CHAPTER IV

COMPUTATION OF ALL 2D FRICTIONLESS FORCE CLOSURE GRASPS

4.1 Introduction

In this chapter, we consider the problem of computing all force closure grasps of four frictionless contacts. Given a set of n frictionless contact points together with corresponding inward normals, the task is to report every quadruple of four contact points that achieves force closure. Be noted that the number of output is $O(n^4)$. The contribution of this chapter is a method to circumvent this limitation. We introduce an output sensitive algorithm that runs in $O(n^3 \lg^2 n + K)$ where K is the number of output. This section describes the detail of the algorithm along with its analysis.

A naive approach is to enumerate every possible quadruple of wrenches and then apply Proposition 2.9 or Proposition 2.11 to test each quadruple. This approach involves four nested loops which enumerate each member of a quadruple thus yield $O(n^4)$ complexity. To achieve better time complexity, this four level enumeration structure has to be modified. Instead of enumerating all four members, our method enumerates only two members of a quadruple, i.e., every pair of wrenches is enumerated. For each pair, the method identifies all pairs of the remaining wrenches that satisfy Proposition 2.11. This identification of the satisfying pair is done in an output sensitive manner and thus the overall complexity is reduced. Let us denote a routine which takes as an input a pair of wrench $\boldsymbol{w}_a, \boldsymbol{w}_b$ and a set of wrench W and reports every pair in W that satisfies Proposition 2.11 with respect to $\text{SPAN}^+(\{\boldsymbol{w}_a, \boldsymbol{w}_b\})$ by $\text{FINDINTS}(\boldsymbol{w}_a, \boldsymbol{w}_b, W)$. The pseudocode of our algorithm is listed as follows.

Algorithm 1 Algorithm for Computing All Frictionless Closure Grasp

Require: $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n$: n contact wrenches
Ensure: R: all quadruples of four wrenches that positively span the space
1: $R \Leftarrow \emptyset$
2: **for** $i = 1$ to n **do**
3: **for** $j = i + 1$ to n **do**
4: $W \Leftarrow \boldsymbol{w}_{j+1}, \ldots, \boldsymbol{w}_n$
5: $R \Leftarrow R \cup \text{FINDINTS}(\boldsymbol{w}_i, \boldsymbol{w}_j, W)$
6: **end for**
7: **end for**

The rest of this chapter is dedicated to the explanation of $\text{FINDINTS}(\boldsymbol{w}_a, \boldsymbol{w}_b, W)$. Section

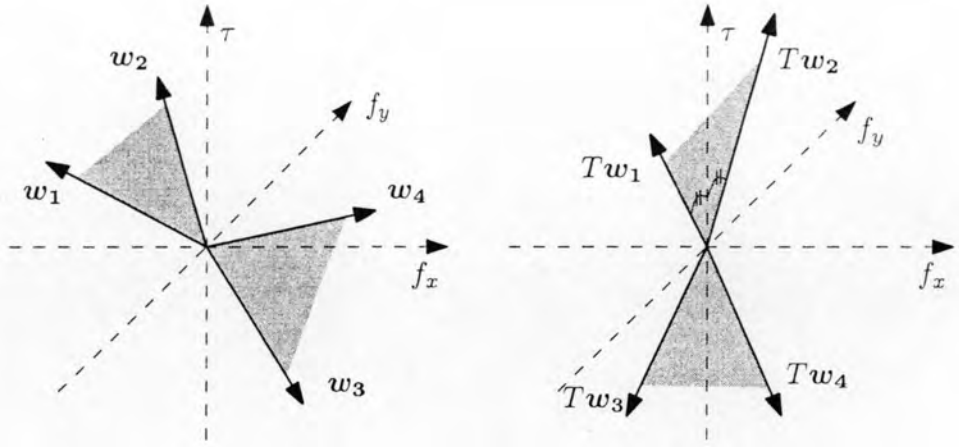

Figure 4.1: Four wrenches before and after rotated by T. (a) four wrenches forming two fans. (b) The wrenches rotated by T.

4.2 describes a condition of force closure that is utilized by FINDINTS$(\boldsymbol{w}_a, \boldsymbol{w}_b, W)$. Section 4.3 describes the algorithm and its complexity analysis. Numerical example and empirical comparison are given in Section 4.4. Finally, Section 4.5 summarizes this chapter.

