ, and it is a polyhedron of a constant complexity with a face on the plane $\eta_z = 0$. The projection $\pi(\Delta)$ is a polygon, a side of which lies on the line $\eta_z = 0$. The shapes of Δ and $\pi(\Delta)$ remain the same when the horizontal concave edge is from other families.

In phase II, we wish to report all triangle pairs from $\hat{f}^\nu_U$ and/or $\hat{f}^\nu_D$ with a set of points satisfying the second condition of Lemma 8.2 with each set of $\mathcal{A}'$. Without loss of generality, we assume that $\pi(\Delta)$ has a red part r_0. For r_0, all such face pairs belong to one of the following two cases: (i) the convex hull of two blue triangles b_1 and b_2 of $\pi(\hat{f}^\nu_U)$ or $\pi(\hat{f}^\nu_D)$ intersects r_0 in the interior, and (ii) the convex hull of r_0 and a red triangle r_1 (of red $\pi(\hat{f}^\nu_U)$ or $\pi(\hat{f}^\nu_D)$) intersects a blue triangle b_1 (of blue $\pi(\hat{f}^\nu_U)$ or $\pi(\hat{f}^\nu_D)$) in the interior.

The blue triangle pairs of case (i) can be identified as follows. We pick a blue triangle b_1 from $\pi(\hat{f}^\nu_U)$ or in $\pi(\hat{f}^\nu_D)$, and compute the tangent lines ℓ_1 and ℓ_2 of b_1 and r_0, such that they separates b_1 and r_0. Let σ denote the region bounded by ℓ_1, ℓ_2 and some edges of r_0, such that σ contains the interior of r_0.[17] We report all blue triangles intersecting σ in the interior. See Figure 8.14 (i). We build an order 2 tree described in Section 8.1.3 on the vertices of the $O(n)$ blue triangles of $\pi(\hat{f}^\nu_U)$

[15] Note that $h_U = \vec{0}$.
[16] Since $\kappa_1 h_S$ must be a non-zero vector, κ_1 must be non-zero, thus κ_2 must not be 1.
[17] This is very similar to the case (i) in the previous case: a vertical edge and two faces.

and $\pi(\hat{f}_D^\nu)$ in $O(n^2)$ time, and report all k blue vertices in σ in $O(\log^2 n + k)$ time. We also build a segment intersection search structure on the sides of the $O(n)$ blue triangles of $\pi(\hat{f}_U^\nu)$ and $\pi(\hat{f}_D^\nu)$ in $O(n^2 \log^2 n)$ time, and report all k sides intersecting the boundary of σ in $O(\log^4 n + k)$ time. The time complexity of case (i) is thus $O(n^2 \log^2 n + nK' \log^4 n + K)$.

The face pairs of case (ii) can be identified similarly. We pick a blue triangle b_1 from $\pi(\hat{f}_U^\nu)$ or $\pi(\hat{f}_D^\nu)$, and compute the tangent lines ℓ_1 and ℓ_2 of r_0 and b_1, such that they separate b_1 and r_0. Then we compute the region σ as described in case (iii) in Section 8.2.1; σ is the region bounded by ℓ_1, ℓ_2, and some edges of b_1[18], such that σ contains the interior of b_1. See Figure 8.14 (ii). A triangle intersects σ, if and only if it has at least one vertex in σ, or its sides intersect the boundary of σ.

We use the triangle search structure to identify all the vertices of red triangles in σ. When σ is bounded by four lines, we divide σ into two regions, each of which is bounded by at most three lines. For example in Figure 8.14 (ii), σ can be divided into b_1 and $\sigma - b_1$. We store the vertices of $O(n)$ red triangles in a triangle search structure in $O(n^2 \log n)$ time, and report all k red points in σ in $O(\log^3 n + k)$ time. We also build a segment intersection search structure on the sides of $O(n)$ red triangles of $\pi(\hat{f}_U^\nu)$ and $\pi(\hat{f}_D^\nu)$ in $O(n^2 \log^2 n)$ time. We report all k sides intersecting the boundary of σ in $O(\log^4 n + k)$ time. The total time complexity of case (ii) is thus $O(n^2 \log^2 n + nK' \log^4 n + K)$.

We repeat these processes for the blue part of $\pi(\Delta)$. Note that this will report all pairs of faces from one family of U or D, or from two distinct families (one from U and the other from D). There are $O(n)$ red and blue triangles of $\pi(\hat{f}_U^\nu)$ and $\pi(\hat{f}_D^\nu)$, so the total preprocessing time for a horizontal edge and five faces is $O(n^2 \log^2 n)$. Hence the time complexity of reporting all sets of an edge and five faces is $O(n^2 \log^2 n + nK' \log^4 n + K)$.

Theorem 8.7 *All K sets of one concave edge and five faces of P that allow form-closure grasps with six frictionless point fingers can be enumerated in $O(n^2 K' \log^4 n + K)$ time, where $K' = |\mathcal{A}|$.*

Two concave edges and three faces

We wish to report all sets of two concave edges and three faces[19] that yield form-closure grasps with five frictionless point fingers; C is two concave edges and three faces. Remember that each set of $\mathcal{A}$ reported by the algorithms presented in this section yields a form-closure grasp with five fingers, and two of the five fingers induce two face normals of one family of U or D. To identify all sets of two concave edges and three faces that allow a form-closure grasp, we apply the algorithms to P and rotated P with Rot_1 and Rot_2. The following lemma shows that the algorithms on P and rotated P with Rot_1 and Rot_2 can report all sets of two concave edges and three faces that yield form-closure grasps.

Lemma 8.8 *When we rotate a set of two concave edges and three faces that yields a form-closure grasp with Rot_1 or Rot_2, the rotated set belongs to one of the following four cases:*

(i) two vertically parallel edges and three horizontal faces,

(ii) two horizontally parallel edges, two vertical faces and one horizontal face,

(iii) two horizontally skewed edges, two vertical faces and one horizontal face,

(iv) one vertical edge, one horizontal edge, one vertical face and two horizontal faces.

[18] At most two edges of b_1 bound σ.

[19] In fact, this set can have two concave edges and two faces, because two fingers are allowed to be on one face.

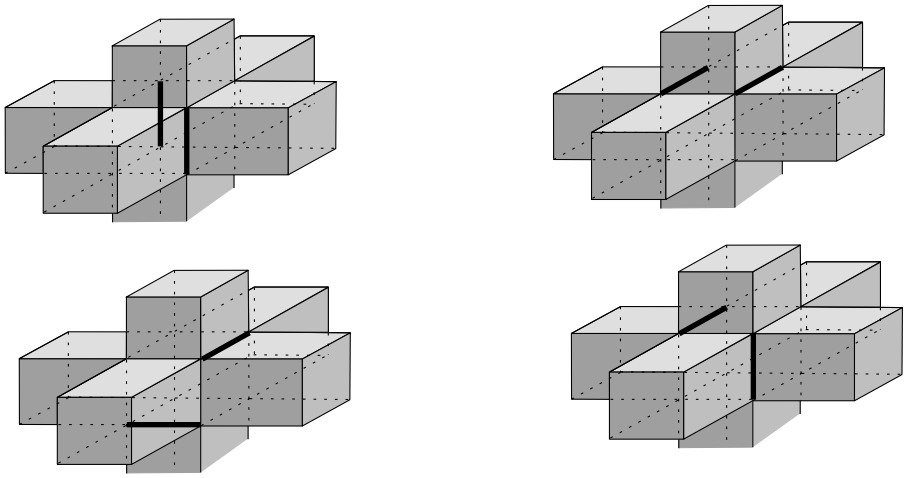


Figure 8.15: The four cases of the relative positions of two edges of P.

Proof: Note that an edge pair is either parallel or skewed. More precisely, when two edges are parallel, they can be both vertical or both horizontal. When two edges are skewed, they can be both horizontal edges, or one vertical edge and one horizontal edge. See Figure 8.15. All these four cases of an edge pair match the four cases of an edge pair stated in the lemma.

We take any set of two concave edges and three faces of P that yields a form-closure grasp with five fingers. Since two edges and three faces involve seven face normals, five face normals are for five families (one for each family), and two face normals are for one family. If two of the five fingers induce two face normals of a family U or D, then we are done. Hence we will look at the cases when two fingers induce two face normals of a family of E, W, N or S. If there are two face normals of E or W, we apply Rot_1; the three face normals of E and W rotated with Rot_1 will become three face normals of D and U. If there are two face normals of N or S, we apply Rot_2; the three face normals of N and S rotated with Rot_2 will become three face normals of D and U. Note that these three face normals could be induced by the fingers on edges. Next we will look at the four cases of the relative positions of two edges.

First, we look at the case of two vertical edges and three faces that yield a form-closure grasp. When we rotate any vertical edge with Rot_1 or Rot_2, it becomes horizontal—see Table 8.1. Thus when we rotate a set of two vertical edges and three faces with a form-closure grasp, this set becomes a set of two horizontal edges and three faces. Observe that the set of two horizontal edges and three faces induces three face normals of D and U, two of which are induced by two fingers on two horizontal edges. Hence this set has two horizontally parallel edges, two vertical faces and one horizontal face (case (ii)).

Let us look at the second case of two horizontally parallel edges and three faces that yield a form-closure grasp. Without loss of generality, assume that two fingers induce two face normals of E. When we rotate a horizontally parallel edge pair from EU, ED, WU and/or WD with Rot_1, it remains as a horizontally parallel edge pair—see Table 8.1. Observe that this rotated set induces three face normals of D and U, two of which are induced by two fingers on the two horizontal edges. Thus this set remains as two horizontally parallel edges and three faces of case (ii). When we rotate a horizontally parallel edge pair from NU, ND, SU and SD, it becomes a vertical edge pair—see Table 8.1. The rotated set induces three face normals of D and U. Since both of the two edges are vertical, none of these three face normals is induced by the fingers on the edges. Thus this rotated set is two vertical edges and three horizontal faces of case (i).

Now we look at the case of a horizontally skewed edge pair and three faces that yield a form-closure grasp. Without loss of generality, assume that two fingers induce two face normals of E. Note that two horizontally skewed edges have one edge from EU, ED, WU and WD, and the other edge from NU, ND, SU and SD. When we rotate a horizontally skewed edge pair with Rot_1, this set becomes one vertical edge and one horizontal edge—see Table 8.1. Observe that the set of one vertical edge, one horizontal edge and three faces induces three face normals of D and U, one of which is induced by the finger on the horizontal edge. Thus this rotated set is one vertical edge, one horizontal edge, one vertical face and two horizontal faces, which is case (iv).

Finally, we look at the case of one vertical edge, one horizontal edge and three faces that yield a form-closure grasp. Without loss of generality, assume that two fingers induce two face normals of E. When we rotate a vertical edge with Rot_1, it becomes horizontal. When we rotate a horizontal edge from EU, ED, WU and WD with Rot_1, it remains horizontal—see Table 8.1. Observe that this set of two horizontally skewed edges and three faces induces three face normals of D and U, two of which are induced by two fingers on two horizontal edges. Thus the rotated set will be a horizontally skewed edge pair and three faces, which is case (iii). When we rotate a horizontal edge from NU, ND, SU and SD with Rot_1, it becomes vertical—see Table 8.1. Observe that this rotated set induces three face normals of D and U, one of which is induced by a finger on the horizontal edge. Hence this rotated set will be one vertical edge, one horizontal edge and three faces, which is case (iv). □

We first look at the case of two vertical edges and three faces of case (i) that yield a form-closure grasp. Two vertical edges are necessary and sufficient to have a set of points that satisfy the first condition of Lemma 8.2; C' is two vertical edges. Note that $\hat{e}^h$ of a vertical concave edge e corresponds to a vertex wrench set of a polygon. See Figure 3.17 (c). There are t vertical concave edges, so the algorithm proposed in Section 3.3.3 can report $\mathcal{A}'$ in $O(t \log^2 t + K')$ time.

We pick an edge pair from $\mathcal{A}'$. Without loss of generality, assume that the pair is (e_{EN}, e_{WS}). The edge pair induces two wrench sets $\kappa_1\omega_E + \kappa_2\omega_N$ and $\gamma_1\omega_W + \gamma_2\omega_S$, where $\omega_E \in \hat{e}_{EN,E}$, $\omega_N \in \hat{e}_{EN,N}$, $\omega_W \in \hat{e}_{WS,W}$, $\omega_S \in \hat{e}_{WS,S}$, $0 \leq \kappa_1, \kappa_2, \gamma_1, \gamma_2 \leq 1$, $\kappa_1 + \kappa_2 = 1$ and $\gamma_1 + \gamma_2 = 1$. Let $h_E \in \hat{e}^h_{EN,E}$, $h_N \in \hat{e}^h_{EN,N}$, $h_W \in \hat{e}^h_{WS,W}$ and $h_S \in \hat{e}^h_{WS,S}$. Since $(e_{EN}, e_{WS}) \in \mathcal{A}'$, there exist α_1, α_2, α_3, α_4 among κ_1, κ_2, γ_1 and γ_2 respectively, such that $\alpha_1 h_E + \alpha_2 h_N + \alpha_3 h_W + \alpha_4 h_S = \vec{0}$, where $\alpha_1 + \alpha_2 = 1$ and $\alpha_3 + \alpha_4 = 1$. Note that α_1, α_2, α_3, α_4 are determined because $\pi(h_E h_N)$ and $\pi(h_W h_S)$ are two intersecting line segments on Γ_h. We define Δ to be the set of points in $\mathcal{V}$ space induced by the two fingers sliding on e_{EN} and e_{WS}. The set Δ has the following form:

$$\Delta := \{\alpha_1\nu_E + \alpha_2\nu_N + \alpha_3\nu_W + \alpha_4\nu_S \mid \nu_E \in \hat{e}^\nu_{EN,E}, \nu_N \in \hat{e}^\nu_{EN,N}, \nu_W \in \hat{e}^\nu_{WS,W}, \nu_S \in \hat{e}^\nu_{WS,S}\}.$$

The set Δ consists of vectors $(0, -\alpha_2 p_{zn} + \alpha_4 p_{zs}, \alpha_1 p_{ze} - \alpha_3 p_{zw})^T = (0, -\alpha_2 p_{zn} + \alpha_4 p_{zs}, \alpha_1 p_{zn} - \alpha_3 p_{zs})^T$, where the p_{zn} and p_{zs} are in some open ranges. The equality holds because the finger at e_{EN} touches two incident faces from E and N, and the finger at e_{WS} touches two incident faces from W and S. The set Δ is a polygon of a constant complexity on the plane $\eta_z = 0$ in $\mathcal{V}$ space, and the projection $\pi(\Delta)$ is a line segment on the line $\eta_z = 0$ on Γ_ν.

In phase II, we wish to report all triangle triples from $\pi(\hat{f}^\nu_U)$ and $\pi(\hat{f}^\nu_D)$ with a set of points satisfying the second condition of Lemma 8.2 with each set of $\mathcal{A}'$. The phase II of the algorithm in Section 8.2.1 can report all such triangle triples, thus $\mathcal{A}$. There are $O(n)$ faces from U and D, therefore, the total time complexity of this case is $O(n^2 K' \log^4 n + K)$.

We now look at the case of two horizontal edges and three faces that yield a form-closure grasp. These horizontal edges can be parallel or skewed. In any case, two horizontal edges and two faces are necessary and sufficient to have a set of points that satisfy the first condition of Lemma 8.2; C' is two horizontal edges and two faces. The algorithm in Section 3.3.1 can report

$\mathcal{A}'$ in $O(n \log n + K')$ time; there are $O(n)$ faces and t edges.

We pick a quadruple from $\mathcal{A}'$. We first look at the case when the two horizontal edges are parallel. Without loss of generality, assume that the quadruple is $(e_{EU}, e_{WD}, f_N, f_S)$; e_{EU} is from EU, e_{WD} is from WD, f_N is from N and f_S is from S. The quadruple induces four wrench sets $\hat{f}_N$, $\hat{f}_S$, $\kappa_1\omega_E + \kappa_2\omega_U$ and $\gamma_1\omega_W + \gamma_2\omega_D$, where $\omega_E \in \hat{e}_{EU,E}$, $\omega_U \in \hat{e}_{EU,U}$, $\omega_W \in \hat{e}_{WD,W}$, $\omega_D \in \hat{e}_{WD,D}$, $0 \leq \kappa_1, \kappa_2, \gamma_1, \gamma_2 \leq 1$, $\kappa_1 + \kappa_2 = 1$ and $\gamma_1 + \gamma_2 = 1$. Let $h_E \in \hat{e}^h_{EU,E}$, $h_U \in \hat{e}^h_{EU,U}$, $h_W \in \hat{e}^h_{WD,W}$, $h_D \in \hat{e}^h_{WD,D}$, $h_N \in \hat{f}^h_N$ and $h_S \in \hat{f}^h_S$. Since $(e_{EU}, e_{WD}, f_N, f_S) \in \mathcal{A}'$, there exist α_1, α_2, α_3, α_4 in the intervals of 0 and 1, such that $\alpha_1(\kappa_1 h_E + \kappa_2 h_U) + \alpha_2 h_N + \alpha_3(\gamma_1 h_W + \gamma_2 h_D) + \alpha_4 h_S = \vec{0}$ for $0 < \kappa_1, \gamma_1 \leq 1$, $0 \leq \kappa_2, \gamma_2 < 1$[20] ($\kappa_1 + \kappa_2 = 1$ and $\gamma_1 + \gamma_2 = 1$), where $\alpha_1 + \alpha_2 = 1$ and $\alpha_3 + \alpha_4 = 1$. We define Δ to be the set of points in $\mathcal{V}$ space induced by four fingers along the interior of e_{EU}, e_{WD}, f_N and f_S. The set Δ has the following form:

$$\begin{aligned}\Delta \;:=\; \{&\alpha_1(\kappa_1\nu_E + \kappa_2\nu_U) + \alpha_2\nu_N + \alpha_3(\gamma_1\nu_W + \gamma_2\nu_D) + \alpha_4\nu_S \mid \\ &\nu_E \in \hat{e}^\nu_{EU,E}, \nu_U \in \hat{e}^\nu_{EU,U}, \nu_N \in \hat{f}^\nu_N, \nu_W \in \hat{e}^\nu_{WD,W}, \nu_D \in \hat{e}^\nu_{WD,D}, \nu_S \in \hat{f}^\nu_S \\ &0 < \kappa_1, \gamma_1 \leq 1, 0 \leq \kappa_2, \gamma_2 < 1, \kappa_1 + \kappa_2 = 1, \gamma_1 + \gamma_2 = 1\}.\end{aligned}$$

The set Δ consists of vectors $(\alpha_1\kappa_2 - \alpha_3\gamma_2, -\alpha_2 p_{zn} + \alpha_1\kappa_2 p_{yu} - \alpha_3\gamma_2 p_{yd} + \alpha_4 p_{zs}, \alpha_1\kappa_1 p_{ze} - \alpha_1\kappa_2 p_{xu} - \alpha_3\gamma_1 p_{zw} + \alpha_3\gamma_2 p_{xd})^T = (\alpha_1\kappa_2 - \alpha_3\gamma_2, -\alpha_2 p_{zn} + \alpha_1\kappa_2 p_{ye} - \alpha_3\gamma_2 p_{yw} + \alpha_4 p_{zs}, \alpha_1\kappa_1 p_{ze} - \alpha_1\kappa_2 p_{xe} - \alpha_3\gamma_1 p_{zw} + \alpha_3\gamma_2 p_{xw})^T$. Note that p_{xe}, p_{ze}, p_{xw}, p_{zw} are fixed numbers, and p_{zn}, p_{zs}, p_{ye}, p_{yw} are in some ranges. Also note that α_1, α_2, α_3, α_4 are in the ranges dictated by the equation.

