CONVEX HULL OF SURFACE PATCHES: CONSTRUCTION AND APPLICATIONS

by

Chao-Kuei Hung

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(Computer Science)

December 1995

Dedication

To my parents:

洪振坤　　蕭金葉

whose encouragement made the completion of this thesis possible.

Acknowledgements

Many ideas in this thesis were first conjectured by Prof. Doug Ierardi, my advisor. Doug is also my oracle for pointers to other information sources, besides being a wonderful source of information himself in subjects ranging from algebra to Linux, from philosophy to Deep Space Nine. The interesting application in stable poses and capture regions would not exist without the helpful discussions with and references from Prof. David Kriegman, who pioneers the study in this field. I am fortunate to have Prof. Francis Bonahon in my committee for consulting mathematical problems. Prof. Aristides Requicha's input lead to the discussions on practical issues and finally an implementation, which in turn enhanced my familiarity with `C++`. I owe my gratitude to Prof. Ken Goldberg, who first posed the convex hull problem to me. He and Anil Rao subsequently provided me with several important references on grasping and width functions. Profs. Ram Nevatia and Ming-Deh Huang, along with my other committee members, helped shaping and focusing the thesis.

I was first initiated into the world of duality by Prof. Alfred Inselberg, although in a different setting. His "tangential" ideas in parallel coordinates had a subtle but definite impact on the design of the duality in this thesis. I am grateful to him, Avijit Chatterjee, and Yui-Bin Chen for much valuable advice during the early stage of my Ph.D. study. Hsun-Ko Chan and Shih-Hui Lu, a warm and wonderful couple, pushed me gently back to the track when Professor Inselberg had to leave USC and before I found another supportive advisor. Many friends around me, especially Mourad Zerroug, Chih-Wen Hsueh, and dear Martha Escobar, have cooled me down and encouraged me during the typical manic-depression cycles of a Ph.D. student.

Finally, it has always been my wish to express my gratitude towards the gurus on the Internet who substantiate the concept of free software. To me, the collaborative nature of such programs as Linux, gnu*, RLaB, PARI/gp, and many others result in their influential and *lasting* contributions to the computing community. It is they who have been practicing software reuse and citing some of the most relevant papers without necessarily advertising the fact, and on to whose shoulders I wish to climb.

Contents

List Of Figures

Abstract

We solve the problem of constructing the convex hull of piecewise smooth objects in 3-D. This work was motivated by a need to compute the width function of parts in a manufacturing environment where an oblivious grasping algorithm drives a pair of parallel jaw grippers to orient parts on a conveyor belt. We show that constructing a planar graph describing the convex hull takes randomized expected time $O(N^{2+\epsilon})$ for arbitrary $\epsilon > 0$, if the input consists of N algebraic faces of bounded degree each bordered by a bounded number of subfaces.

We use duality in our construction. This approach allows us to solve two other related problems—finding the stable poses and capture regions of a piecewise smooth object. Such information can be utilized in a model-based vision system to reduce the complexity of the algorithm for recognizing immobile parts. The duality approach provides other advantages as well, such as simplifying the treatment of algebraic singularities and unifying the treatment of faces of different dimensions.

In practice many industrial parts can be described solely in terms of quadratic surface patches cut out by planes. We show that in this case the algebraic overhead in constructing the convex hull is significantly lower than estimated in the general algorithm. For example, the algebraic complexity of the systems of equations in the stable pose problem is reduced by almost two orders of magnitude compared to existing literature. In this specialized context, it is feasible to adopt a simple scheme for safely processing coincidences, as demonstrated in an implementation in `C++` based on the dual approach.

Chapter 1

Introduction

1.1 Problem Statements and a Brief Survey

Computational geometers have exhaustively studied the problem of constructing convex hulls of discrete point sets in various dimensions. Indeed, Chazelle give optimal algorithms to this problem for arbitrary dimension in [8]. The convex hull of curve segments, however, have received little attention until late eighties. There is little existing literature which addresses the issues in solving the three dimensional, curved version of this problem.

We solve the problem of constructing the convex hull of piecewise smooth objects in 3-D using duality. We then apply these methods to two other related problems arising from computer vision and manufacturing.