4.2 Force Closure Condition Using Force Dual Representation

The next lemma provides a condition for four wrenches to positively span $\mathbb{R}^3$, utilizing the force dual representation given in Section 2.6. Intuitively speaking, the condition groups the wrenches into two groups, forming two fans. All wrenches are then rotated to align one fan exactly on $x = 0$ plane where τ axis divides the fan equally (see Figure 4.1). The condition states that the wrenches positively span the space when the relative interior of the slices of the two fans intersect as a point.

Lemma 4.1 *Let the given four wrenches be denoted by* $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_4$. *Let* T *be a rotational transformation that aligns* $\boldsymbol{w}_1$ *and* $\boldsymbol{w}_2$ *on* $x = 0$ *plane and the positive* z *axis divides the fan* $F_a = \text{SPAN}^+(\{\boldsymbol{w}_1, \boldsymbol{w}_2\})$ *equally. Let* F_b *be the negative fan of the other two wrenches, i.e.,* $F_b = \text{SPAN}^-(\{\boldsymbol{w}_3, \boldsymbol{w}_4\})$. *These wrenches positively span* $\mathbb{R}^3$ *if and only if* $\text{RI}(\Omega(TF_a))$ *and* $\text{RI}(\Omega(TF_b))$ *intersect as a point on* Π.

Proof: It is obvious that $\text{RI}(\Omega(\text{SPAN}^+(T\boldsymbol{w}_1, T\boldsymbol{w}_2)))$ is a segment. Hence, the intersection between $\text{RI}(F_a)$ and $\text{RI}(F_b)$, if exists, will entirely intersect Π. The intersection between the two slices is a point only when $\text{RI}(F_a) \cap \text{RI}(F_b)$. By Proposition 2.11, the proof is complete. ■

FINDINTS routine relies on Lemma 4.1 directly. It takes $\boldsymbol{w}_a$ and $\boldsymbol{w}_b$ in the form of the fan F_a in the lemma and compute the corresponding rotational transformation T. The transformation is applied to all wrenches. Then, every pair of wrenches that satisfies Lemma 4.1 with respect to $F_a = \text{SPAN}^+(\{Tvw_a, T\boldsymbol{w}_b\})$ is identified by the routine.

4.3 FINDINTS Routine

Given two wrenches $\boldsymbol{w}_a, \boldsymbol{w}_b$ and a set of wrenches $W = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_m\}$, FINDINTS computes all pairs of wrenches in W such that each of three pairs, together with $\boldsymbol{w}_a$ and $\boldsymbol{w}_b$, positively span the space. This routine has the worst case running time in $O(n \lg^2 n + K)$ where n is the number of wrenches in W. This routine also serves as a required core module for the computation of all frictional force closure grasps to be discussed in Chapter 5.

Let us refer to the wrench fan of $\boldsymbol{w}_a$ and $\boldsymbol{w}_b$ as the *anchor fan*. The routine starts by computing the rotation matrix T capable of rotating the anchor fan as described in Lemma 4.1. The transformation matrix can be computed from the following equation.

$$T = \begin{pmatrix} T_x & = & \frac{\boldsymbol{w}_a \times \boldsymbol{w}_b}{\|(\boldsymbol{w}_a \times \boldsymbol{w}_b)\|} \\ T_y & = & T_z \times T_x \\ T_z & = & \frac{\hat{\boldsymbol{w}}_a + \hat{\boldsymbol{w}}_b}{\|\hat{\boldsymbol{w}}_a + \hat{\boldsymbol{w}}_b\|} \end{pmatrix} \tag{4.1}$$

All wrenches in W, as well as $\boldsymbol{w}_a$ and $\boldsymbol{w}_b$, are rotated by T and their dual representation is computed. We let $V = TW$ be the set of rotated wrenches.

The rotated wrenches are then represented by their intersection with Π. Obviously, $\Omega(TF_a)$ is a vertical segment on Π lying parallel to f_y axis. The routine identifies every pair of wrenches whose negative slice intersects the segment $\Omega(TF_a)$. This approach transforms the problem of 3D fan intersection into 2D segment intersections.