A fixed position vector on e_{EU}, e_{WU}, f_N and f_S determines α_1, α_2, α_3 and α_4; only κ_1, κ_2 γ_1 and γ_2 are variable. This implies that the fingers at fixed positions induce a tetrahedron δ in $\mathcal{V}$ space, whose vertices are determined by the combinations of the extreme values of κ_1, κ_2 γ_1 and γ_2, such as $\kappa_1 = 1$, $\kappa_2 = 0$, and $\gamma_1 = 0$, $\gamma_2 = 1$. The vertices of δ move continuously in certain regions, which are polytopes of constant complexities. Note that one vertex of any δ lies on the plane $\eta_z = 0$ in $\mathcal{V}$ space. The set Δ is a convex polytope in $\mathcal{V}$ space, thus $\pi(\Delta)$ is a convex polygon on Γ_ν. Δ has a face on the plane $\eta_z = 0$, thus $\pi(\Delta)$ has a side on the line $\eta_z = 0$. The shapes of Δ and $\pi(\Delta)$ for two horizontally parallel or skewed edges and two faces of other cases are similar to these described above.

In phase II, we wish to find all triangles from $\pi(\hat{f}^\nu_U)$ and $\pi(\hat{f}^\nu_D)$, such that each triangle has a set of points satisfying the second condition of Lemma 8.2 with each set of $\mathcal{A}'$. Assume that $\pi(\Delta)$ has a red part, and we call it r_0. The case of a blue part of $\pi(\Delta)$ is similar. We report all blue triangles of $\pi(\hat{f}^\nu_U)$ and $\pi(\hat{f}^\nu_D)$ on Γ_ν, which intersect r_0. A blue triangle intersects r_0 in the interior, if and only if one of the following holds: (a) the red and blue boundary segments intersect each other in the interior, or (b) a blue vertex is contained in the red query triangle or vice versa. To identify these intersections, we use the segment intersection structure and the triangle search structure. We store the sides of $O(n)$ blue triangles in a segment intersection structure in $O(n^2 \log^2 n)$ time, and report all k blue sides intersecting a red line segment in $O(\log^4 n + k)$ time. We also store the vertices of $O(n)$ blue triangles in a triangle search structure in $O(n^2 \log n)$ time, and report all k blue vertices in a red triangle in $O(\log^3 n + k)$ time. The total time complexity is thus $O(n^2 \log^2 n + K' \log^4 n + K)$.

Finally, we look at the case of one vertical edge, one horizontal edge and three faces that yield a form-closure grasp. One vertical edge, one horizontal edge and a face are necessary and sufficient to have a set of points that satisfy the first condition of Lemma 8.2; C' is one vertical edge, one horizontal edge and a face. Note that $\hat{e}^h$ of a horizontal edge e and $\hat{f}^h$ of a face f correspond to edge wrench sets of a polygon. There are $O(n)$ faces and t edges. Therefore, the algorithm

[20] Since $\kappa_1 h_E$ and $\gamma_1 h_W$ must be non-zero vectors, κ_1 and γ_1 must be non-zero, thus κ_2 and γ_2 must not be 1.

proposed in Section 3.3.2 can report $\mathcal{A}'$ in $O(n \log n + K')$ time.

We pick a triple of one vertical edge, one horizontal edge and a face from $\mathcal{A}'$. Without loss of generality, assume that it is (e_{EN}, e_{SU}, f_W); e_{EN} is from EN, e_{SU} is from SU, and f_W is from W. Let $h_E \in \hat{e}^h_{EN,E}$, $h_N \in \hat{e}^h_{EN,N}$, $h_W \in \hat{f}^h_W$, $h_S \in \hat{e}^h_{SU,S}$ and $h_U \in \hat{e}^h_{SU,U}$. With the same argument in the case of a concave edge and five faces in Section 8.2.2, there exist α_1, α_2, α_3, α_4 in the intervals of 0 and 1, such that $\alpha_1 h_E + \alpha_2 h_N + \alpha_3 h_W + \alpha_4(\kappa_1 h_S + \kappa_2 h_U) = \vec{0}$ for $0 < \kappa_1 \leq 1$ and $0 \leq \kappa_2 < 1$, where $\kappa_1 + \kappa_2 = 1$. We define Δ to be the set of points in $\mathcal{V}$ space induced by three fingers along the interior of e_{EN}, e_{SU} and f_W. The set Δ has the following form:

$$\begin{aligned}\Delta := \; & \{\alpha_1\nu_E + \alpha_2\nu_N + \alpha_3\nu_W + \alpha_4(\kappa_1\nu_S + \kappa_2\nu_U) \mid \\ & \nu_E \in \hat{e}^\nu_{EN,E}, \nu_N \in \hat{e}^\nu_{EN,N}, \nu_W \in \hat{f}^\nu_W, \nu_S \in \hat{e}^\nu_{SU,S}, \nu_U \in \hat{e}^\nu_{SU,U}, \\ & 0 < \kappa_1 \leq 1, 0 \leq \kappa_2 < 1, \kappa_1 + \kappa_2 = 1\}.\end{aligned}$$

The set Δ consists of vectors $(\alpha_4\kappa_2, -\alpha_2 p_{zn} + \alpha_4\kappa_1 p_{zs} + \alpha_4\kappa_2 p_{yu}, \alpha_1 p_{ze} - \alpha_3 p_{zw} - \alpha_4\kappa_2 p_{xu})^T = (\alpha_4\kappa_2, -\alpha_2 p_{zn} + \alpha_4\kappa_1 p_{zs} + \alpha_4\kappa_2 p_{ys}, \alpha_1 p_{zn} - \alpha_3 p_{zw} - \alpha_4\kappa_2 p_{xs})^T$, where p_{yn}, p_{ys} and p_{zs} are fixed numbers, and p_{zn}, p_{zw} and p_{xs} are in some intervals.

When the finger positions are fixed, only κ_1 and κ_2 are variable, thus they induce a line segment δ in $\mathcal{V}$ space; the extreme values of κ_1 and κ_2 determines the endpoints of δ: when $\kappa_1 = 0$, $\kappa_2 = 1$, and when $\kappa_1 = 1$, $\kappa_2 = 0$. Note that an endpoint of δ is on the plane $\eta_z = 0$. When the three fingers move in the interior of e_{EN}, e_{SU} and f_W independently, the corresponding segment δ also moves in a certain region. This region is Δ, and it is a polyhedron of a constant complexity, a face of which lies on the plane $\eta_z = 0$. Thus the projection $\pi(\Delta)$ is a polygon, a side of which lies on the line $\eta_z = 0$ on Γ_ν. See Figure 8.14. The shapes of Δ and $\pi(\Delta)$ of other cases such as (e_{EN}, e_S, f_{WU}) are similar to these described above.

In phase II, we wish to report all triangle pairs from $\hat{f}^\nu_U$ and/or $\hat{f}^\nu_D$ with a set of points satisfying the second condition of Lemma 8.2 with each set of $\mathcal{A}'$. The phase II of the algorithm for the case of a horizontal edge and five faces in Section 8.2.2 can report all such triangle pairs, thus $\mathcal{A}$ in $O(n^2 \log^2 n + nK' \log^4 n + K)$ time; there are $O(n)$ faces.

Theorem 8.9 *All K sets of two parallel and skewed concave edges and three faces of P that allow form-closure grasps with five frictionless point fingers can be enumerated in $O(n^2 K' \log^4 n + K)$ and $O(n^2 \log^2 n + nK' \log^4 n + K)$ time, respectively, where $K' = |\mathcal{A}|$.*

Three concave edges and one face

We wish to report all sets of three concave edges and a face that yield form-closure grasps with four frictionless point fingers; C is three concave edges and a face. Remember that each set of $\mathcal{A}$ reported by the algorithms presented in this section yields a form-closure grasp with four fingers, and two of the four fingers induce two face normals of a family of U or D. To identify all sets of three concave edges and a face that allow a form-closure grasp, we apply the algorithms to P and rotated P with Rot_1 and Rot_2. The following lemma shows that the algorithms on P and rotated P with Rot_1 and Rot_2 can report all sets of three concave edges and a face that yield form-closure grasps.

Lemma 8.10 *When we rotate a set of three concave edges and one face that yields a form-closure grasp, with Rot_1 or Rot_2, the rotated set belongs to one of the following two cases:*

(i) two horizontal edges, one vertical edge, and a horizontal face,

(ii) three horizontal edges and a vertical face.

Proof: We take any set of three concave edges and a face of P that yields a form-closure grasp with four fingers. Any set of three edges belongs to one of the following four cases: (a) two horizontal edges and one vertical edge, (b) three horizontal edges, (c) two vertical edges and one horizontal edge, and (d) three vertical edges. Fingers on three vertical edges and a face (case (d)) cannot yield a form-closure grasp, because fingers on vertical edges never induce a face normal of U or D, and we have only one face to cover both families of U and D. Without loss of generality, we place a finger on a face from U. Then no finger induces a face normal of D, thus the fingers fail to achieve form-closure. Hence the set of three concave edges and a face of P that yields a form-closure grasp belongs to one of the first three cases (a), (b) and (c).

We first look at the case when fingers on two horizontal edges, a vertical edge and a face yield a form-closure grasp (case (a)). If this set induces two face normals of a family of U or D, then we are done. If this set induces two face normals of a family of E, W, N or S (the face normals could be induced by the fingers on edges), then we apply Rot_1 or Rot_2 accordingly. Without loss of generality, assume that the set induces two face normals of E. Then we rotate the set with Rot_1. The three face normals of E and W rotated with Rot_1 will become three face normals of D and U, which may be induced by the fingers on edges. When we rotate any vertical edge with Rot_1, it becomes horizontal—see Table 8.1. When we rotate two horizontally skewed edges with Rot_1, they become one vertical edge and one horizontal edge—see the proof of Lemma 8.8 and Table 8.1. When we rotate two horizontally parallel edges from EU, ED, WU and WD with Rot_1, they remain as two horizontally parallel edges—see Table 8.1. The set cannot have two horizontally parallel edges from NU, ND, SU and SD. If it does, it must induce two face normals of N or S, which contradicts that it induces two face normals of E.[21] Therefore, the rotated edge triple will be one of the two combinations: (1) two horizontal edges and one vertical edge, (2) three horizontal edges. Since the rotated set involve three face normals of D and U, for the combination of (1), two of the three face normals are induced by two fingers on the two horizontal edges. Thus we need a horizontal face for two horizontal edges and one vertical edge to yield a form-closure grasp, which leads to case (i). Likewise, for the combination of (2), all three face normals of D and U are induced by three fingers on the three horizontal edges. Thus we need a vertical face for three horizontal edges to yield a form-closure grasp, which leads to case (ii).

We now look at the case when fingers on three horizontal edges and a face yield a form-closure grasp (case (b)). Since a horizontal edge is incident to a horizontal face, this set of case (b) already involves three face normals of U and D. The vertical faces that the three horizontal edges are incident to must be from three distinct families. Thus we need a vertical face for three horizontal edges and a face to yield a form-closure grasp, which leads to case (ii).

Finally we look at the case when fingers on two vertical edges, one horizontal edge and a face yield a form-closure grasp (case (c)). Observe that this set involves two horizontal face normals and five vertical face normals. This implies that two fingers induce two face normals of a family E, W, N or S. Without loss of generality, assume that the set induces two face normals of E. Then we rotate the set with Rot_1. When we rotate any vertical edge with Rot_1, it becomes horizontal. When we rotate any horizontal edge with Rot_1, it may either remain horizontal, or become vertical. See Table 8.1. Therefore, the rotated edge triple will be one of the two combinations: (1) two horizontal edges and one vertical edge, (2) three horizontal edges. As shown in handling case (a), these two combinations lead to case (i) and case (ii) respectively. □

We wish to report all sets of three horizontal edges and a face that yield form-closure grasps with four frictionless point fingers; C is three horizontal edges and a face. We need all of three horizontal edges and a face to have a set of points that satisfy the first condition of Lemma 8.2; C'

[21] If the set induces two face normals of a family N or S, we should rotate P with Rot_2, not with Rot_1.

is three horizontal edges and a face. In phase I, the algorithm proposed in Section 3.3.1 can report $\mathcal{A}'$ in $O(n \log n + K')$ time. In phase II, we check whether each set of $\mathcal{A}'$ satisfies the second condition of Lemma 8.2; we need the whole set of three horizontal edges and a vertical face to compute the coefficients α_1, α_2, α_3 and α_4, which also determines the coefficients β_5, β_6 and β_7 in Lemma 8.2.

Now we provide an efficient algorithm to report all sets of two horizontal edges, one vertical edge, and one horizontal face. Two horizontal edges and one vertical edge are necessary and sufficient to have a set of points that satisfy the first condition of Lemma 8.2; C' is three edges. Note that $\hat{e}^h$ of a horizontal edge e corresponds to an edge wrench set, and $\hat{e}^h$ of a vertical edge e to a vertex wrench set of a polygon. There are t edges. Therefore, the algorithm proposed in Section 3.3.2 can report $\mathcal{A}'$ in $O(t \log t + K')$ time.

We pick a triple from $\mathcal{A}'$. Without loss of generality, assume that the triple is (e_{EN}, e_{WD}, e_{SU}); e_{EN} is from EN, e_{SU} is from SU and e_{WD} is from WD. Let $h_E \in \hat{e}^h_{EN,E}$, $h_N \in \hat{e}^h_{EN,N}$, $h_W \in \hat{e}^h_{WD,W}$, $h_D \in \hat{e}^h_{WD,D}$, $h_S \in \hat{e}^h_{SU,S}$ and $h_U \in \hat{e}^h_{SU,U}$. With the same argument in the case of a concave edge and five faces in Section 8.2.2, there exist α_1, α_2, α_3, α_4 in the intervals of 0 and 1, such that $\alpha_1 h_E + \alpha_2 h_N + \alpha_3(\gamma_1 h_W + \gamma_2 h_D) + \alpha_4(\kappa_1\ h_S + \kappa_2 h_U) = \vec{0}$ for $0 < \kappa_1, \gamma_1 \leq 1$, $0 \leq \kappa_2, \gamma_2 < 1$ ($\kappa_1 + \kappa_2 = 1$ and $\gamma_1 + \gamma_2 = 1$), where $\alpha_1 + \alpha_2 = 1$ and $\alpha_3 + \alpha_4 = 1$. We define Δ to be the set of points in $\mathcal{V}$ space induced by three fingers along the interior of e_{EN}, e_{SU} and e_{WD}. The set Δ has the following form:

$$\begin{aligned}\Delta := \ & \{\alpha_1\nu_E + \alpha_2\nu_N + \alpha_3(\gamma_1\nu_W + \gamma_2\nu_D) + \alpha_4(\kappa_1\ \nu_S + \kappa_2\nu_U) \mid \\ & \nu_E \in \hat{e}^\nu_{EN,E}, \nu_N \in \hat{e}^\nu_{EN,N}, \nu_W \in \hat{e}^\nu_{WD,W}, \nu_D \in \hat{e}^\nu_{WD,D}, \nu_S \in \hat{e}^\nu_{SU,S}, \nu_U \in \hat{e}^\nu_{SU,U}, \\ & 0 < \kappa_1, \gamma_1 \leq 1, 0 \leq \kappa_2, \gamma_2 < 1, \kappa_1 + \kappa_2 = 1, \gamma_1 + \gamma_2 = 1\}.\end{aligned}$$

The set Δ consists of vectors $(\alpha_4\kappa_2 - \alpha_3\gamma_2, -\alpha_2 p_{zn} - \alpha_3\gamma_2 p_{yd} + \alpha_4\kappa_1 p_{zs} + \alpha_4\kappa_2 p_{yu}, \alpha_1 p_{ze} - \alpha_3\gamma_1 p_{zw} + \alpha_3\gamma_2 p_{xd} - \alpha_4\kappa_2 p_{xu})^T = (\alpha_4\kappa_2 - \alpha_3\gamma_2, -\alpha_2 p_{zn} - \alpha_3\gamma_2 p_{yw} + \alpha_4\kappa_1 p_{zs} + \alpha_4\kappa_2 p_{ys} -, \alpha_1 p_{zn} - \alpha_3\gamma_1 p_{zw} + \alpha_3\gamma_2 p_{xw} - \alpha_4\kappa_2 p_{xs})^T$, where p_{zs}, p_{ys}, p_{zw}, p_{xw} are fixed numbers, and p_{zn}, p_{yw} and p_{xs} are in some ranges.