Relative interiors of two segments intersect as a point only when the endpoints of one segment lies on different sides of the line through the other segment, and vice versa. Since $\Omega(TF_a)$ is a segment on f'_y axis, the endpoints of the intersecting segment clearly have to be on different sides of f'_y axis. Hence, we can neglect any wrench that lies on f'_y axis. We partition the members of V, according to the sign of f'_x coordinates, into two subsets: the left subset, denoted by V_L, and the right subset, denoted by V_R. The left subset (resp. the right subset) contains every dual point

having negative (resp. positive) f'_x coordinate. Each subset is further divided into three smaller subsets according to the flag of its member. We denote by V_L^+, V_L^0, V_L^-, the set of points lying on the left of f'_y axis that is flagged as a positive, zero and negative dual point, respectively. The sets V_R^+, V_R^0, V_R^- are the respective counterparts for the right subset. To identify all intersecting segments, we consider all possible pairing cases of each kind of dual points, namely, negative-negative, negative-positive, negative-zero, positive-positive, positive-zero, and finally, zero-zero. For each kind of pairing, we derive a condition such that the pair satisfies Lemma 4.1.

4.3.1 Negative-Negative Dual Points Pairing

Let us start with the case of two negative dual points. The segment joining them represents a slice of a negative fan. Consider an arbitrary negative dual point p lying on the left of $\Omega(TF_a)$, i.e., $p \in V_L^-$. We can identify the area of another negative dual point where the segment joining them intersects $\Omega(TF_a)$. The area is bounded by the f'_y axis and the line joining p and the endpoints of $\Omega(TF_a)$. Figure 4.2 illustrates the area. The condition describing the area can be easily described by the angle of a vector from endpoints of $\Omega(TF_a)$ to the point and the f'_y axis. Let us define the alpha angle of a point q, denoted by α_q, to be the angle between the negative f'_y axis and the vector from $\Omega(\boldsymbol{w}'_a)$ to q. Similarly, let the beta angle of a point q, denoted by β_q, be the angle between the positive f'_y axis and the vector from $\Omega(\boldsymbol{w}'_b)$ to q (see Figure 4.2). The condition for a point q, lying on the right of f'_y axis, to be in the area can be written as

$$0 < \alpha_q < \pi - \alpha_p \tag{4.2}$$

$$0 < \beta_q < \pi - \beta_p \tag{4.3}$$

Equation (4.2) and (4.3) represent a rectilinear region in α-β space. The problem is transformed into an *Orthogonal Range Search* problem (Goodman and O'Rourke, 1997). Points in the reduced space are plotted in α-β space and stored in a two dimensional range tree (Willard, 1985). Given n points, the tree can be constructed in $O(n \lg^2 n)$. A query for all point lying in a rectilinear region takes $O(\lg^2 n + K)$.

For all points in V_R^-, we plot them in $\alpha\beta$ space and put them into a 2D range tree $\mathcal{P}_L^-$. Since there are $O(n)$ points in V_R^-, this step requires $O(n \lg^2 n)$. Next, for each point $p \in V_L^-$, we calculate α_p and β_p and query for every point in V_R^- that satisfies Equation (4.2) and (4.3). This

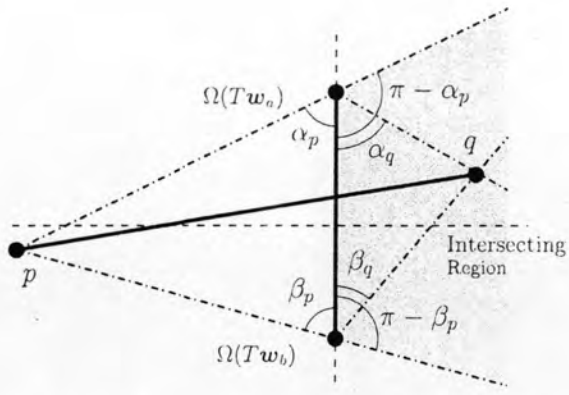

Figure 4.2: The region that satisfies Equation (4.2) and (4.3).

step takes $O(n \lg^2 n + K)$. The result are all pairs of two negative dual points that the segment joining them intersect $\Omega(TF_a)$.

4.3.2 Negative-Zero Dual Points Pairing

One negative dual point and one zero dual point also form a negative slice as a negative ray. A zero point is a point lying at the infinity at some direction. The ray starts from the negative dual point and going in the direction of the zero dual point. Since the zero dual point lies farthest amongst all points in the same direction, the vector from any point to the zero dual point is the same and is the vector describing the direction of the zero point. Hence, the alpha and beta angles of the zero dual point are equal. Figure 4.3 shows various points lying on the same direction, including a zero dual point and vectors from the anchor wrench to them. Equation (4.2) and (4.3) still holds for the case of zero dual point. However, since the alpha and beta angles are equal, we can simply sort the zero points by their angles and use binary search in lieu of 2D range search. This would reduce the complexity to $O(n \lg n + K)$.

4.3.3 Negative-Positive Dual Points Pairing

The other kind of a negative slice is a ray formed by a one negative dual point and one positive dual point. As shown in Figure 2.8, a negative slice starts from the negative dual point, going toward the opposite direction of the direction from the positive dual point to the negative dual point. It is trivial in this case to verify that both the positive dual point and the negative dual point must be on the same side of $\Omega(TF_a)$ to have their rays intersect $\Omega(TF_a)$. Given a fixed

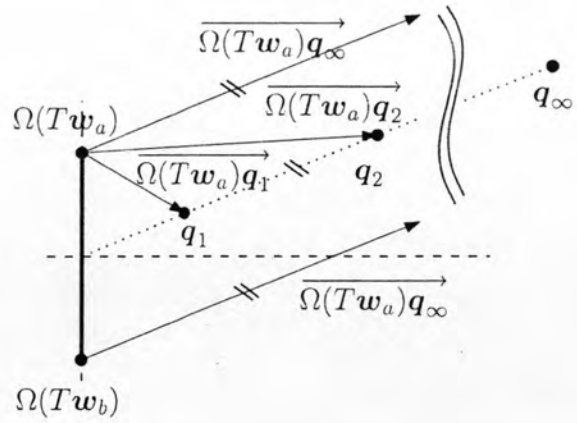

Figure 4.3: Various dual points lying on the same direction. The direction of vectors from the anchor points to the points at infinity are the same, regardless on the position of the anchor points.

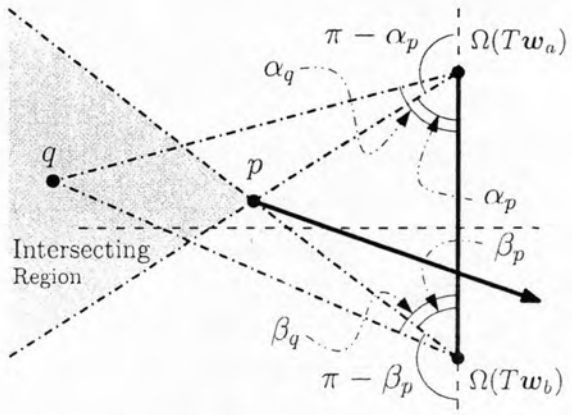

Figure 4.4: The region that satisfies Equation (4.4) and (4.5).

negative dual point p, the area in which a positive dual point q can form a ray intersecting $\Omega(TF_a)$ is bounded by the line joining the endpoints of $\Omega(TF_a)$ and the negative dual point p. Figure 4.4 illustrated the region. The constraint describing the region can be written, in the form of α-β space, as follows.

$$\alpha_p < \alpha_q < \pi \tag{4.4}$$

$$\beta_p < \beta_q < \pi \tag{4.5}$$

Similarly to the case of two negative points, we can efficiently identify all points satisfying Equation (4.4) and (4.5) by using a 2D range tree. For all left positive points in V_L^+, we plot them in α-β space and put them into a 2D range three $\mathcal{P}_L^+$. Next, for each left negative dual point p in V_L^-, we calculate α_p and β_p and then query for every point in $\mathcal{P}_L^+$ that satisfies Equation (4.4) and

(4.5). The same process is also applied to the right positive dual points and the right negative dual points in V_R^+ and V_R^-. The time complexity in this case is also $O(n \lg^2 n + K)$ for both left and right cases.

4.3.4 Remaining Pairing

The remaining cases, which are positive-zero, positive-positive and zero-zero pairing, do not represent a negative slice. These cases can be neglected.

4.3.5 Complexity Analysis

The FINDINTS algorithm described in Section 4.3 computes all pairs of wrenches from the construction of range trees and a sorted list. There are three range trees to be constructed: trees containing points in V_L^-, V_L^+ and V_R^+, respectively. Two lists are constructed: a list containing points in V_L^0 and V_R^0. The construction takes $O(n \lg^2 n + K)$. There is at least one query for each negative point, each takes $O(n \lg n + K)$ and there are $O(n)$ negative dual points. Hence, FINDINTS takes $O(n \lg^2 n + K)$. FINDINTS is applied to every pair of wrenches. There are $n(n-1)/2 \in O(n^2)$ pairs of anchor wrenches to be checked, the entire process can be done in $O(n^3 \lg^2 n + K)$.