When the position vectors are fixed, α_1, α_2, α_3 and α_4 are also determined; only κ_1, κ_2 γ_1 and γ_2 are variable. This implies that the fingers induce a tetrahedron δ in $\mathcal{V}$ space, whose vertices are determined by the combinations of the extreme values of κ_1, κ_2 γ_1 and γ_2. When the three fingers move in the interior of e_{EN}, e_{SU} and e_{WD} independently, the corresponding δ also moves in a certain region. This region is Δ, and it is a convex polytope in $\mathcal{V}$ space; $\pi(\Delta)$ is a convex polygon on Γ_ν. The shapes of Δ and $\pi(\Delta)$ for other sets of an edge triple are similar to these described above.

In phase II, we wish to find all triangles from $\pi(\hat{f}^\nu_U)$ and $\pi(\hat{f}^\nu_D)$ with a set of points satisfying the second condition of Lemma 8.2 with each set of $\mathcal{A}'$. The phase II of the algorithm for the case of two horizontal edges and three faces in Section 8.2.2 can identify all such triangles in $O(n^2 \log^2 n + K' \log^4 n + K)$ time; there are $O(n)$ triangles of $\pi(\hat{f}^\nu_U)$ and $\pi(\hat{f}^\nu_D)$.

When t is sufficiently small, we can find $\mathcal{A}$ in a brute-force manner. When $t < n^{1/3}$, we report $\mathcal{A}$ in $O(t^3 n)$ time.

Theorem 8.11 *All K sets of two horizontal edges, one vertical edge and one face of P that yield form-closure grasps with four frictionless point fingers can be enumerated in:*

1. $O(t^3 n)$ *time, when* $t < n^{1/3}$,
2. $O(n^2 \log^2 n + K' \log^4 n + K)$ *time ($K' = |\mathcal{A}|$), when* $n^{1/3} \leq t \leq n$.

8.2.3 Combinations of faces and concave vertices

Let m be the number of concave vertices of a rectilinear polyhedron P. In this section, we report all combinations of faces and concave vertices that allow form-closure grasps with at most five frictionless point fingers. The combinations that we consider in this section are: (i) one concave vertex and four faces, and (ii) two concave vertices and one face.

One concave vertex and four faces

We wish to report all sets of one concave vertex and four faces[22] that yield form-closure grasps with five frictionless point fingers; C is a concave vertex and four faces. Remember that each set of $\mathcal{A}$ reported by the algorithms presented in this section yields a form-closure grasp with five fingers, and two of the five fingers induce two face normals of a family of U or D. To identify all sets of a concave vertex and four faces that allow a form-closure grasp, we apply the algorithms to P and rotated P with Rot_1 and Rot_2. The following lemma shows that the algorithms on P and rotated P with Rot_1 and Rot_2 can report all sets of a concave vertex and four faces with form-closure grasps.

Lemma 8.12 *When we rotate a set of a concave vertex and four faces that yields a form-closure grasp with five fingers, using Rot_1 or Rot_2, the rotated set becomes a set of a concave vertex and four faces, such that two of the five fingers induce two face normals of a family of U or D.*

Proof: We take any set of a concave vertex and four faces that yields a form-closure grasp with five fingers. If two of the five fingers induce two face normals of a family of U or D, we are done. Hence we look at the case when the two fingers induce two face normals of E, W, N or S.

Without loss of generality, assume that two fingers induce two face normals of E. When we rotate a concave vertex with Rot_1, it is still incident to three faces: one from E or W, one from N or S, and one from U or D. And the two face normals of E will be two face normals of D, with Rot_1. □

One vertex and two faces are necessary and sufficient to have a set of points that satisfy the first condition of Lemma 8.2; C' is a concave vertex and two faces. We first see the shape of $\hat{v}^h$ of a vertex v on Γ_h. Without loss of generality, we take a vertex v_{ENU} from ENU. A finger at v_{ENU} induces a wrench set $\mu_1\hat{v}_{ENU,E} + \mu_2\hat{v}_{ENU,N} + \mu_3\hat{v}_{ENU,U}$[23] for all $0 \leq \mu_1, \mu_2, \mu_3 \leq 1$, where $\mu_1 + \mu_2 + \mu_3 = 1$. Note that $\pi(\mu_1\hat{v}^h_{ENU,E})$ for all μ_1 and $\pi(\mu_2\hat{v}^h_{ENU,N})$ for all μ_2 map to two points on Γ_h, thus $\pi(\mu_1\hat{v}^h_{ENU,E}\mu_2\hat{v}^h_{ENU,N})$ for any μ_1 and μ_2 forms a line segment. Therefore, $\hat{v}^h$ of a vertex v corresponds to a vertex wrench set, and $\hat{f}^h$ of a face f corresponds to an edge wrench set of a polygon. There are $O(n)$ faces and m vertices, therefore, the algorithm proposed in Section 3.3.2 can report $\mathcal{A}'$ in $O(n \log n + K')$ time.

We pick a triple of a concave vertex and two faces from $\mathcal{A}'$. Without loss of generality, assume that the triple is (v_{ENU}, f_W, f_S); v_{ENU} is from ENU, and f_W and f_S are from W and S respectively. The triple induces three wrench sets $\hat{f}_W$, $\hat{f}_S$ and $\mu_1\hat{v}_{ENU,E} + \mu_2\hat{v}_{ENU,N} + \mu_3\hat{v}_{ENU,U}$. Let $h_E = \hat{v}^h_{ENU,E}$, $h_N = \hat{v}^h_{ENU,N}$, $h_U = \hat{v}^h_{ENU,U}$, $h_W \in \hat{f}^h_W$ and $h_S \in \hat{f}^h_S$. When $\mu_3 = 0$, there exist α_1, α_2, α_3 and α_4 in the intervals of 0 and 1, such that $\alpha_1 h_E + \alpha_2 h_N + \alpha_3 h_W + \alpha_4 h_S = \vec{0}$, where $\alpha_1 + \alpha_2 = 1$ and $\alpha_3 + \alpha_4 = 1$, because $(v_{ENU}, f_W, f_S) \in \mathcal{A}'$. This implies that μ_1 and μ_2 can be any number, as long as $\mu_1 : \mu_2 = \alpha_1 : \alpha_2$. Since $h_U = \vec{0}$, μ_3 can also be any number as long as $0 \leq \mu_3 < 1$. Thus we set $\mu_1 = \kappa_1\alpha_1$, $\mu_2 = \kappa_1\alpha_2$ and $\mu_3 = \kappa_2$.[24] Then we get

[22] In fact, this set can have one concave vertex and three faces, because two fingers are allowed to be on one face.

[23] Note that $\hat{v}_{ENU,E}$, $\hat{v}_{ENU,N}$ and $\hat{v}_{ENU,U}$ are three wrench points.

[24] Observe that $\kappa_1\alpha_1$, $\kappa_1\alpha_2$ and κ_2 are in the intervals of 0 and 1, and $\kappa_1\alpha_1 + \kappa_1\alpha_2 + \kappa_2 = 1$.

$\kappa_1(\alpha_1 h_E + \alpha_2 h_N) + \kappa_2 h_U + \alpha_3 h_W + \alpha_4 h_S = \vec{0}$ for $0 < \kappa_1 \leq 1$, $0 \leq \kappa_2 < 1$,[25] where $\kappa_1 + \kappa_2 = 1$. We define Δ to be the set of points in $\mathcal{V}$ space induced by three fingers at v, and anywhere on f_W and f_S. The set Δ has the following form:

$$\begin{aligned}\Delta \;:=\; & \{\kappa_1(\alpha_1\nu_E + \alpha_2\nu_N) + \kappa_2\nu_U + \alpha_3\nu_W + \alpha_4\nu_S \mid \\ & \nu_E = \hat{v}^{\nu}_{ENU,E}, \nu_N = \hat{v}^{\nu}_{ENU,N}, \nu_U = \hat{v}^{\nu}_{ENU,U}, \nu_W \in \hat{f}^{\nu}_W, \nu_S \in \hat{f}^{\nu}_S, \\ & 0 < \kappa_1 \leq 1, 0 \leq \kappa_2 < 1, \kappa_1 + \kappa_2 = 1\}.\end{aligned}$$

The set Δ consists of vectors $(\kappa_2, -\alpha_2\kappa_1 p_{zn} + \kappa_2 p_{yu} + \alpha_4 p_{zs}, \alpha_1\kappa_1 p_{ze} - \kappa_2 p_{xu} - \alpha_3 p_{zw})^T = (\kappa_2, -\alpha_2\kappa_1 p_{zn} + \kappa_2 p_{yn} + \alpha_4 p_{zs}, \alpha_1\kappa_1 p_{zn} - \kappa_2 p_{xn} - \alpha_3 p_{zw})^T$, where p_{zn}, p_{xn}, p_{yn} are fixed numbers, and p_{zs} and p_{zw} are in some ranges.

The finger positions on v_{ENU}, f_W and f_S determine the coefficients α_1, α_2, α_3 and α_4; only κ_1 and κ_2 are variable. Hence the three fingers at fixed positions induce a line segment δ in $\mathcal{V}$ space; the extreme values of κ_1 and κ_2 determine the endpoints of δ. Observe that an endpoint of Δ is on the plane $\eta_z = 0$, (the endpoint for the case when $\kappa_1 = 1$ and $\kappa_2 = 0$). When a finger is at v_{ENU} and the two fingers move in the interior of f_W and f_S independently, the corresponding segment δ also moves in a certain region in $\mathcal{V}$ space. This region is Δ, and it is a convex polytope, a face of which lies on the plane $\eta_z = 0$. The projection of Δ is a polygon, an edge of which lies on the line $\eta_z = 0$ on Γ_ν. When the vertex is from other families, the shapes of Δ and $\pi(\Delta)$ are similar to these described above.

In phase II, we wish to report all triangle pairs from $\pi(\hat{f}^{\nu}_U)$ and $\pi(\hat{f}^{\nu}_D)$ with a set of points satisfying the second condition of Lemma 8.2 with each set of $\mathcal{A}'$. The phase II of the algorithm for the case of a horizontal edge and five faces in Section 8.2.2 can report all such triangle pairs, thus $\mathcal{A}$ in $O(n^2 \log^2 n + nK' \log^4 n + K)$ time; there are $O(n)$ faces.

Theorem 8.13 *Then all K sets of one concave vertex and four faces of P that yield form-closure grasps with five frictionless point fingers can be enumerated in $O(n^2 \log^2 n + nK' \log^4 n + K)$ time, where $K' = |\mathcal{A}|$.*

Two concave vertices and one face

We wish to report all sets of two concave vertices and one face that yield form-closure grasps with three frictionless point fingers; C is two concave vertices and one face. Remember that each set of $\mathcal{A}$ reported by the algorithms presented in this section yields a form-closure grasp with three fingers, and two of the three fingers induce two face normals of a family of U or D. To identify all sets of two concave vertices and a face that allow a form-closure grasp, we apply the algorithms to P and rotated P with Rot_1 and Rot_2. The following lemma shows that the algorithms on P and rotated P with Rot_1 and Rot_2 can report all sets of two concave vertices and a face with form-closure grasps.

Lemma 8.14 *When we rotate a set of two concave vertices and a face that yields a form-closure grasp, with Rot_1 or Rot_2, the rotated set becomes a set of two concave vertices and a face, such that it induces two face normals of a family of U or D.*

Proof: We take any set of two concave vertices and a face of P that yields a form-closure grasp with five fingers. If it induces two face normals of a family U or D (these face normals could be induced by two fingers at the vertices), then we are done. Hence we will look at the cases when it induces two face normals of a family of E, W, N or S.

[25] Since $\kappa_1(\alpha_1 h_E + \alpha_2 h_N)$ must be a non-zero vector, κ_1 must be non-zero, thus κ_2 must not be 1.

Without loss of generality, assume that the set induces two face normals of E. When we rotate a concave vertex with Rot_1, it is still incident to three faces: one from E or W, one from N or S, and one from U or D. With Rot_1, the two face normals of E will be two face normals of D. □

A face must be horizontal, if the face and two concave vertices yield a form-closure grasp and they induce two face normals of a family U or D. Thus C' is two concave vertices, because they can have a set of points satisfying the first condition of Lemma 8.2. There are m concave vertices, therefore, the algorithm proposed in Section 3.3.3 can report $\mathcal{A}'$ in $O(m \log^2 m + K')$ time.

We pick a vertex pair from $\mathcal{A}'$. Without loss of generality, assume that the pair is (v_{ENU}, v_{WSD}); v_{ENU} is from ENU and v_{WSD} is from WSD. Let $h_E = \hat{v}^h_{ENU,E}$, $h_N = \hat{v}^h_{ENU,N}$, $h_U = \hat{v}^h_{ENU,U}$, $h_W = \hat{v}^h_{WSD,W}$, $h_S = \hat{v}^h_{WSD,S}$ and $h_D = \hat{v}^h_{WSD,D}$. With the arguments in the case of one concave vertex and four faces in Section 8.2.3, there exist α_1, α_2, α_3, α_4 in the intervals of 0 and 1, such that $\kappa_1(\alpha_1 h_E + \alpha_2 h_N) + \kappa_2 h_U + \gamma_1(\alpha_3 h_W + \alpha_4 h_S) + \gamma_2 h_D = \vec{0}$ for $0 < \kappa_1, \gamma_1 \leq 1$ and $0 \leq \kappa_2, \gamma_2 < 1$, where $\kappa_1 + \kappa_2 = 1$, $\gamma_1 + \gamma_2 = 1$, $\alpha_1 + \alpha_2 = 1$ and $\alpha_3 + \alpha_4 = 1$. Two fingers at v_{ENU} and v_{WSD} induce the whole set of points of Δ in $\mathcal{V}$ space, which has the following form:

$$\begin{aligned}\Delta \;:=\; \{&\kappa_1(\alpha_1\nu_E + \alpha_2\nu_N) + \kappa_2\nu_U + \gamma_1(\alpha_3\nu_W + \alpha_4\nu_S) + \gamma_2\nu_D \mid \\ &\nu_E = \hat{v}^\nu_{ENU,E}, \nu_N = \hat{v}^\nu_{ENU,N}, \nu_U = \hat{v}^\nu_{ENU,U}, \nu_W = \hat{v}^\nu_{WSD,W}, \nu_S = \hat{v}^\nu_{WSD,S}, \\ &\nu_D = \hat{v}^\nu_{WSD,D}, 0 < \kappa_1, \gamma_1 \leq 1, 0 \leq \kappa_2, \gamma_2 < 1, \kappa_1 + \kappa_2 = 1, \gamma_1 + \gamma_2 = 1\}.\end{aligned}$$

Then Δ consists of vectors $(\kappa_2 - \gamma_2, -\alpha_2\kappa_1 p_{zn} + \kappa_2 p_{yu} + \alpha_4\gamma_1 p_{zs} - \gamma_2 p_{yd}, \alpha_1\kappa_1 p_{ze} - \kappa_2 p_{xu} - \alpha_3\gamma_1 p_{zw} + \gamma_2 p_{xd})^T = (\kappa_2 - \gamma_2, -\alpha_2\kappa_1 p_{zn} + \kappa_2 p_{yn} + \alpha_4\gamma_1 p_{zs} - \gamma_2 p_{ys}, \alpha_1\kappa_1 p_{zn} - \kappa_2 p_{xn} - \alpha_4\gamma_1 p_{zs} + \gamma_2 p_{xs})^T$. Observe that the position vectors are all fixed. This determines α_1, α_2, α_3 and α_4 as well, hence only κ_1, κ_2, γ_1 and γ_2 are variable. When κ_1, κ_2, γ_1 and γ_2 have the extreme values 0 and 1, they make the four vertices of Δ—Δ is a tetrahedron. The projection of Δ on Γ_ν is a convex quadrilateral in general. Note that Δ and $\pi(\Delta)$ for the other vertex pairs from $\mathcal{A}'$ are similar to these described above.

In phase II, we wish to find all triangles from $\pi(\hat{f}^\nu_U)$ and $\pi(\hat{f}^\nu_D)$ with a set of points satisfying the second condition of Lemma 8.2 with each set of $\mathcal{A}'$. The phase II of the algorithm for the case of two horizontal edges and three faces in Section 8.2.2 can identify all such triangles in $O(n^2 \log^2 n + K' \log^4 n + K)$ time; there are $O(n)$ triangles of $\pi(\hat{f}^\nu_U)$ and $\pi(\hat{f}^\nu_D)$.

When m is sufficiently small, we can find $\mathcal{A}$ in a brute-force manner. When $m < n^{1/2}$, we report $\mathcal{A}$ in $O(m^2 n)$ time.

Theorem 8.15 *All K sets of two concave vertices and one face of P that allow form-closure grasps with three frictionless point fingers can be enumerated in:*

1. *$O(m^2 n)$ time, when $m < n^{1/2}$,*
2. *$O(n^2 \log^2 n + K' \log^4 n + K)$ time ($K' = |\mathcal{A}|$), when $n^{1/2} \leq m \leq n$,*

8.2.4 Combinations of concave vertices, concave edges and faces

Let n, t and m be the numbers of faces, concave edges and concave vertices of a rectilinear polyhedron P. In this section, we report all sets of one concave vertex, one concave edge and two faces that allow form-closure grasps with four frictionless point fingers. We do not consider one concave vertex and two edges, because Lemma 8.2 does not provide an efficient algorithm for this case.

We wish to report all sets of one concave vertex, one concave edge and two faces[26] that allow form-closure grasps with four frictionless point fingers. Remember that each set of $\mathcal{A}$ reported by

[26] In fact, this set may have one concave vertex, one concave edge and one face, because two fingers are allowed to be on one face.

the algorithms presented in this section yields a form-closure grasp with four fingers, and two of the four fingers induce two face normals of a family of U or D. To identify all sets of one concave vertex, one concave edge and two faces that allow a form-closure grasp, we apply the algorithms to P and rotated P with Rot_1 and Rot_2. The following lemma shows that the algorithms on P and rotated P with Rot_1 and Rot_2 can report all sets of one concave vertex, one concave edge and two faces with form-closure grasps.

Lemma 8.16 *When we rotate a set of one concave vertex, one concave edge and two faces that yields a form-closure grasp, with Rot_1 or Rot_2, the rotated set becomes one of the two cases:*

(i) one concave vertex, one vertical concave edge and two horizontal faces,

(ii) one concave vertex, one horizontal concave edge, one vertical face and one horizontal face.