4.4 Numerical Example and Comparison

The methods presented in Section 4.3 are theoretically shown to be efficient to compute all force closure grasps in terms of time complexity. Nevertheless, to be fair it is necessary to consider an empirical example of the method. In this section, we provide numerical comparison between the presented method with selected reference methods. To the best of our knowledge, we are the first to introduce methods to compute all grasps from discrete contact points. This implies the lack of other methods that solve exactly the same problem. Alternately, we consider using single query force closure tests which receive much interest in the recent years. The single query test is embedded within a loop that enumerate every combination of three contact points. The selected methods are the same as the method used in Section 3.3.1. Additionally, a direct application of Proposition 2.9, described in Section 4.4.1, is also used.

The comparison is designed to display the efficiency and accuracy of the methods. For each method, the same set of input is given and the time used to produce the results and their accuracy is measured. All methods are implemented in C++ and running on Intel Pentium 4 3.0GHz machine with 1GB of RAM. We took our best effort to optimize every methods. Of all methods, only the Quick Hull employs arbitrary precision arithmetic in order to guarantee the validity of the result. The result from Quick Hull is not used to compare the efficiency but acts as an accurate reference result. The other methods, including our methods, use primitive data type (64-bit floating point) in the computation.

The comparison is conducted on the same object used in Section 3.3.2 (see Figure 3.4). Each object is scaled such that it fits in a unit sphere. This is intended to reduce the effect from force-torque different unit scaling. The sampling of contact points from the objects is carried out using primitive data type. This means that the input itself inherently contains some numerical errors, e.g., contact points sampling from the circle might have their normal direction pointing slightly off-center. For each object, there are $C_{200,4}$ = 64,684,950 queries of frictionless grasps to be tested for the single query approach.

4.4.1 Direct Approach for Frictionless Contact Cases

We directly employ Proposition 2.9 to identify the frictionless grasps. Proposition 2.9 indicates that we have to determine whether one wrench has its negative lie inside the cone of the other wrenches. In frictionless grasp, four wrenches, says $\boldsymbol{w}_1 \ldots \boldsymbol{w}_4$, are considered and hence the cone involved is formed by three wrenches. Let the three wrenches be $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_3$. The cone is represented by the intersection of three half spaces. The plane that bounds each half space contains two wrenches from $\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_3\}$. The plane is oriented such that the other wrench is on the positive side of the plane. Let us assume that $\mathcal{H}_1$ is bounded by the plane containing $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$. The normal vector of the plane is either $\boldsymbol{w}_1 \times \boldsymbol{w}_2$ or $\boldsymbol{w}_2 \times \boldsymbol{w}_1$. The side of $\boldsymbol{w}_3$ with respect to the plane determines the normal vector. After the normal vector of each half space is identified, we test whether $-\boldsymbol{w}_4$ lies outside all half spaces. The entire process is very simple. To the best of our knowledge, this is the fastest method to determine whether four wrenches achieve force closure. This method is selected as a representative of a fastest single query method for frictionless grasps.

Table 4.1: Actual Running Time of frictionless cases.

Test Object	Running Time of NSC-AL(s.)	Running Time Ratio (%)	
		Naive	QD
(a)	19.65	1,139	10,146
(b)	17.98	1,223	6,178
(c)	19.53	1,147	10,268
(d)	19.57	1,142	5,989
(e)	19.41	1,151	6,181
(f)	19.24	1,161	6,296
(g)	19.44	1,151	6,466
(h)	16.57	1,317	8,753
(i)	11.65	1,799	11,773
Avg.	18.12	1,248	8,006

4.4.2 Comparison Result

In Table 4.1, the time used by each method is shown. Each row represents the result from each test object. The first column gives the actual running time of our algorithm, labeled "NSC-AL". The remaining columns give the ratio (in percent) between the actual time used by other methods and our method. The final row shows the average over all test objects. The Quick Hull is not included in the table. This is because, by its nature, the arbitrary precision arithmetic requires tremendous amount of time (more than 1,000 folds of our method). The Quick Hull method is included in the experiment only for validating the results. The result from the Quick Hull is compared to the result of other methods to confirm the validity of the result.

From the result, it can be seen that our method significantly outperforms other methods in terms of efficiency. On average, our method takes time roughly one tenth of the time used by the naive approach.