Proof: We take any set of one concave vertex, one concave edge and two faces of P that yields a form-closure grasp with four fingers. If two of the four fingers induce two face normals of a family U or D (the face normals could be induced by the finger on the edge), then we are done. Hence we will look at the cases when two fingers induce two face normals of a family E, W, N or S.

Without loss of generality, assume that two fingers induce two face normals of E. When we rotate a concave vertex with Rot_1, it is still incident to three faces: one from E or W, one from N or S, and one from U or D. When we rotate a vertical edge with Rot_1, it becomes horizontal. When we rotate a horizontal edge with Rot_1, it either remains horizontal or becomes vertical. See Table 8.1. Observe that the rotated set involves three face normals of D and U. When the edge is vertical, only one of these three face normals is induced by a finger at the vertex. This implies that the two faces are horizontal, which leads to case (i). When the edge is horizontal, two of these three face normals are induced by two fingers at the vertex and on the horizontal edge. This implies that one face is horizontal, and another face is vertical, which leads to case (ii). □

When the edge is vertical, a vertex and an edge are necessary and sufficient to have a set of points that satisfy the first condition of Lemma 8.2; C' is a vertex and an edge. There are m vertices and t edges. Therefore, the algorithm proposed in Section 3.3.3 can report $\mathcal{A}'$ in $O(\max(m,t)\log^2\max(m,t)+K')$ time.

We pick a pair of a concave vertex and a vertical concave edge from $\mathcal{A}'$. Without loss of generality, assume that the pair is (v_{ENU}, e_{WS}); v_{ENU} is from ENU and e_{WS} is from WS. We define Δ to be the set of points in $\mathcal{V}$ space induced by the two fingers at v_{ENU} and anywhere on e_{WS}. With the arguments in the case of a horizontal concave edge and four faces in Section 8.2.2 and in the case of one concave vertex and four faces in Section 8.2.3, we have the following formula for Δ:

$$\begin{aligned}\Delta \;:=\; &\{\kappa_1(\alpha_1\nu_E+\alpha_2\nu_N)+\kappa_2\nu_U+\alpha_3\nu_W+\alpha_4\nu_S \mid \\ &\nu_E=\hat{v}^{\nu}_{ENU,E}, \nu_N=\hat{v}^{\nu}_{ENU,N}, \nu_U=\hat{v}^{\nu}_{ENU,U}, \nu_W\in\hat{e}^{\nu}_{WS,W}, \nu_S\in\hat{e}^{\nu}_{WS,S} \\ &0<\kappa_1\le 1, 0\le\kappa_2<1, \kappa_1+\kappa_2=1\}.\end{aligned}$$

Since $\pi(\hat{v}^h_{ENU})$ and $\pi(\hat{e}^h_{WS})$ are line segments on Γ_h, α_1, α_2, α_3 and α_4 are fixed numbers. The set Δ consists of vectors $(\kappa_2, -\alpha_2\kappa_1 p_{zn}+\kappa_2 p_{yu}+\alpha_4 p_{zs}, \alpha_1\kappa_1 p_{ze}-\kappa_2 p_{xu}-\alpha_3 p_{zw})^T = (\kappa_2, -\alpha_2\kappa_1 p_{zn}+\kappa_2 p_{yn}+\alpha_4 p_{zs}, \alpha_1\kappa_1 p_{zn}-\kappa_2 p_{xn}-\alpha_3 p_{zs})^T$. Note that p_{zs} is in a range, and p_{xn}, p_{yn} and p_{zn} are fixed numbers. The equality holds because one finger at v_{ENU} induces three face normals of E, N and U, and another finger on e_{WS} induces two face normals of S and W.

The finger position at e determines the coefficients α_1, α_2, α_3 and α_4; only κ_1 and κ_2 are variable. The two fingers at v_{ENU} and anywhere on e_{WS} induce a line segment δ in $\mathcal{V}$ space, one end point of which lies on the plane $\eta_z=0$. When one of the two fingers move in the interior of

e_{WS}, the corresponding segment δ also moves in a certain region in $\mathcal{V}$ space. This region is Δ, and it is a convex polytope, a side of which lies on the plane $\eta_z = 0$ in $\mathcal{V}$ space. The projection $\pi(\Delta)$ is a convex polygon with a side on the line $\eta_z = 0$ on Γ_ν. The shapes of Δ and $\pi(\Delta)$ are similar to these described above, when the edge and the vertex are from other families.

In phase II, we wish to report all triangle pairs from $\pi(\hat{f}_U^\nu)$ and $\pi(\hat{f}_D^\nu)$ with a set of points satisfying the second condition of Lemma 8.2 with each set of $\mathcal{A}'$. The phase II of the algorithm for the case of a horizontal edge and five faces in Section 8.2.2 can report all such triangle pairs, thus $\mathcal{A}$ in $O(n^2 \log^2 n + nK' \log^4 n + K)$ time; there are $O(n)$ faces.

When the edge is horizontal, a vertex, an edge and a face are necessary and sufficient to have a set of points that satisfy the first condition of Lemma 8.2; C' is a vertex, an edge and a face. There are $O(n)$ faces, vertices and edges. Therefore, the algorithm proposed in Section 3.3.2 can report $\mathcal{A}'$ in $O(n \log n + K')$ time.

We pick a triple of a concave vertex, a vertical concave edge and a face from $\mathcal{A}'$. Without loss of generality, assume that the triple is (v_{ENU}, e_{SD}, f_W); v_{ENU} is from ENU, e_{SD} is from SD and f_W is from W. We define Δ to be the set of points in $\mathcal{V}$ space induced by the three fingers at v_{ENU} and anywhere on e_{SD} and f_W. With the arguments in the case of a horizontal concave edge and four faces in Section 8.2.2, and in the case of a horizontal edge and five faces in Section 8.2.2, we have the following formula for Δ:

$$\begin{aligned}\Delta \;:=\; \{&\kappa_1(\alpha_1\nu_E + \alpha_2\nu_N) + \kappa_2\nu_U + \alpha_3\nu_W + \alpha_4(\gamma_1\nu_S + \gamma_2\nu_D) \mid \\ &\nu_E = \hat{v}^\nu_{ENU,E}, \nu_N = \hat{v}^\nu_{ENU,N}, \nu_U = \hat{v}^\nu_{ENU,U}, \nu_W \in \hat{f}^\nu_W, \nu_S \in \hat{e}^\nu_{SD,S}, \nu_D \in \hat{e}^\nu_{SD,D}, \\ &0 < \kappa_1, \gamma_1 \leq 1, 0 \leq \kappa_2, \gamma_2 < 1, \kappa_1 + \kappa_2 = 1, \gamma_1 + \gamma_2 = 1\}.\end{aligned}$$

Note that α_1, α_2, α_3, α_4 are in the ranges dictated by the equation $\kappa_1(\alpha_1 h_E + \alpha_2 h_N) + \kappa_2 h_U + \alpha_3 h_W + \alpha_4(\gamma_1 h_S + \gamma_2 h_D) = \vec{0}$, where $h_E = \hat{v}^h_{ENU,E}$, $h_N = \hat{v}^h_{ENU,N}$, $h_U = \hat{v}^h_{ENU,U}$, $h_W \in \hat{f}^h_W$, $h_S \in \hat{e}^h_{SD,S}$ and $h_D \in \hat{e}^h_{SD,D}$. The set Δ consists of vectors $(\kappa_2 - \alpha_4\gamma_2, -\alpha_2\kappa_1 p_{zn} + \kappa_2 p_{yu} + \alpha_4\gamma_1 p_{zs} - \alpha_4\gamma_2 p_{yd}, \alpha_1\kappa_1 p_{ze} - \kappa_2 p_{xu} + \alpha_4\gamma_2 p_{xd} - \alpha_3 p_{zw})^T = (\kappa_2 - \alpha_4\gamma_2, -\alpha_2\kappa_1 p_{zn} + \kappa_2 p_{yn} + \alpha_4\gamma_1 p_{zs} - \alpha_4\gamma_2 p_{ys}, \alpha_1\kappa_1 p_{zn} - \kappa_2 p_{xn} + \alpha_4\gamma_2 p_{xs} - \alpha_3 p_{zw})^T$. Note that p_{zw} and p_{xs} are in some open ranges, and p_{xn}, p_{yn}, p_{zn}, p_{ys} and p_{zs} are fixed numbers. The equality holds because one finger at v_{ENU} induces three face normals from E, N and U, and another finger on e_{SD} induces two face normals of S and D.

The finger positions on v_{ENU}, e_{SD} and f_W determine α_1, α_2, α_3 and α_4; only κ_1, κ_2 and γ_1, γ_2 are variable. Thus the three fingers (at fixed positions) induce a tetrahedron δ in $\mathcal{V}$ space, a vertex of which lies on the plane $\eta_z = 0$; when $\kappa_2 - \alpha_3\gamma_2 = 0$. When two of the three fingers move in the interior of e_{SD} and f_W independently, the corresponding tetrahedron δ also moves in a certain region in $\mathcal{V}$ space. This region is Δ, and it is a convex polytope in $\mathcal{V}$ space, one face of which lies on the plane $\eta_z = 0$. The projection $\pi(\Delta)$ is a convex polygon, a side of which lies on the line $\eta_z = 0$ on Γ_ν. The shapes of Δ and $\pi(\Delta)$ are similar to these described above, when the vertex, the edge and the face are from other families.

In phase II, we wish to report all triangles from $\pi(\hat{f}_U^\nu)$ and $\pi(\hat{f}_D^\nu)$ with a set of points satisfying the second condition of Lemma 8.2 with each set of $\mathcal{A}'$. The phase II of the algorithm for the case of two horizontal edges and three faces in Section 8.2.2 can identify all such triangles in $O(n^2 \log^2 n + K' \log^4 n + K)$ time; there are $O(n)$ triangles of $\pi(\hat{f}_U^\nu)$ and $\pi(\hat{f}_D^\nu)$.

The following theorem summarizes the result.

Theorem 8.17 *All K sets of one concave vertex, one concave edge and two faces of P that allow form-closure grasps with four frictionless point fingers can be enumerated in $O(n^2 \log^2 n + nK' \log^4 n + K)$ time, where $K' = |\mathcal{A}|$.*

8.3 Conclusion

This work is the first step to provide efficient ways of reporting all form-closure grasps of a three-dimensional object. The form-closure condition in six-dimensional wrench space is converted into two subproblems in three-dimensional space, which are closely related. These subproblems are again transformed into two-dimensional intersection search problems. The two subproblems correspond to the immobilizations of the object against the horizontal movements, and then against the vertical movements. Because of the nature of a rectilinear polyhedron, this formulation produced efficient algorithms for a rectilinear polyhedron.

This formulation, unfortunately, does not lead to attractive efficient output-sensitive algorithms for other kinds of three-dimensional objects; the difficulty lies in combining the two subproblems with arbitrary normal directions. It is open to efficiently report all form-closure grasps on any three-dimensional object. But it is already a great challenge to design an efficient output-sensitive algorithm to report all combinations of faces, concave edges and concave vertices of any polyhedron, such that fingers on the set yield at least one form-closure grasp.

All algorithms presented in this chapter except one are sensitive to both K' and K, where K' and K are the sizes of the intermediate output and the final output respectively; the algorithm to compute all sets of three horizontal concave edges and a face is sensitive to K' only. It remains open to design an efficient algorithm to report all sets of three horizontal concave edges and a face of a rectilinear polyhedron that is sensitive to K' and K.

Another attractive characteristics of a grasp is insensitivity to small misplacements of the fingers or minor shape variations of the objects. In such a setting, we should also be able to guarantee insensitivity to minor variations in the directions of the surface normals or their locations. Although such an extension is definitely challenging, we believe that the insights presented in this chapter could be of use to attack those problems as well.

Chapter 9

Immobilizing Hinged Polygons

Most of the existing results on immobilization apply to rigid bodies, and none of the above results is about immobilizing non-rigid objects. As a first step in this direction, we study immobilization of a serial chain of polygons connected by hinges. This can be seen as a case study of immobilization of non-rigid objects. A hinge allows the two adjacent polygons to rotate around it. We shall assume that the hinges are located at vertices. Our aim is to immobilize any serial chain of n hinged polygons in a priorly specified placement, with minimum number of fingers.[1]

We analyze motions by identifying the areas where a vertex of a polygon can move locally from a given configuration with given set of point fingers on the boundary. Our analysis is based on curvature effects, precisely as the analysis of Czyzowicz et al. [33] is on curvature effects. We show that $(n+2)$ frictionless point fingers suffice to immobilize any serial chain of $n \neq 3$ polygons without parallel edges in a given placement; it is unclear whether five fingers can achieve it. We observe that the number of fingers required to immobilize a serial chain of n polygons equals the number of degrees of freedom of the chain. Note that the number of hinges h equals to $(n-1)$, and that the degrees of freedom of a serial chain is $(n+2)$, which is $(h+3)$. All the proofs are constructive in the sense that we give actual configurations with $(n+2)$ fingers for chains of $n \neq 3$ polygons. Allowing for parallel edges leads to an increase in the number of fingers of at most one.

One observation about the first-order immobility is that any perturbation of any combination of the $(n+2)$ fingers along the object's boundary maintains the immobility. This has motivated us to also investigate the number of point fingers required to obtain a more robust immobilization, which has the property—like form closure—that any finger can be perturbed slightly along the edges without destroying the immobility. We construct finger configurations for robust immobility for a serial chain of n polygons with $\lceil \frac{6}{5}(n+2) \rceil$ fingers if the polygons have no parallel edges, and with $\lceil \frac{5}{4}(n+2) \rceil$ fingers if the polygons are allowed to have parallel edges. Informally speaking, we achieve robustness at the cost of one additional finger per twenty polygons.

This chapter is structured as follows. We first introduce the concept of immobility and robust immobility in Section 9.1. In Section 9.2, we show how we immobilize a serial chain of polygons with $n+2$ fingers. Section 9.2.1 is about constructing an immobility grasp for a chain of polygons without parallel edges, and Section 9.2.2 is for polygons with parallel edges. In Section 9.3, we show how we robustly immobilize a serial chain of polygons. Section 9.3.1 is about constructing a robust immobility grasp for a chain of polygons without parallel edges, and Section 9.3.2 is for polygons with parallel edges. Section 9.4 is about a variation of hinged polygons: constructing an immobility and robust immobility grasp for a cycle of hinged polygons and a chain of hinged

[1] This chapter is based on "Fixturing hinged parts" [22] by J.-S. Cheong, K. Goldberg, M.H. Overmars and A.F. van der Stappen in ICRA (2002), and "Immobilizing hinged parts" [26] by J.-S. Cheong, A.F. van der Stappen, K. Goldberg, M.H. Overmars and E. Rimon, which will appear in the International Journal on Computational Geometry and Applications.

polygons with one vertex attached to a wall. In Section 9.5, we discuss the work presented in this chapter.

9.1 Immobility and robust Immobility

In this section, we introduce notations, and discuss our notions of immobility and robust immobility. In addition, we report some known results on immobilization of a single polygon.

Let $(P_1, P_2, \cdots, P_n)$ be a serial chain of n hinged polygons. Each polygon P_i in the chain shares a vertex—the hinge—with its successor P_{i+1}; we denote the hinge connecting P_i and P_{i+1} by v_i. A hinge v_i allows the adjacent polygons P_i and P_{i+1} to rotate relative to each other. It is our aim to study how many frictionless point fingers along the boundaries are sufficient to immobilize the chain in a given placement, or at a configuration q. We assume that the two edges of polygon P_i incident to its hinge v_i are not collinear. In addition, we assume that the polygons including the boundaries are strictly disjoint except at the hinges. The finger arrangement for $(P_1, P_2, \cdots, P_n)$ is represented as $(\mathbf{n}_1, \mathbf{n}_2, ..., \mathbf{n}_n)$, where $\mathbf{n}_i$ is the number of fingers placed on P_i.

A set of point fingers immobilizes the chain $(P_1, P_2, \cdots, P_n)$ at configuration q, if these fingers prevent the chain from leaving q. In other words, there exists no free continuous motion from q to a neighbor configuration q'. Showing that an object is immobilized (at an isolated configuration q) involves considering the curvatures of potential motions, which are dictated by the shape of the object and the fingers. Czyzowicz et al's notion of immobility [33] and second-order immobility [75] uses this, while form closure [70, 58, 46] and first-order immobility [74] does not need to consider these details. In other words, an object in form closure or first-order immobility maintains the immobility regardless of the shape of the object and the fingers. Our notion of immobility takes the curvature of potential motions into account. Contrary to Rimon and Burdick, who carry out their analysis in configuration space, we perform our analysis on the plane of the chain itself, as Czyzowicz et al. To show that a chain is immobilized, we use an intuitive two-step analysis. The first step is to show that none of the hinges v_i can move. Then we show that P_1 cannot rotate around v_1, and that P_n cannot rotate around v_{n-1}; P_i $(i = 2, \cdots, n-1)$ is immobilized because two points of P_i, v_{i-1} and v_i are fixed.

The immobility of a chain is analyzed by looking at the *free areas* of some vertices. *Free area* is where a vertex of a polygon P can locally move around, when P is held with some fingers. There are two different free areas depending on the nature (convex or concave) of the vertices. Let ϑ be the angle at a vertex v of P. Let $\mathcal{C}(p, p', p'')$ be the unique circle defined by three non-collinear points p, p' and p''. We denote the interior including the boundary of $\mathcal{C}(p, p', p'')$ by $\mathcal{C}^+(p, p', p'')$, and the exterior including the boundary of $\mathcal{C}(p, p', p'')$ by $\mathcal{C}^-(p, p', p'')$. The following lemma describes the behavior of a vertex v of a free polygon P, when two point fingers p_1 and p_2 are placed along two adjacent edges e_1 and e_2 respectively. It is a generalization of the result on page 61–62 in [10].