Accuracy of the result is not less important than efficiency. Theoretically, all methods have been shown to be correct. However, these methods rely on several floating point calculations in which numerical errors are inevitable. We are interested in the effect of numerical stability of each method. Result from the Quick Hull method with arbitrary precision arithmetic library is used as reference for comparison among the other algorithms implemented with limited precision primitive data type. We count the number of faults from each algorithm. Table 4.2 indicates the percentage of the number of faults, both positive and negative, of each method. The first column shows the number of force closure grasps, identified by the Quick Hull algorithm. The remaining columns give the number of faults of each method in percentage.

Table 4.2: Fault on frictionless cases.

Test Object	Number of Grasps	Fault Ratio (%)		
		NSC-AL	Naive	QD
(a)	8,199,395	0.330	-	0.099
(b)	6,961,946	-	-	-
(c)	8,247,103	0.309	-	0.00024
(d)	8,160,057	-	-	-
(e)	8,104,160	-	-	-
(f)	8,010,207	-	-	-
(g)	8,146,578	-	-	-
(h)	5,402,545	0.980	1.009	0.140
(i)	3,036,240	0.121	-	0.00006

The result shows that, our method has similar accuracy with the other methods, i.e., similar number of fault, on a number of test objects. These errors happen in the boundary case. For example, the case when the negative wrench lies near the boundary of the convex hull of the other wrenches. With limited precision, it is possible to incorrectly determine side of a wrench.

From the result, our method shows noticeable inferior accuracy on particular objects: (a) a circle, (c) a donut, and (i) a rectangle. All methods exhibit several faults for the test object (h), a hexagon. These objects have special characteristic. The wrenches of these objects is very likely to cause the boundary case. For example, let us consider a circle. An ideal circle, in fact, cannot be grasped by any number of frictionless contact points, since the line of action of force intersect at the same point. However, the actual data possess some error from the limited precision representation. This causes the line of action to lie slightly off-center. Each contact point then can therefore produce a very small magnitude of torque around the center, which is not zero. Four wrenches from this object could produce a force closure grasp. However, the grasp is a marginal one. A cone formed by the three wrenches has very small volume and the negative of the forth wrench is very near a facet of the cone. Since our method transforms the input several times: rotating, intersecting with a plane and transforming into angular representation. The round off error takes its toll in each operation and could finally result in a fault.

The object (c) is also similar to the circle and the accuracy is also alike. Further examination of the object (i) case reveals that errors are all false positive resulting from having three contact points on a parallel side, e.g., top and bottom side or left and right side. Three contact points on a parallel side generates three wrenches lying on the same plane, since the force components are in the same line. With the forth contact points lying on any other side, the wrenches form a half space. This is a boundary case which is not a force closure. A slight deviation of the three coplanar wrenches could make the grasp become force closure. Again, the transformation in our

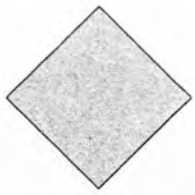

Figure 4.5: Test Objects (j)

Table 4.3: Fault on frictionless cases.

Test Object	Number of Grasps	Fault Ratio (%)		
		NSC-AL	Naive	QD
(j)	974,340	2.50	3.03	0.33

algorithm introduces noticeable number of this deviation.

The hexagonal object (h) causes errors in all methods. The error are all false positive which originates from the same reason as in the rectangle case. The fact that the other methods do not exhibit errors in the rectangle case is because the rectangle is also rectilinear, i.e., the side is aligned to the axis. This makes the force component of the wrench becomes axis-aligned and is represented by an accurate number such as 1 and 0. The other methods which operate the wrench directly without much transformation suffers less round off error. The vast majority of errors from the other methods on the hexagonal object result from the contact point lying on a slanted side of the polygon. To test this hypothesis, we rotate the rectangle by 45 degree (see Figure 4.5), tilting the sides of the rectangle and test all methods again. Table 4.3 displays the result of this case showing the same degree of errors on every method.

In conclusion, our method performs much faster than the other methods, taking time approximately one-tenth of the fastest method, while slightly sacrificing some accuracy. From the example, the error is less than one percent in a malicious objects in which all methods also suffering.

4.5 Summary

We present a novel algorithm for identifying all frictionless force closure grasps of a planar object represented by a set of discrete points. The algorithm utilizes force dual representation of a wrench. A condition of force closure in force dual representation is represented. The condition can be efficiently computed using an output sensitive data structure which result in a computational

complexity of $O(n^3 \lg^2 n + K)$. It is also shown empirically that the algorithm also works well in practice; it outperforms other methods in terms of speed while still maintaining the same level of accuracy.