Lemma 9.1 *Any motion of P causes v to initially move into $\mathcal{C}^+(v, p_1, p_2)$ when v is convex, and $\mathcal{C}^-(v, p_1, p_2)$ when v is concave.*

Proof: When v is a convex vertex, we assume that v can reach outside of $\mathcal{C}(v, p_1, p_2)$ by translation and rotation from the current position, under the restriction of p_1 and p_2. Let v'' be the point outside of $\mathcal{C}^+(v, p_1, p_2)$ as in Figure 9.1 (a). Let v' be the intersection point of the line $\overline{p_1 v''}$ and $\mathcal{C}(v, p_1, p_2)$. It is a well known geometrical fact that the angle $\angle p_1 v' p_2 = \angle p_1 v p_2 = \vartheta$. A simple trigonometric calculation shows that $\angle p_1 v'' p_2 < \angle p_1 v p_2$, thus $\angle p_1 v'' p_2 < \vartheta$, which is a contradiction. Therefore, v can only move locally in $\mathcal{C}^+(v, p_1, p_2)$.

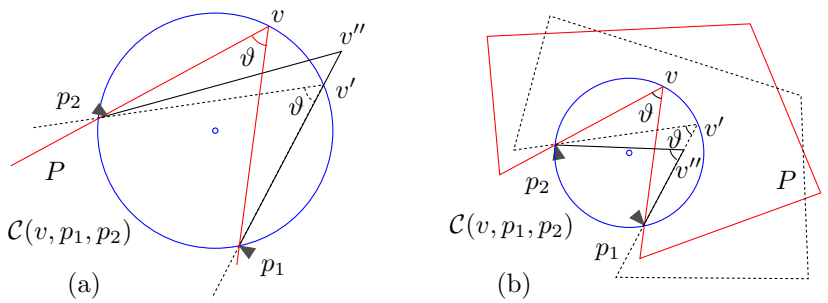


Figure 9.1: The free area where v can locally move around under the restriction of the two fingers p_1 and p_2 is: (a) the interior and the boundary of $\mathcal{C}$ when v is convex, and (b) the exterior and the boundary of $\mathcal{C}$ when v is concave.

When v is a concave vertex, we assume that v can be placed inside of $\mathcal{C}(v, p_1, p_2)$ by translation and rotation from the current position, under the restriction of p_1 and p_2. Let v'' be the point inside of $\mathcal{C}(v, p_1, p_2)$ as in figure 9.1 (b). Let v' be the intersection point of the boundary of $\mathcal{C}(v, p_1, p_2)$ and the supporting line of $\overline{p_1 v''}$. For the same reason described in the previous case, $\angle p_1 v' p_2 = \angle p_1 v p_2 = \vartheta$, thus $\angle p_1 v'' p_2 > \vartheta$, which is a contradiction. Therefore, v can only move locally in $\mathcal{C}^-(v, p_1, p_2)$. □

Besides the fact that first-order immobility (form closure) do not take into account the curvature of potential motions, there is another intuitive and essential difference with second-order immobility (Czyzowicz et al's notion). Any slight perturbation of the frictionless fingers along the edges can maintain first-order immobility, which is highly unlikely for second-order immobility. Our notion of immobility behaves consistently, as it will be easy to see that small perturbations of the fingers destroy the immobility. This motivates us to explore the price of insensitivity to small perturbations. Before we introduce the new notion, we formulate the notion of the immobility of a chain $(P_1, P_2, \cdots, P_n)$: a set of fingers immobilize a chain $(P_1, P_2, \cdots, P_n)$ in its given placement, if the chain cannot change its placement without violating the rigidity of the object and the fingers, or the connectivity of the chain.

Definition 9.1 *A set of point fingers robustly immobilize the chain $(P_1, P_2, \cdots, P_n)$ if these fingers immobilize $(P_1, P_2, \cdots, P_n)$, and if there exists a real number $\epsilon > 0$ for each finger placed along the interior of an edge, such that any perturbation of the finger in the ϵ-interval in both directions along the edge maintains the immobility.*

We end this section by reporting a few simple results. The fairly easy proof of Lemma 9.2 is left to the readers. Lemma 9.3 and 9.4 are from Lemma 9.2 and Markenscoff et al. [51], Mishra et al. [56], Rimon and Burdick [73], van der Stappen et al. [82] and Czyzowicz et al. [33].

Lemma 9.2 *A two dimensional polygon P in form closure is robustly immobilized.*

Lemma 9.3 *Any polygon can be robustly immobilized with four frictionless point fingers in linear time.*

Lemma 9.4 *Any polygon without parallel edges can be immobilized with three frictionless point fingers in linear time.*

9.2 Immobility of a serial chain of hinged polygons

A polygon without parallel edges can be immobilized with three point fingers, while a polygon with parallel edges may need one more finger to be immobilized. Likewise, a serial chain of

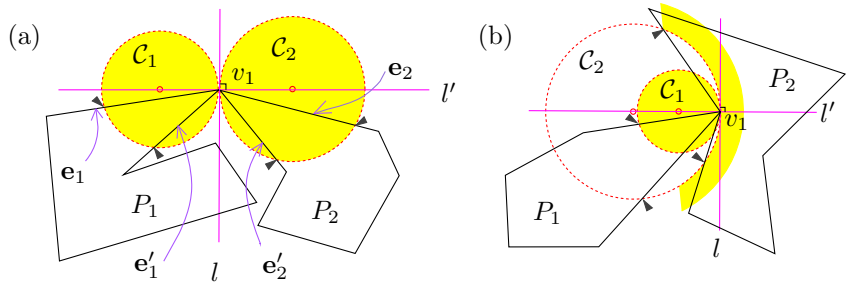


Figure 9.2: Two hinged polygons with four fingers.

n polygons with parallel edges needs more fingers in general. First, we consider immobility of hinged polygons without parallel edges, and then those with parallel edges.

9.2.1 Polygons without parallel edges

We will subsequently discuss the immobilization of serial chains of two, three and four polygons without parallel edges. The immobilities of a single polygon, and of serial chains of two and four polygons serve as building blocks to immobilize longer chains.

Two polygons without parallel edges

We show how *four* fingers can immobilize two hinged polygons P_1 and P_2. At most one polygon can be concave at v_1; the rest are convex at v_1. We first focus on the case when both P_1 and P_2 are convex at v_1 (Figure 9.2 (a)). Let $\mathbf{e}_i$ and $\mathbf{e}'_i$ be the two edges of P_i incident to the hinge v_1. Let l be a line containing v_1 such that $\mathbf{e}_1$ and $\mathbf{e}'_1$ are strictly on one side of l, and that $\mathbf{e}_2$ and $\mathbf{e}'_2$ are strictly on the other side. Let l' be the perpendicular line of l at v_1. Take a circle $\mathcal{C}_i$ for $P_i (i = 1, 2)$, which satisfies the following two conditions:

1. The center of $\mathcal{C}_i$ is on l' so that $\mathcal{C}_i$ touches l at v_1, and
2. $\mathcal{C}_i$ intersects $\mathbf{e}_i$ and $\mathbf{e}'_i$ in their interiors.

Place four fingers at the intersection points of the circles and the polygons.

When one polygon, say P_2, is concave at v_1, the construction is the same as in the previous case, except for a few details. First, the line l is a line through v_1, only one side of which contains all the adjacent edges—see Figure 9.2 (b). Second, $\mathcal{C}_1$ is smaller than $\mathcal{C}_2$. The next lemma shows why these immobilize two hinged polygons.

Lemma 9.5 *Four fingers suffice to immobilize two hinged polygons.*

Proof: The free area of v_1 is $\mathcal{C}^+$ or $\mathcal{C}^-$ according to Lemma 9.1. In any case, the two free areas ($\mathcal{C}_1^+$ and $\mathcal{C}_2^+$ or $\mathcal{C}_1^+$ and $\mathcal{C}_2^-$ or $\mathcal{C}_1^-$ and $\mathcal{C}_2^+$) touch each other at v_1. The position of v_1 is fixed, because it is the only intersection of the free areas. In other words, v_1 cannot move without breaking the fingers or disconnecting the polygons. Four fingers on P_1 and P_2 prohibit he rotations of P_1 and P_2 around v_1. Therefore, four fingers suffice to immobilize two hinged polygons. □

In general, less than four fingers can not immobilize two hinged polygons; in most cases, one finger on P_i cannot prevent the rotation of P_i around the hinge away from the finger.

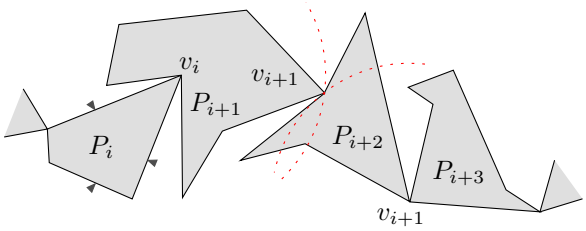


Figure 9.3: A configuration of six fingers for four hinged polygons without parallel edges.

Four polygons without parallel edges

Four polygons without parallel edges can be immobilized with *six* fingers; immobilize the first and the last polygons with three fingers for each. The finger arrangement is $(3, 0, 0, 3)$. The following lemma shows that the two polygons in the middle are immobilized.

Lemma 9.6 *Two adjacent polygons P_2 and P_3 are immobilized if their neighbors P_1 and P_4 are immobilized.*

Proof: Let $\mathcal{C}_1$ and $\mathcal{C}_3$ be the circles around v_1 and v_3 that v_2 follows respectively (Figure 9.3). Since the hinges v_1 and v_3 are in fixed positions, v_2 of P_2 and P_3 can move along the arc of $\mathcal{C}_1$ and $\mathcal{C}_3$ respectively. Only at one intersection point of $\mathcal{C}_1$ and $\mathcal{C}_3$, v_2 can lie such that the distances $|\overline{v_1v_2}|$ and $|\overline{v_2v_3}|$ are preserved at the same time. In other words, the position of v_2 is also fixed. Because v_1 and v_2 of P_2, and v_2 and v_3 of P_3 are fixed, P_2 and P_3 are immobilized. □

Corollary 9.1 *Six point fingers suffice to immobilize a serial chain of four hinged polygons.*

Immobilizing n polygons without parallel edges

Here we immobilize a serial chain of $n \geq 5$ hinged polygons, using the finger arrangements for one polygon, and two and four hinged polygons. From the right end of the chain, cut off a trailing multiple of four polygons, until at most four polygons are left. When three are left, combine them with the next four polygons. Nine fingers with finger arrangement of $(3, 0, 0, 3, 0, 0, 3)$ immobilize these seven polygons. Lemma 9.4, 9.5 and Corollary 9.1 immobilize the remaining one polygon, two or four polygons. Each of the remaining quadruples are immobilized with the arrangement $(0, 0, 2, 2)$. Since the total number equals to the degree of freedom of the whole chain, the number of fingers is tight.

Theorem 9.7 *A serial chain of n $(\neq 3)$ hinged polygons without parallel edges can be immobilized with $(n + 2)$ fingers, which is tight. Six fingers can immobilize three polygons without parallel edges.*

9.2.2 Immobility of hinged polygons with parallel edges

Four fingers are necessary to immobilize a polygon with parallel edges (Lemma 9.3). The immobility for two polygons in Lemma 9.5 still holds when the polygons have parallel edges. Six $(= n+3)$ fingers can immobilize three arbitrary polygons as follows. Immobilize the first two polygons P_1 and P_2 with $(2, 2)$ finger arrangement. The last polygon P_3 will rotate around the hinge v_2. Placing two fingers on the incident edges to v_2 will immobilize P_3. The finger arrangement is $(2, 2, 2)$ (Figure 9.4.)

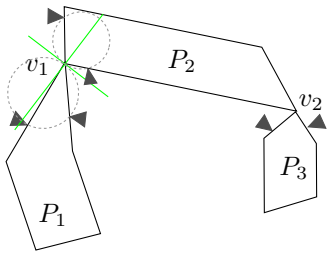


Figure 9.4: A finger arrangement that immobilizes three arbitrary polygons.

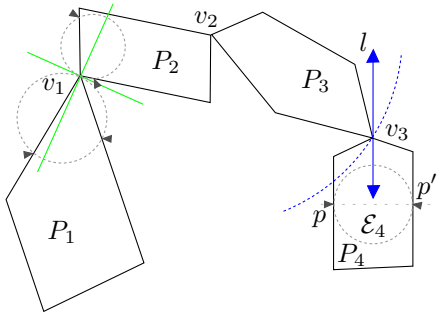


Figure 9.5: Four arbitrary polygons can be immobilized with seven fingers.

Lemma 9.8 *Six fingers suffice to immobilize three polygons with parallel edges.*

Proof: Since P_1 and P_2 are immobilized (Lemma 9.5), the positions of v_1 and v_2 are fixed. The two fingers on P_3 prohibits the rotation of P_3 around v_3. □

Seven $(= n + 3)$ fingers can immobilize four polygons with parallel edges as follows. Immobilize the first two polygons P_1 and P_2 with four fingers. Take a maximal inscribed circle of the last polygon P_4. If the touching points of the circle and P_4 gives an immobility finger arrangement, place fingers at the intersections. Otherwise, place a finger p_1 at one touching point, and p_2 and p_3 on both sides of the other touching point—see P_4 in Figure 9.5.

Lemma 9.9 *Seven fingers suffice to immobilize four polygons with parallel edges.*

Proof: Since P_1 and P_2 are immobilized (Lemma 9.5), the positions of v_1 and v_2 are fixed. If the touching points of a maximal inscribed circle and P_4 gives an immobility finger arrangement, we are done. Otherwise, two of the touching edges are parallel edges. Let e and e' be the parallel edges of P_4 along which the three fingers p_1, p_2 and p_3 are placed. The vertex v_3 of P_3 follows the circular arc $\mathcal{C}_3$ around v_2, while v_3 of P_4 slides along the line l through v_3 that is parallel to e and e'. Only at the intersections of $\mathcal{C}_3$ and l, P_3 preserve the distance $|\overline{v_2v_3}|$, and P_4 touches the fingers at the same time. It is impossible for v_3 to move from one intersection to the other by rotations and translations, so v_3 is fixed. Because v_2 and v_3 are fixed, P_3 is immobilized, and P_4 cannot rotate around v_3 because of the fingers. Therefore, the four polygons are immobilized. □

Now we show how at most $(n + 3)$ fingers immobilize n hinged polygons with parallel edges. From the right end of a serial chain, cut off a trailing multiple of four polygons, until at most four polygons are left. Immobilize these left polygons as described in Lemma 9.5, 9.8 and 9.9; four fingers are required to immobilize any single polygon [51, 56, 73].Immobilize each of the trailing quadruples using the arrangement of $(0, 0, 2, 2)$. Note that the finger arrangement of $(0, 0, 2, 2)$ can still be used, because the finger arrangement of $(2, 2)$ still works for two arbitrary polygons (Lemma 9.5). The number of fingers $(n + 3)$ is tight in the sense that there exist n polygons that

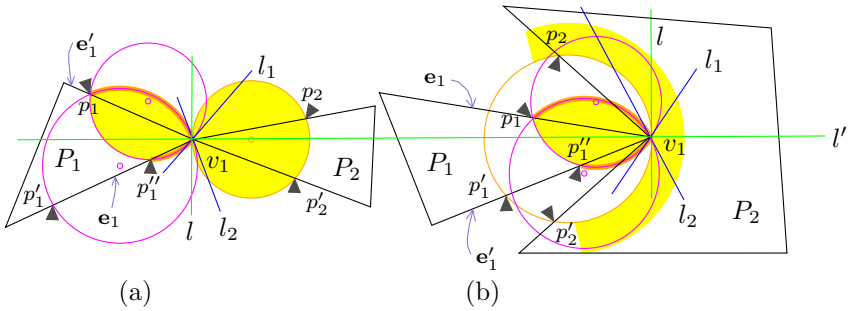


Figure 9.6: Two hinged polygons are robustly immobilized with five fingers, (a) when both of the hinged vertices are convex, and (b) when one hinged vertex is concave.

cannot be immobilized with less than $(n+3)$ fingers. For example, when one polygon, or three or four polygons remain on the left after cutting off multiples of quadruple, $(n+3)$ fingers are indeed necessary to immobilize the chain; when two polygons remain on the left, $(n+2)$ fingers suffice to immobilize the chain.

Theorem 9.10 *At most $(n+3)$ fingers suffice to immobilize a serial chain of n hinged arbitrary polygons.*

9.3 Robust immobility of a serial chain of hinged polygons

As in the case of immobility, we have different results for a serial chain of polygons with, and without parallel edges.

9.3.1 Robust immobility of polygons without parallel edges

The finger arrangements for one polygon, two, three, and four hinged polygons will serve as building blocks to achieve robust immobility. The result of robust immobility for a single polygon is presented in Lemma 9.3. We proceed to show how to robustly immobilize two, three and four polygons.

Two polygons without parallel edges

Here we show how to achieve robust immobility for two hinged polygons with five fingers from Lemma 9.5. Note that both polygons cannot be simultaneously concave at the hinge v_1, i.e., at least one polygon is convex at v_1. Without loss of generality, let P_2 is convex at v_1. Let $\mathbf{e}_2$ and $\mathbf{e}'_2$ be the edges of P_2 incident to v_1, and $\mathbf{e}_1$ and $\mathbf{e}'_1$ be those of P_1 incident to v_1 as in Figure 9.6. Line l is chosen in the same way as in Section 9.2.1. We rotate l around v_1 such that it lies between the original l and the lower edge incident to v_1; we call it l_1. Now we rotate l around v_1 such that it lies between the original l and the upper edge incident to v_1; we call it l_2. Take a circle for P_1 that touches l at v_1 and that intersects $\mathbf{e}_1$ and $\mathbf{e}'_1$ in their interiors. Place two fingers p_4 and p_5 at these intersections. Take a circle for P_2 that touches l_1 at v_1 and that intersects $\mathbf{e}_2$ and $\mathbf{e}'_2$ in their interiors. Place two fingers p_1 and p_2 at the intersections. Take another circle for P_2 that satisfies the following three conditions: (i) it touches l_2 at v_1, (ii) it intersects $\mathbf{e}_2$ and $\mathbf{e}'_2$ in their interiors, and (iii) it passes through one of p_1 and p_2 (in Figure 9.6, it is p_1). Place a finger p_3 at the empty intersection point.

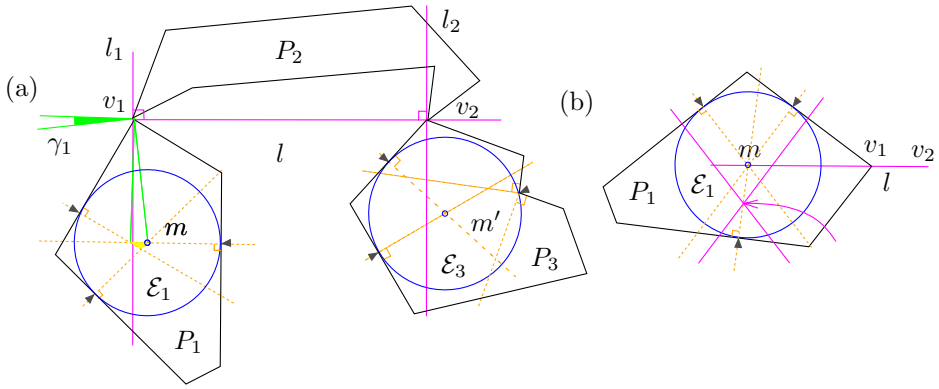


Figure 9.7: (a) A robust immobility of three hinged polygons with six fingers when c_1, v_1 and v_2 are not collinear. (b) When c_1, v_1 and v_2 are collinear, a new position of c_1 can be computed.

Lemma 9.11 *Five fingers suffice to robustly immobilize two hinged polygons.*

Proof: The free area of v_1 induced by p_4 and p_5 on P_1 is either $\mathcal{C}^+(v_1, p_4, p_5)$ or $\mathcal{C}^-(v_1, p_4, p_5)$. In Figure 9.6, the free area of v_1 of P_2 is partially defined by the intersection of $\mathcal{C}^+(v_1, p_1, p_2)$ and $\mathcal{C}^+(v_1, p_1, p_3)$—the thick arcs from the two circles make the partial boundary of this free area. The two free areas touch each other at a single point on their boundaries, which is v_1. For P_1 and P_2 to remain connected, v_1 has to stay at the intersection, which is isolated from other intersections of the free areas. None of the polygons can rotate around v_1 because of the fingers, thus the two hinged polygons are immobilized.

There exists a set of perturbations of all fingers on the polygons such that the induced free areas still touch each other at one isolated point. Therefore, five fingers can robustly immobilize two hinged polygons. □

Three polygons without parallel edges

Three polygons can be robustly immobilized as follows. Compute maximal inscribed circles $\mathcal{M}_1$ and $\mathcal{M}_3$ for P_1 and P_3 respectively, and let c_1 and c_3 be their centers. Let l_1 and l_2 be the perpendicular lines to the line $\overline{v_1v_2}$ at the hinges v_1 and v_2 (Figure 9.7 (a)). For the time being, we have two assumptions for simplicity. First, the touching points of $\mathcal{M}_i$ and P_i are in the interior of the edges (see P_1 in Figure 9.7 (a)). Second, none of c_1 and c_3 are collinear with $\overline{v_1v_2}$.

Since the polygons do not have parallel edges, three fingers can immobilize P_i, and the three normals induced by the fingers meet at one point. If we perturb one finger, the normals form a triangular region, which is a set of rotation centers in either clockwise or counterclockwise direction. Infinitesimal rotations of v_1 of P_1 around a point q in this triangular region move v_1 in a direction along a half-line emanating from v_1 orthogonal to the line $\overline{qv_1}$. All these half-lines lie in a wedge-like region—the shaded region on the left side of l_1 in Figure 9.7 (a). It is important that we can always choose the direction of rotations, so that v_1 move strictly towards the left or the right side of l_1. For P_1 in Figure 9.7 (a), v_1 can be only on the left side of l_1. The two boundary lines of the shaded wedge region are perpendicular to the tangent lines of the triangle through v_1. Notice that the wedge region does not include any point of l_1 except v_1; otherwise, P_1 can rotate around v_2. We construct the same finger arrangement for P_3 so that the wedge region for v_2 lies strictly on the right side of l_2.

Now we remove the first assumption; assume that one of the fingers touches a vertex of a polygon. Without loss of generality, let this polygon be P_3. The vertex must be concave, and three

fingers at the intersection of $\mathcal{M}_3$ and P_3 achieve form closure [81] (Figure 9.7 (a)). If this is the case with P_1 as well, P_1 is in form closure; otherwise, it can be held with three fingers such that v_1 can move away from P_3, thus towards the left side of l_1 (Figure 9.7 (a)).

To remove the second assumption, assume that one of c_1 or c_3, say c_1, is collinear with $\overline{v_1v_2}$. If one of the intersection points of $\mathcal{M}_1$ and P_1 is a vertex, it is concave, and the collinearity does not affect the form closure. So assume that $\mathcal{M}_1$ touches P_1 in the interior of some edges. The meeting point of the three normals can be perturbed by moving two normals together along the third one (Figure 9.7 (b)). After perturbing the meeting point c_1, we can use the same method described before.

Lemma 9.12 *Six fingers suffice to robustly immobilize three hinged polygons without parallel edges.*

Proof: When a polygon is in form closure with three fingers, the proof is rather straightforward. Thus we consider the other case when none of the polygons are in form closure. To maintain the distance $\overline{v_1v_2}$ of P_2, v_1 and v_2 should stay at the apexes of the wedges. Since P_1 and P_3 cannot rotate around v_2 or v_1, they are immobilized. There exists a set of perturbation intervals of the fingers, such that either it still keeps form closure of a polygon, or an induced wedge-like region still stays on the same side of its corresponding line. This concludes the proof. □

Four polygons without parallel edges

Eight fingers can robustly immobilize four hinged polygons as follows: robustly immobilize the first and the last polygons with four fingers each. The finger arrangement is $(4, 0, 0, 4)$.

Lemma 9.13 *Two adjacent polygons P_2 and P_3 are robustly immobilized if P_1 and P_4 are robustly immobilized.*

Proof: The two polygons in the middle are immobilized by Lemma 9.13. Since the neighbors are robustly immobilized, the whole chain is robustly immobilized. □

Five polygons without parallel edges

The construction for five hinged polygons is a variation of that for four polygons: robustly immobilize the first polygon with four fingers, and the last two polygons with five fingers. The finger arrangement is $(4, 0, 0, 3, 2)$.

Lemma 9.14 *Nine fingers suffice to robustly immobilize five hinged polygons without parallel edges.*

Proof: Since the first and the last polygons are in robust immobility (Lemma 9.3 and 9.11), the whole chain of polygons is robustly immobilized according to Lemma 9.13. □

Robust immobility of n polygons without parallel edges

The following is how to robustly immobilize $n \geq 6$ polygons. From the right end of the chain, cut off a trailing multiple of five polygons, until at most five polygons are left. These left polygons can be robustly immobilized as in [56, 51, 58] and in Lemma 9.11, 9.12, 9.13, and 9.14. Each group of five polygons can be immobilized with the finger arrangement of $(0, 0, 3, 0, 3)$, where $(3, 0, 3)$ is the construction in Lemma 9.12.

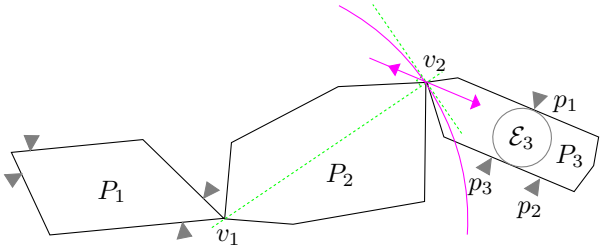


Figure 9.8: Seven fingers can immobilize three arbitrary polygons.

Theorem 9.15 *A serial chain of n hinged polygons without parallel edges can be robustly immobilized with $\lceil \frac{6}{5}(n+2) \rceil$ fingers.*

9.3.2 Robust immobility of polygons with parallel edges

The building blocks to robustly immobilize arbitrary polygons are Lemmas 9.3, 9.11, 9.12 and 9.13. Except the finger arrangement of $(3, 0, 3)$ for three polygons (Lemma 9.12), all of these can be used for polygons with parallel edges without any modification. We proceed to show how to modify the construction in Lemma 9.12 to robustly immobilize three polygons.

The finger arrangement of $(4, 0, 3)$ robustly immobilizes three hinged polygons as follows. Robustly immobilize the first polygon P_1 with four fingers. Take a maximal inscribed circle $\mathcal{C}_3$ for P_3. If it allows a finger arrangement for immobility by placing fingers at the touching points, we can use Lemma 9.12. Otherwise, $\mathcal{C}_3$ touches P_3 at parallel edges. Place a finger p_1 at one touching point, and p_2 and p_3 on both sides of the other touching point (see P_3 in Figure 9.8).

Lemma 9.16 *Seven point fingers can robustly immobilize three arbitrary polygons.*

Proof: When $\mathcal{C}_3$ touches P_3 on parallel edges, the chain is immobilized with the same argument as in Lemma 9.9, so we show that the immobility is robust. The first polygon is in robust immobility. We can perturb the three fingers on P_3 along the parallel edges, without changing the line along which v_2 moves. This concludes the proof. □

Now we show how to robustly immobilize $n \geq 5$ arbitrary polygons. From the right end of the chain, cut off a trailing multiple of four polygons, until at most four polygons are left. Each quadruple can be robustly immobilized by the finger arrangement of $(0, 0, 3, 2)$. The remaining polygon(s) can be robustly immobilized as described above with four, five, seven and eight fingers respectively.

Theorem 9.17 *A serial chain of n arbitrary polygons can be robustly immobilized with $\lceil \frac{5}{4}(n+2) \rceil$ fingers.*

9.4 Immobilizing other types of hinged polygons

Hinged polygons may not form a serial chain. It can be a single cycle, a tree, a general graph, or a serial chain with one vertex of a polygon at the end attached to a wall. Here we consider the cases when the hinged polygons form a single cycle and when an end vertex of the chain is attached to a wall. The previous results can easily be used for these cases.

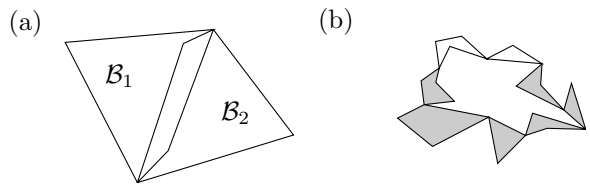


Figure 9.9: (a) Two polygons forming a cycle. (b) A cycle with more than two polygons.

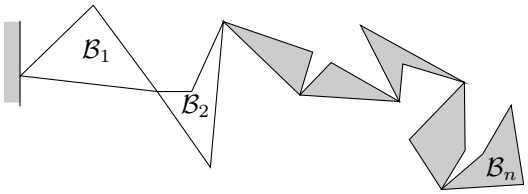


Figure 9.10: A chain of n polygons one end of which is attached to a wall.

9.4.1 A cycle of hinged polygons

First let's look at the case of a cycle. A cycle needs at least two polygons (Figure 9.9). Two polygons forming a cycle can be considered as one polygon, hence three and four point fingers can immobilize or robustly immobilize them.

When a cycle contains three or more polygons, this can be divided into two groups: any two adjacent polygons, and the rest (the rest are shaded in Figure 9.9 (b). The rest polygons can be seen as a free serial chain of $(n-2)$ hinged polygons. Immobilizing or robustly immobilizing the chain of shaded polygons immobilizes or robustly immobilizes the entire cycle. This leads to the next theorem.

Theorem 9.18 *A cycle of n hinged polygons without parallel edges can be immobilized with n point fingers, when $n \geq 3$ and $n \neq 5$; two and five polygons can be immobilized with three and six point fingers respectively. A cycle of n hinged polygons without parallel edges can be robustly immobilized with $\lceil \frac{6}{5}n \rceil$ point fingers; two and five polygons can be robustly immobilized with four and six fingers respectively.*

9.4.2 A chain of hinged polygons attached to a wall

Now consider the case when one vertex of a polygon at the end of a chain is attached to a wall as in Figure 9.10. Let $P_1, P_2, \cdots, P_n$ be the polygons from the one attached to the wall. This can be (robustly) immobilized in a similar way. Skip the two polygons P_1 and P_2, and (robustly) immobilize the rest. The number of the point fingers needed for immobility is $n-2+2=n$, when $n \geq 3$ and $n \neq 5$; six fingers suffice for five polygons. Likewise, the number for robust immobility for $n \geq 3$ polygons is $\lceil \frac{6}{5}(n-2+2) \rceil = \lceil \frac{6}{5}n \rceil$.

Two fingers are necessary to immobilize a single polygon with one vertex attached to a wall. Now we look at the case when $n=2$. Let v_1 be the hinge between P_1 and P_2. For two fingers to immobilize two polygons attached to a wall, the last polygon P_2 must satisfy the following condition: (i) if P_2 has two edges whose normals meet at the line $\overline{vv_1}$ and (ii) if the half-planes induced by these edges have counterclockwise rotational centers above $\overline{vv_1}$, and clockwise rotational centers below $\overline{vv_1}$, two point fingers suffice to immobilize the two polygons (Figure 9.11 (a)).

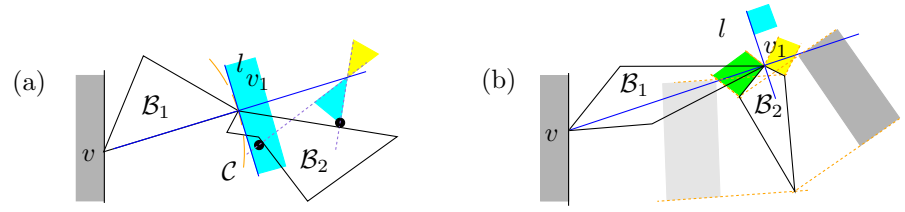


Figure 9.11: (a) These two polygons can be immobilized with two fingers. (b) These two polygons cannot be immobilized with two fingers.

Let $\mathcal{C}$ be the circle around v that v_1 follows, and let l be the half-plane bounded by the tangent line of $\mathcal{C}$ at v_1, which does not contain $\mathcal{C}$. If P_2 does not satisfy the conditions (i) and (ii) described above, the immediate rotation of P_2 around some point in the wedges will not confine the immediate motion of v_1 in the half-plane l.

Unfortunately, P_2 may not have such an edge pair that satisfy the condition (i) (Figure 9.11 (b)). Thus we need three point fingers to immobilize them. Surprisingly, three point fingers can also robustly immobilize them; use a similar method to that for three polygons in Section 9.3. This leads to the next lemma.

Lemma 9.19 *Two and three point fingers suffice to (robustly) immobilize a single polygon and two hinged polygons attached to a wall respectively.*

The next theorem summarizes the results so far.

Theorem 9.20 *When a chain of $n \geq 3, n \neq 5$ hinged polygons without parallel edges has a vertex at an end attached to a wall, n point fingers can immobilize the chain; two, three and six $(= n+1)$ point fingers can immobilize one, two and five such polygons respectively. Moreover, $\lceil \frac{6}{5}n \rceil$ point fingers can robustly immobilize such a serial chain.*

9.5 Conclusion

Note that the number of hinges h is $(n-1)$, and that the degree of freedom of the serial chain is $n+2 = h+3$. We believe that at least $(n+3)$ fingers—which is degree of freedom plus one—are necessary to robustly immobilize n hinged polygons without parallel edges. The reason comes from the equilibrium condition in [74], which requires that the origin be in the convex hull of the finger normals in the configuration space. The future research includes to verify whether this is indeed the lower bound. For the same reason, we believe that $(n+2)$ is the lower bound for immobility of a serial chain of hinged polygons without parallel edges.

Throughout the paper, we have assumed that the placement in which the serial chain has to be immobilized is given. The number of fingers required for immobilization is expected to be smaller when the placement can be chosen freely. In the future, we intend to study whether or not this is indeed the case. We also plan to work on immobilizing other types of hinges, more general structures of connected polygons other than serial chains or single cycles, and on the possibility of exploiting curvature for chain immobilization.

Chapter 10

Conclusion and Future Work

This thesis connects two areas: part manipulation and computational geometry. Here we reformulate fixturing problems as geometric intersection search problems. A form-closure grasp is formulated as a set of vectors that positively span a three-dimensional space (called wrench space), which is a geometrical problem. The fundamental question underlying most synthesis problems in this thesis (Chapters 3 to 5 and 7 to 8) is as follows. Given a number of sets in three- or six-dimensional space, determine tuples (of various dimensionalities) of sets so that there exists a convex hull defined by one point (sometimes two or three points) per set that contains the origin of the space. The shapes of these sets depends on the types of parts at hand; we showed that these shapes are line segments and/or algebraic arcs in three-dimensional wrench space for a planar object, and polyhedral shapes in six-dimensional wrench space for a three-dimensional object.

The formulation of form closure in wrench space (Theorem 2.2) has an advantage, namely it is easily extendible. It can be used to report form- or force-closure grasps of two or three dimensional parts with different shapes, as long as the intersections of the projected wrench sets are computable. The algorithms presented in this thesis efficiently reported all form-closure grasps on polygons, semi-algebraic sets and rectilinear polyhedra, as well as all force-closure grasps on polygons and semi-algebraic sets. This formulation can also be employed to tackle many variations of immobilization problems, such as efficiently computing all force-closure grasps of three-dimensional objects, or all form-closure grasps that is tolerant to minor shape variations or misplacements of the fingers, which will be discussed later.

The main contribution of this thesis is the conversion of fixturing problem into geometric problem with reduced dimensionality. The problem in three-dimensional wrench space is solved by searching for all red and blue intersections on planes. The problem in six-dimensional wrench space is divided into two closely related subproblems in three-dimensional space, each of which is again solved on planes.

Our approach is flexible in the sense that it is independent of the choice of solution method for the lower-level red-blue intersection problems. It merely breaks down a high-level immobilization problem into lower-dimensional red-blue intersection problems. Future improvements of current solutions to these problems may well result in improvements of our synthesis algorithms. This is why we believe that our approach could be used to attack many variations of immobilization problems, such as computing all form- or force-closure grasps of arbitrary planar objects, or computing all independent form- or force-closure grasp regions of planar and three-dimensional objects. Once we identify the characteristics of their projected wrench sets, and we find tools to search for red and blue intersections between these projected wrench sets, this method can easily be employed to solve these problems.

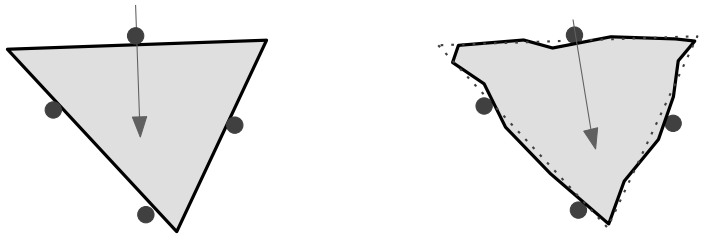


Figure 10.1: A form-closure grasp tolerant to a small shape variation.

In many cases, we do not need all grasps, but a few good ones. Among the reported grasps, we can always filter those of good quality for a given quality metric. If we can characterize the wrench sets of good grasps, we could incorporate the characteristics directly into the algorithm. This will make the solution simpler and more efficient. We believe that the insights gained in this thesis can be of good use to identify the characteristics of the wrench sets for good grasps for a given quality metric.

In practice, there are often errors in finger placements and also in object shapes. Hence grasp planning must take errors into account. If a grasp is sensitive to these errors, it will not hold the object in form closure. We will first discuss the grasps insensitive to finger misplacements.

As mentioned in Chapters 7 and 9, form closure is less sensitive to finger mispacements than for example second-order immobility (with fewer fingers). The insensitivity can, however, occasionally be very small, and it differs from finger to finger. We wish to report grasps insensitive to finger misplacements of specified size ε. In Chapter 7, we proposed output-sensitive algorithms to compute all such sets of prespecified regions for polygons. It is open to efficiently report all such sets for planar arbitrary objects and for three-dimensional objects. With the approach and insights gained in Chapter 8, it seems promising to first study the problem of computing all independent form-closure grasp regions of a rectilinear polyhedron.

There are many issues related to shape variations. One important problem is to efficiently synthesize grasps that are insensitive to minor shape variations. A form-closure grasp tolerant to shape variation is a form-closure grasp on an ideal part, such that it keeps a part with slightly different shape in form closure, when the fingers are moved to touch the new part in a specified way. Figure 10.1 illustrates such a grasp. Changes in the part shape result in changes of the normal line. When we give restrictions on how much the shape can change, we actually restrict how much the normal lines can change. This leads to a region in wrench space where the wrench points for these normal lines are. We believe that our approach can easily be used to tackle the problems related to this tolerance issue in an efficient way, if we can identify these regions in wrench space and on screen.

Extending our approach to immobilization problems in a modular setting (see Section 1.1.2) is relatively easier. When the object is in a given position, the set of finger positions will be a set of points in wrench space. We believe that our approach can efficiently compute all form- or force-closure grasps or a few good grasps or form-closure grasps tolerant to shape variations. When the object is allowed to change its position, computing all such grasps of an object remains open.

In Chapter 6, we proposed the first efficient output-sensitive algorithms to compute all second-order immobility grasps of polygons. It remains open to identify a necessary and sufficient condition for three fingers on an arbitrary planar object and for four fingers on a three-dimensional object to achieve second-order immobility. Ponce et al. [61] identified a necessary and sufficient condition for a three-dimensional object held with four fingers to be in equilibrium. Even for a polyhedron, to find a necessary and sufficient condition that takes into account relative curvatures

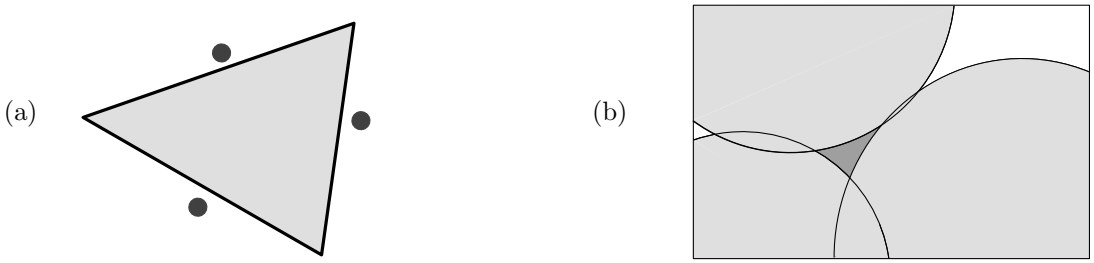


Figure 10.2: (a) A planar object caged by three fingers. (b) The dark gray region is the free region, where the caged object lies. Light gray regions are forbidden areas induced by the three fingers.

of the fingers and a polyhedron is challenging.

So far, we considered the problem of constructing immobilizing grasps of rigid parts. In practice, however, many parts are not rigid. When a finger presses against the part, it will deform. As a first step to tackle immobilization problem of deformable objects, we considered hinged polygons in Chapter 9. Among the problems that we considered in this thesis, this problem of Chapter 9 is the only one where we showed how to build one immobilizing grasp. To identify all immobilizing grasps of a serial chain of hinged polygons is definitely a challenging problem. To be able to find all immobilizing grasps of hinged polygons systematically, it would be useful to have a notion that is equivalent to wrench for rigid objects—a mathematical formulation to describe how a given force on one of the polygons affects the whole chain. To find immobilizing grasps of hinged polygons with a formulation of immobility based on this notion will be difficult, because of the high dimensionality, which is three plus the number of polygons in the chain.

Gopalakrishnan and Goldberg [41, 40] proposed a nice definition of form closure for deformable objects. They called this immobility *deform closure*. If we can formulate the problem of immobilizing a deformable object as the problem of searching for positively spanning vectors in some space, in combination with the proposed definition of deform closure, searching for all deform-closure grasps or choosing a few good grasps according to a given quality metric could be solved in an efficient way. The approach presented in this thesis then could be of use to attack the problem.

In this thesis, we focused on immobilizing grasps. But sometimes, it is enough to hold an object in a certain region, such that the object cannot escape the region. This is called *caging*. See Figure 10.2. Caging has many applications. For example, three independently moving robots can cage an object, and take it to a destination. The nature of caging problem is quite different from that of immobilization problem, since the caged object can move around in a certain region. Therefore analyzing instantaneous velocities alone is not enough to check whether an object is caged or not. A caged object in certain position has a corresponding point in configuration space. Thus the object with all possible positions and rotations in the caging fingers is mapped to a set of points in configuration space. This set is bounded and isolated from other points in free region. See Figure 10.2 (b). There are many attractive open problems related to caging, such as to find all finger positions for caging or to find good caging grasps.

Bibliography

[1] P. K. Agarwal. Partitioning arrangements of lines II: Applications. *Discrete & Computational Geometry*, 5:533–573, 1990.

[2] P. K. Agarwal and J. Matoušek. On range searching with semialgebraic sets. *Discrete & Computational Geometry*, 11:393–418, 1994.

[3] Pankaj K. Agarwal and Jeff Erickson. Geometric range searching and its relatives. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, volume 223 of *Contemporary Mathematics*, pages 1–56. American Mathematical Society, Providence, RI, 1999.

[4] S. Akella and M. T. Mason. Orienting toleranced polygonal parts. *International Journal of Robotics Research*, 19(12):1147–1170, 2000.

[5] E. Anshelevich, S. Owens, F. Lamiraux, and L. Kavraki. Deformable volumes in path planning applications. In *Proceedings of The IEEE International Conference on Robotics and Automation (ICRA)*, pages 2290–2295, San Fransisco, CA, April 2000.

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

[7] J. Basch, L. J. Guibas, and G. Ramkumar. Sweeping lines and line segments with a heap. In *Annual ACM Symposium on Computational Geometry*, pages 469–471, 1997.

[8] J. Bausch and K. Youcef-Toumi. Kinematic methods for automated fixture reconfiguration planning. In *International Conference on Robotics and Automation*, pages 1396–1401, May, 1990.

[9] M. de Berg, M. van Kreveld, M. H. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.

[10] R-P. Berretty. *Geometric design of part feeders*. PhD thesis, Institute of Information and Computing Sciences, Utrecht University, 2000.

[11] A. Blake and M. Taylor. Planning planar grasps of smooth contours. In *IEEE International Conference on Robotics and Automation (ICRA)*, page 834, 1993.

[12] G. M. Bone and Y. Du. Multi-metric comparison of optimal 2d grasp planning algorithms. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3061–3066, May 2001.

[13] Prosenjit K. Bose, David Bremner, and Godfried T. Toussaint. All convex polyhedra can be clamped with parallel jaw grippers. *Computational Geometry: Theory and Applications*, 6(5):291–302, 1996.

[14] R. Brost and K. Goldberg. A complete algorithm for designing planar fixtures using modular components. In *IEEE Transactions on Robotics and Automation*, volume 12, pages 31–46, 1996.

[15] R. C. Brost and K. Y. Goldberg. A complete algorithm for synthesizing modular fixtures for polygonal parts. *IEEE International Conference on Robotics and Automation (ICRA)*, 12:535–542, 1994.

[16] R. C. Brost and R. Peters. Automatic design of 3-d fixtures and assembly pallets. *International Journal of Robotics Research*, 17(12):1243–1281, 1998.

[17] F. Cazals and J.-C. Latombe. Effect of tolerancing on the relative positions of parts in an assembly. In *IEEE International Conference on Robotics and Automation (ICRA)*, 1997.

[18] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete & Computational Geometry*, 9(1):145–158, 1993.

[19] B. Chazelle, L. J. Guibas, and D. T. Lee. The power of geometric duality. *BIT*, 25(1):76 – 90, 1985.

[20] I.-M. Chen and J.W. Burdick. Finding antipodal point grasps on irregularly shaped objects. *IEEE Transactions on Robotics and Automation*, 9(4):507–512, 1993.

[21] J. Chen, K. Y. Goldberg, M. H. Overmars, D. Halperin, K-F. Böhringer, and Y. Zhuang. Shape tolerance in feeding and fixturing. In *Robotics, the algorithmic perspective*, pages 297–311. A.K. Peters, 1998.

[22] J.-S. Cheong, K. Y. Goldberg, M. H. Overmars, and A. F. van der Stappen. Fixturing hinged polygons. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 876–881, Washington DC, 2002.

[23] J.-S. Cheong, H.J. Haverkort, and A. F. van der Stappen. On computing all immobilizing grasps of a simple polygon with few contacts. In *ISAAC: 14th Internat. Sympos. Algorithms Computation*, pages 260–269, 2003.

[24] J.-S. Cheong, H.J. Haverkort, and A. F. van der Stappen. On computing all immobilizing grasps of a simple polygon with few contacts. *Algorithmica*, 44:117–136, 2006.

[25] J.-S. Cheong and A. F. van der Stappen. Output-sensitive computation of all form-closure grasps of a part bounded by algebraic arcs. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 784–790, Barcelona, 2005.

[26] J.-S. Cheong, A. F. van der Stappen, K. Y. Goldberg, M. H. Overmars, and E. Rimon. Immobilizing hinged parts. International Journal on Computational Geometry and Applications, to appear.

[27] Y-C. Chou, V. Chandry, and M. M. Barash. A mathematical approach to automatic configuration of machining fixtures: Analysis and synthesis. *Journal of Engineering for Industry, Transactions of the ASME*, 111:199–306, 1990.

[28] J. Cornellá and R. Suárez. On 2d 4-finger frictionless optimal grasps. In *International Workshop On Intelligent Robots and Systems (IROS)*, pages 3680–3685, 2003.

[29] J. Cornellá and R. Suárez. Determining independent grasp regions on 2d discrete objects. In *International Workshop On Intelligent Robots and Systems (IROS)*, pages 2936–2941, 2005.

[30] J. Cornellá and R. Suárez. Fast and flexible determination of force-closure independent regions to grasp polygonal objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 778–783, 2005.

[31] J. Cornellá and R. Suárez. On computing form-closure grasps/fixtures for non-polygonal objects. In *IEEE International Symposium on Assembly and Task Planning*, pages 138–143, 2005.

[32] J. Czyzowicz, I. Stojmenovic, and J. Urrutia. Immobilizing a polytope. In *Algorithms and Data structures – Proc. 2nd Workshop*, volume 519 of *Lecture Notes in Computer Science*, pages 214–227, 1991.

[33] J. Czyzowicz, I. Stojmenovic, and J. Urrutia. Immobilizing a shape. *International Journal of Computational Geometry and Applications*, 9(2):181–206, 1999.

[34] C. Davis. Theory of positive linear dependence. *American Journal of Mathematics*, pages 733–746, 1954.

[35] Dan Ding, Yun-Hui Liu, Yan-Tao Shen, and Guo-Liang Xiang. An efficient algorithm for computing a 3d form-closure grasp. In *IEEE International Conference on Robotics and Automation (ICRA)*, page 834, 2000.

[36] Dan Ding, Yun-Hui Liu, Michael Yu Wang, and S. Wang. Automatic selection of fixturing surfaces and fixturing points for polyhedral workpieces. *IEEE Transactions on Robotics and Automation*, 17(6):833–841, 2001.

[37] Dan Ding, Guoliang Xiang, Yun-Hui Liu, and Michael Yu Wang. Fixture layout design for curved workpieces. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2906–2911, May 2002.

[38] A.J. Goldman and A.W. Tucker. Polyhedral convex cones. *Linear Inequalities and Related Systems*, pages 19–40, 1956.

[39] Gopal Gopalakrishnan and Ken Goldberg. Gripping parts at concave vertices. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1590–1596, May 2002.

[40] K. Gopal Gopalakrishnan and K. Y. Goldberg. Computing deform closure grasps. In *International Workshop on Algorithmic Foundations of Robotics*, pages 203–218, 2004.

[41] K. Gopal Gopalakrishnan and K. Y. Goldberg. D-space and deform closure: A framework for holding deformable parts. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 345–350, 2004.

[42] E. Hoffman. *Modular Fixturing*. Manufacturing Technology Press, Lake Geneva, Wisconsin, 1987.

[43] C. Holleman, L. E. Kavraki, and J. Warren. Planning paths for a flexible surface patch. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 21–26, Leuven, Belgium, 1998.

[44] Yan-Bin Jia. Curvature-based computation of antipodal grasps. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1571–1577, May 2002.

[45] V. Koltun. Segment intersection searching problems in general settings. *Discrete & Computational Geometry*, 30:25–44, 2003.

[46] K. Lakshminarayana. Mechanics of form closure. Technical report, ASME 78-DET-32 American Society of Mechanical Engineers, 1978. to appear in Algorithmica.

[47] F. Lamiraux and L. E. Kavraki. Planning paths for elastic objects under manipulation constraints. *International Journal of Robotics Research*, 20(3):188–208, 2001.

[48] R. M. Lewis and V. Torczon. Rank ordering and positive bases in pattern search algorithms. Technical report, crpc-tr96674, Center for Research on Parallel Computation, Rice U niversity, Nov 1996.

[49] J.-W. Li, M.-H. Jin, and H. Liu. A new algorithm for three-finger force-closure grasp of polygonal objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1800–1804, 2003.

[50] J.-W. Li, M.-H. Jin, and H. Liu. On computing three-finger force-closure grasps of 2-d and 3-d objects. *IEEE Transactions on Robotics and Automation*, 19(1):155–161, 2003.

[51] X. Markenscoff, L. Ni, and C. H. Papadimitriou. The geometry of grasping. *International Journal of Robotics Research*, 9(1):61–74, 1990.

[52] J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete Comput. Geom.*, 10(2):157–182, 1993.

[53] Walter Meyer. Seven fingers allow force-torque closure grasps on any convex polyhedron. *Algorithmica*, 9(3):278–292, 1993.

[54] B. Mirtich and J. Canny. Easily computable optimum grasps in 2-D and 3-D. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 739–747. A.K. Peters, 1994.

[55] B. Mishra. Workholding-analysis and planning. In *International Workshop On Intelligent Robots and Systems (IROS)*, pages 53–57, 1991.

[56] B. Mishra, J. T. Schwartz, and M. Sharir. On the existence and synthesis of multifinger positive grips. *Algorithmica*, 2:541–558, 1987.

[57] G. Moroni and A. A. G. Requicha. Tolerance modeling and application programming interfaces. In *Proc. Symp. Tools and Methods for Concurrent Engineering*, pages 28–38, May 1996.

[58] V-D. Nguyen. Constructing force-closure grasps. *International Journal of Robotics Research*, 7(3):3–16, 1988.

[59] M. H. Overmars, A. Rao, O. Schwarzkopf, and C. Wentink. Immobilizing polygons against a wall. In *Annual ACM Symposium on Computational Geometry*, pages 29–38, 1995.

[60] J. Ponce and B. Faverjon. On computing three-finger force-closure grasps of polygonal objects. *IEEE Transactions on Robotics and Automation*, 11(6):868–881, 1995.

[61] J. Ponce, S. Sullivan, A. Sudsang, J-D. Boissonnat, and J-P. Merlet. On computing four-finger equilibrium and force-closure grasps of polyhedral objects. *International Journal of Robotics Research*, 16(1):13–35, 1996.

[62] Jean Ponce, Joel Burdick, and Elon Rimon. Computing the immobilizing three-finger grasps of planar objects. In *Proc. of the 1995 Workshop on Computational Kinematics*, pages 281–300, 1995.

[63] Jean Ponce, Darrell Stam, and Bernard Faverjon. On computing force-closure grasps of curved two-dimensional objects. *International Journal of Robotics Research*, 12(3):263–273, 1993.

[64] R. Prado and R. Suárez. Heuristic approach to construct 3-finger force-closure grasps for polyhedral objects. In *the 7th IFAC Symposium on Robot Control, SYROCO'2003*, pages 387–392, 2003.

[65] R. Prado and R. Suárez. Heuristic grasp planning with three frictional contacts on two or three faces of a polyhedron. In *IEEE International Symposium on Assembly and Task Planning*, pages 112–118, 2005.

[66] A. Rao and K. Y. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10(2):201–225, 1993.

[67] A. Rao and K. Y. Goldberg. Friction and part curvature in parallel-jaw grasping. *Journal of Robotic Systems*, 12(6):365–382, 1995.

[68] A. Rao and K. Y. Goldberg. Manipulating algebraic parts in the plane. *IEEE Transactions on Robotics and Automation*, 11(4):598–602, 1995.

[69] A.A.G. Requicha. Toward a theory of geometric tolerancing. *International Journal of Robotics Research*, 2(4):45–60, 1983.

[70] F. Reuleaux. *The Kinematics of Machinery*. Macmilly and Company, 1876. Republished by Dover in 1963.

[71] E. Rimon. New bounds on the number of frictionless fingers required to immobilize three dimensional objects. Technical report, UU-CS-1996-49, Department of Mechanical Engineering, Technion, Israel Institute of Technology, 1999.

[72] E. Rimon. A curvature-based bound on the number of frictionless fingers required to immobilize three-dimensional objects. *IEEE Transactions on Robotics and Automation*, 17(5):679–697, 2001.

[73] E. Rimon and J. W. Burdick. New bounds on the number of frictionless fingers required to immobilize planar objects. *J. of Robotic Systems*, 12(6):433–451, 1995.

[74] E. Rimon and J. W. Burdick. Mobility of bodies in contact—part I: A second-order mobility index for multiple-finger graps. *IEEE Transactions on Robotics and Automation*, 14:696–708, 1998.

[75] E. Rimon and J. W. Burdick. Mobility of bodies in contact—part II: How forces are generated by curvature effects. *IEEE Transactions on Robotics and Automation*, 14:709–717, 1998.

[76] E. Rimon and J. W. Burdick. New bounds on the number of frictionless fingers required to immobilize an object. Technical report, rms-94-01, Department of Mechanical Engineering, California Institute of Technology, Sept 1994.

[77] B. Roth. Screws, motors, and wrenches that cannot be bought in a hardware store. In *Int. Symp. on Robotics Research*, pages 679–693, 1984.

[78] U. Roy, C.R. Liu, and T.C. Woo. Review of dimensioning and tolerancing: Representation and processing. *Computer-Aided Design*, 23(7):466–468, 1991.

[79] K. Salisbury. *Kinematic and force analysis of articulated hands*. PhD thesis, Stanford University, 1982.

[80] P. Somov. Über Gebiete von Schraubengeschwindigkeiten eines starren Körpers bei verschiedener Zahl von Stützflächen. *Zeitschrift Mathematik Physik*, 45:245–306, 1900.

[81] A. F. van der Stappen. On the existence of form-closure configurations on a grid. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1237–1242, 2000.

[82] A. F. van der Stappen, C. Wentink, and M. H. Overmars. Computing immobilizing grasps of polygonal parts. *International Journal of Robotics Research*, 19(5):467–479, 2000.

[83] A. Sudsang, J. Ponce, and N. Srinivasa. Algorithms for constructing immobilizing fixtures and grasps of three-dimensional objects. In J-P. Laumond and M. H. Overmars, editors, *Algorithms for Robotic Motion and Manipulation*, pages 363–380. A.K. Peters, 1997.

[84] H. Voelcker. A current perspective on tolerancing and metrology. *Manufacturing Review*, 6:258–268, 1993.

[85] R. Wagner, Y. Zhuang, and K. Y. Goldberg. Fixturing faceted parts with seven modular struts. In *IEEE International Symposium on Assembly and Task Planning*, pages 133–139, 1995.

[86] A. S. Wallack and J. Canny. Modular fixture design for generalized polyhedra. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 830–837, 1996.

[87] A. S. Wallack and J. Canny. Planning for modular and hybrid fixtures. *Algorithmica*, 19(1–2):40–60, 1997.

[88] M. Y. Wang and D. Pelinescu. Optimal fixture layout design in a discrete domain for 3d workpieces. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 792–798, May 2001.

[89] Michael Yu Wang. Characterizations of positioning accuracy of deterministic localization of fixtures. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2894–2899, May 2002.

[90] Y. Wang. An optimum design approach to 3d fixture synthesis in a point set domain. *IEEE Transactions on Robotics and Automation*, 16(6):839–846, 2000.

[91] C. Wentink. *Fixture Planning—Geometry and Algorithms*. PhD thesis, Department of Computer Science, Utrecht University, 1998.

[92] C. Wentink, A. F. van der Stappen, and M. H. Overmars. Algorithms for fixture design. In J-P. Laumond and M. H. Overmars, editors, *Algorithms for Robotic Motion and Manipulation*, pages 321–346. A.K. Peters, 1997.

[93] Chee K. Yap. Exact computational geometry and tolerancing metrology. In Snapshots of Computational and Discrete Geometry, Volume 3, Sept. 1995. McGill School of Computer Science Tech. Report No. SOCS-94.50.

[94] Y. Zhuang and K. Y. Goldberg. On the existence of solutions in modular fixturing. *International Journal of Robotics Research*, 15:646–656, 1996.

Samenvatting

Automatisering van fabricageprocessen is van wezenlijk belang voor de moderne industrie. In veel fabricageprocessen is het nodig om voorwerpen te *immobiliseren*, dat wil zeggen, ze zodanig vast te houden dat ze niet kunnen bewegen. Een machine of een monteur kan er dan onderdelen aan bevestigen of er andere bewerkingen op uitvoeren. Wanneer voorwerpen machinaal worden verplaatst, bijvoorbeeld op een lopende band of door een robot, dan is het vaak van belang dat ze niet kunnen draaien of verschuiven ten opzichte van de band of de robothand die ze verplaatst.

Het immmobiliseren van een voorwerp kan op veel verschillende manieren worden gemodelleerd, afhankelijk van bijvoorbeeld het soort robotvingers waarmee het voorwerp op zijn plaats wordt gehouden, met welke eigenschappen van voorwerp en vingers rekening wordt gehouden, en hoe nauwkeurig de vingers moeten worden geplaatst. Het basismodel is form-closure, in 1876 geformuleerd door Reuleaux [70]. Een stijf voorwerp is *in form-closure* als een aantal wrijvingsloze vingers aan de rand van het voorwerp elke eindige beweging en elke oneindig kleine beweging van het voorwerp uitsluiten. Andere modellen zijn *force-closure*, waarbij we gebruik maken van wrijvingsweerstand tussen de vingers en het voorwerp, en *second-order-immobility*, waarbij door rekening te houden met de kromming van het object met minder vingers kan worden volstaan, mits zeer nauwkeurig geplaatst.

Veel onderzoekers hebben reeds aan immobilisatievraagstukken gewerkt. Ze hebben antwoord gegeven op vragen zoals: hoeveel vingers zijn in het slechtste geval nodig en toereikend om een voorwerp in form-closure te houden? Hoe kunnen we een form-closure-greep (een adequate plaatsing van vingers) berekenen? Dit proefschrift richt zich op de tweede vraag. Tot op zekere hoogte beantwoordden Van der Stappen et al. [82] deze vraag al: zij sommen alle viertallen van kanten van een veelhoek op zodat er een form-closure-greep bestaat met een vinger aan elke kant van het viertal.

In dit proefschrift gaan we verder. We leggen uit hoe we voor veelhoeken en een bepaald soort gebogen voorwerpen in het vlak efficiënt *alle* tweetallen, drietallen en viertallen van concave hoekpunten en (eventueel gebogen) kanten kunnen berekenen, zodat we het voorwerp in form-closure of force-closure kunnen houden met een vinger aan elk hoekpunt en elke kant van het tweetal, drietal of viertal. We geven ook enkele resultaten voor second-order-immobility. Bovendien leggen we uit hoe we alle form-closure-grepen van een een as-parallel veelvlak[1] efficiënt kunnen berekenen. Daarmee wordt voor het eerst resultaat geboekt met betrekking tot het efficiënt opsommen van form-closure-grepen van drie-dimensionale voorwerpen.

De algoritmen in dit proefschrift zijn efficiënt in de zin dat ze weinig rekentijd besteden aan combinaties van hoekpunten en kanten die uiteindelijk geen form-closure-greep opleveren. De algoritmen in hoofdstuk 3 tot en met 7 zijn in feite *uitvoer-gevoelig:* de rekentijd hangt vooral af van het *feitelijke* aantal verschillende combinaties van hoekpunten en kanten die form-closure-grepen opleveren, en niet van het theoretische maximum-aantal combinaties.

[1] Een as-parallel veelvlak is een veelvlak waarvan alle ribben evenwijdig met de x-as, y-as of z-as zijn.

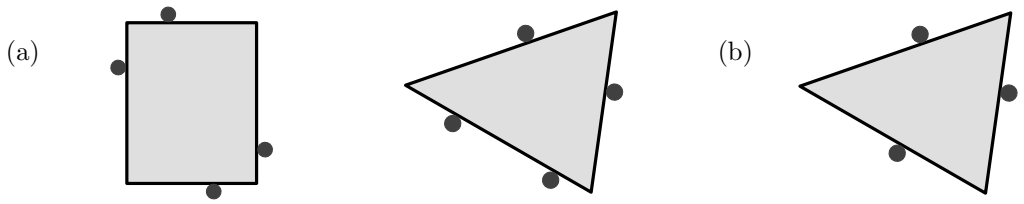


Figure 3: (a) Form-closure-grepen (b) Een second-order-immobility-greep

Hoewel Reuleauxs analyse van form-closure makkelijk te begrijpen is, blijkt het construeren van alle form-closure-grepen voor een voorwerp in het vlak eenvoudiger als we ons baseren op een andere, gelijkwaardige formulering van form-closure. Een aantal vingers houden een voorwerp in form-closure als zij alle mogelijke krachten en momenten die op het voorwerp werken kunnen opheffen. Om een form-closure-greep voor een voorwerp in het vlak te vinden, kunnen we dus zoeken naar een combinatie van vingers waarmee we de volledige drie-dimensionale ruimte van combinaties van krachten en momenten bestrijken. Ons probleem wordt dan een meetkundig probleem van de volgende vorm. We krijgen een verzameling eenvoudige vormen in een drie-dimensionale ruimte, die de mogelijke posities van vingers vertegenwoordigen. We willen nu alle combinaties van vormen vinden zodat elke combinatie vier punten bevat waarvan de convexe omhullende de oorsprong bevat. Dit meetkundige probleem in drie dimensies zetten we vervolgens weer om in een twee-dimensionaal probleem door de vormen op vlakken te projecteren, en naar de vormen te zoeken die elkaar in de projectie snijden. Daartoe maken we gebruik van methoden die bekend zijn uit de computationale geometrie.

De formulering van form-closure in de ruimte van krachten en momenten kan worden uitgebreid tot drie-dimensionale voorwerpen—de ruimte heeft dan zes dimensies.

In hoofdstuk 3 geven we efficiënte algoritmen om voor veelhoeken alle combinaties van concave hoekpunten en kanten op te sommen die minstens één form-closure-greep met minder dan vier wrijvingsloze puntvingers toelaten. Voor as-parallelle veelhoeken geven we algoritmen die efficiënter zijn dan de algoritmen voor willekeurige veelhoeken.

In hoofdstuk 4 doen we hetzelfde voor viervingerige form-closure-grepen voor een bepaald soort gebogen voorwerpen in het vlak, namelijk semi-algebraïsche verzamelingen.

In hoofdstuk 5 zoeken we naar grepen waarbij we gebruik maken van de wrijvingsweerstand tussen de vingers en het voorwerp. Onbeweeglijkheid als gevolg van het gebruik van vingers met wrijving wordt *force-closure* genoemd. We berekenen efficiënt alle twee- en drievingerige force-closure-grepen van veelhoeken en semi-algebraïsche verzamelingen.

In hoofdstuk 6 beschrijven we efficiënte, uitvoer-gevoelige algoritmen om alle twee- en drievingerige immobiliserende grepen voor een veelhoek op te sommen. Het is bekend dat in het algemeen voor de meeste voorwerpen in het vlak vier vingers nodig en voldoende zijn om ze in form-closure te houden [51, 56]. Veel voorwerpen in het vlak (vooral veelhoeken) kunnen echter toch met drie wrijvingsloze vingers op hun plaats worden gehouden dankzij second-order-immobility. Onze algoritmen zijn gebaseerd op de noodzakelijke en voldoende meetkundige voorwaarden voor second-order-immobility zoals beschreven door Czyzowicz et al. [33].

Hoofdstuk 7 gaat over grepen die kleine afwijkingen in de plaatsing van de vingers kunnen verdragen. We verdelen de kanten van de veelhoek die we op zijn plaats willen houden eerst in kleine segmenten van lengte ε. Vervolgens berekenen we alle combinaties van drie of vier concave hoekpunten en segmenten, zodat we de veelhoek in form-closure kunnen houden met vingers aan elk van deze hoekpunten of segmenten—ongeacht waar de vingers de segmenten precies raken. Op deze manier berekenen we form-closure-grepen die ongevoelig zijn voor afwijkingen van max-

imaal $\frac{1}{2}\varepsilon$ in de plaatsing van de vingers aan de kanten. Voor as-parallelle veelhoeken geven we weer een efficiënter algoritme.

In hoofdstuk 8 maken we de stap naar drie-dimensionale voorwerpen: we berekenen alle combinaties van zijvlakken, concave ribben en concave hoekpunten van een as-parallel veelvlak, zodat vier tot zeven wrijvingsloze puntvingers op elk van deze combinaties minstens één form-closure-greep toelaten. De algoritmen filteren eerst alle combinaties van hoekpunten, ribben en zijvlakken om kandidaten voor form-closure-grepen te selecteren. Vervolgens worden deze kandidaten gecontroleerd om de uiteindelijke grepen te bepalen. Bij vrijwel alle voorgestelde algoritmen hangt de rekentijd vooral van de uitvoergrootte af; sommige algoritmen moeten echter alle kandidaatgrepen stuk voor stuk controleren.

In hoofdstuk 9 richten we ons op het immobiliseren van een ingewikkelder voorwerp: een keten van scharnierend aan elkaar bevestigde veelhoeken. Zulke ketens zijn moeilijker op hun plaats te houden dan stijve voorwerpen omdat een keten meer vrijheidsgraden heeft, evenredig met het aantal veelhoeken waaruit de keten bestaat. We bestuderen hoe zo'n keten in een bepaalde stand kan worden gehouden door middel van wrijvingsloze puntvingers. We geven en analyseren voorwaarden voor de onbeweeglijkheid en voor de *robuuste onbeweeglijkheid* (robust immobility) van zulke ketens, vergelijkbaar met de voorwaarden voor second-order-immobility en form-closure van stijve voorwerpen. Voor beide gevallen laten we zien hoeveel vingers in het slechtste geval nodig zijn om een gegeven keten van scharnierende veelhoeken te immobiliseren, en hoe we een greep kunnen vinden die voldoet.

Tot slot bespreken we in hoofdstuk 10 enkele interessante vraagstukken voor verder onderzoek.

Acknowledgement

I am a lucky person. During the stay in the Netherlands, I met not only great people, but also a different culture, which improved my attitude towards life. Moreover, my beloved son Junha was born, and my husband Otfried and my copromotor Frank were promoted. Without the help and support of the people around me, this thesis would not have been completed.

First of all, my promotors Mark Overmars and Frank van der Stappen deserve big thanks. They were very supportive, patient and offered me good guidance and brilliant inspirations. They taught me many things not only in words but also in actions. A few of those that I learned from them are how they handled and arranged their duties and works, how they did research with other people, and how they find interesting research problems. Thank you very much!

I also would like to thank Jean-Daniel Boissonnat, Ken Goldberg, Frans Groen, Rolf Klein and Jan van Leeuwen for being in the reading committee for my thesis and for reading my thesis carefully . Ken Goldberg offered me an intriguing problem about hinged polygons, which lead to publishing a paper together. I appreciate his great inspirations. He also invited me to Berkeley for research, which was a great experience.

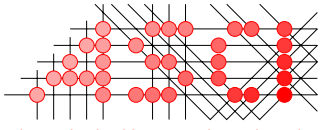
I would like to thank other coauthors, Elon Rimon and Herman Haverkort for their cooperation. Elon Rimon's passion toward research has motivated me and gave me courage for challenging problems. Herman Haverkort offered good cooperation with me. He also kindly helped me with the "samenvatting", title page, and paperwork. He also invited me to Karlsruhe for research. It was nice to see other people doing computational geometry. Alexander Wolff (Sascha) kindly offered me his place to stay in Karlsruhe—I appreciate it, Sascha! I also would like to express thanks to Frans Oort, Pankaj K. Agarwal, Marc van Kreveld, Xavier Goaoc, Hyeon-Suk Na, Chee K. Yap, Mark de Berg and Yan-Bin Jia for their helpful discussions about algebraic curves and data structures. I hope that I can get such good people around me in the future as well.

My colleagues at the institute, my friends and my family also deserve many thanks. Especially, I would like to express my respect, love and thank to my husband Otfried Cheong for his endless support and love. He was the one who showed me that research could be fun. He and his students at KAIST offered me space and warm support so that I could finish my thesis in 2005. I especially thank Hyo-Sil Kim, Jang-Hwan Kim, Chang-Bum Choi and Chang-Yul Choi for proof-reading some chapters and discussions. I also would like to express special thanks to Arno Kamphuis, Mirela Tanase, Iris Reinbacher and Jur van den Berg for helping me with formatting the thesis, proof-reading and paperwork. Mirela and Iris, thank you for your support, and for agreeing to be my paranymphs. There are a lot of people, who indirectly contributed to this thesis filling my life with happiness and pleasure. My parents, my three dear sisters, and my friends Bertha, Madalina, Karina, Twan, René, Geert-Jan, Remco, Roland, Frank ter Haar, Joachim, Dennis, Monique, Lydia, Sandra, Corine, Wilke, Rita, Floor, Esther, Eveline, Ate, Koos, Sim, Henk and Henk, Thomas, Hyojeong, Seol-Ah, Jae-Seok, Mira, Jung-Gun, Sang-Won, Linda, Nokyoung and Manfred, I thank you all. Those friends and family who are not listed here, please forgive me, but the space is limited. Dank je wel, allemaal! Thank you all! Modu Komawayo!

Curriculum Vitae

Jaesook Cheong

22. 03. 1971	born in Seoul, South Korea
1991 - 1996	Computer Science and Engineering at Pohang University of Science and Technology, South Korea
1997 - 2000	Computer Science at Hong Kong University of Science and Technology, Hong Kong
2000 - 2005	Information and Computing Sciences at Utrecht University, The Netherlands
July 2002	guest at the department of Industrial Engineering and Operations Research at Berkeley University, USA
Aug 2004	guest at the department of Computer Science at Karlsruhe University, Germany



This work was carried out in the ASCI graduate school.
ASCI dissertation series number 136.

ISBN-10: 90-393-4380-2
ISBN-13: 978-90-393-4380-7