1.1.1 Convex Hulls of Curved Objects

Convex hull algorithms for the curved version of a simple planar polygon have been proposed independently by several researchers. Shäffer and Van Wyk [43], Dobkin and Souvaine [13], and Bajaj and Kim [5] each give an optimal (linear time) algorithm, although they use different terms such as piecewise smooth Jordan curves and splinegon. Their emphasis is more on the combinatorial aspect than on the algebraic computations, although such issues are also addressed. In contrast and complementarily, the work of Kriegman, Yeh, and Ponce in [32] focus on the algebraic aspect and implementation issues by studying the convex hull of a single closed algebraic curve in the plane.

We present a much simpler algorithm for achieving the same optimal results as in [43, 13, 5]. It is shown further that if the input consists of unordered curve segments along with isolated points, the convex hull can still be computed in optimal time $\Theta(N \log N)$.

In extending his results to 3-D and to related problems, Kriegman [30, 31] also discusses the algebraic issues. Schwartz and Sharir [45] show that the complexity of the convex hull of a set N spheres is $\Theta(N^2)$ in the worst case. Boissonnat, Cérézo, Devillers, Duquesne, and Yvinec [6] later present an algorithm for the convex hull of a set of spheres in higher dimensions. They apply duality in a special way which allows the result to be generalized to homothetic objects. However, we have been unable to locate other literature which give a more general result on this problem.

We show that constructing a planar graph describing the convex hull takes randomized expected time $O(N^{2+\epsilon})$ for arbitrary $\epsilon > 0$, if the input consists of N algebraic faces of bounded degree each bordered by a bounded number of subfaces. Several operations can be performed on the convex hull without its explicit construction. Point classification against the hull can be effected in time $O(N \log N)$. The supporting cone from a point to the convex hull is shown to have a structural complexity of $\Theta(N)$ and can be constructed in time $O(N \log N)$.

1.1.2 Stable Poses and Capture Regions

Given a description of a piecewise smooth object and the location of its center of mass, its gravitational potential energy is a function of its "pose." The stable pose problem seeks to determine those poses of the object at which the potential energy function assumes local minima. This problem occurs in model-based vision systems where such information can be used to reduce the complexity of recognizing a static object resting on a plane.

Kriegman pioneered the study of stable poses of piecewise smooth objects in [30]. The combinatorial complexity is not given explicitly but as he pointed out, the dominant factor is the algebraic part. Experimental results show that it takes roughly two days on a Sun SPARCStation 1 to process figures similar to Figures 7.1 and 7.2 using homotopy method for root-finding.

We show that, as a corollary to the complexity of the convex hull, the number of stable poses of a piecewise smooth object having N faces each with a bounded number of subfaces, is $O(N^{2+\epsilon})$. As a result of theoretical interest, it is shown that they can be computed in the same amount of randomized expected time with an algebraic overhead.

A direct extension to the stable pose problem is the computation of the capture regions—the regions in the parameter space within each of which the object will eventually converge to one stable pose starting from any pose. A general planner needs this information in controlling a robot manipulator to place an object in the plane in the presence of mechanical uncertainty.

In [31] Kriegman extends the idea to computing the capture regions. The analysis is based on stratified Morse theory, under the assumption that the potential energy function is Morse. Implementations for either closed individual algebraic surfaces or general polyhedra are given. The combinatorial time complexity in the latter case is further shown to be $O(N^2)$.

We show that the capture regions can be described by a planar graph of size $O(N^{4+\epsilon})$ and can be constructed in corresponding time.

1.1.3 Width Function and Tight Bounding Boxes

The distance between the pair of tight supporting planes which bound a given object is a function of their orientation. Such information is essential to the algorithm which generates oblivious grasping plans to operate a pair of parallel jaw grippers in a parts feeder.

Parts orienting and the width function are extensively studied by Mason, Goldberg and Taylor in [36], Goldberg in [20], Rao and Goldberg in [41], Rao, Kriegman, and Goldberg in [40], and by Chen, Hung, and Ierardi in [11]. An efficient algorithm for finding the overall width of a 3-D point set in time $O(N^{8/5+\epsilon})$ without constructing the entire width function is given by Chazelle, Edelsbrunner, Guibas, and Sharir in [10]

We study the width function and generalize the idea to the description of the area or volume of the tight bounding box as a function of its orientation relative to the object.

1.1.4 Practical Issues

The duality approach eases the treatment of several practical issues in the 3-D versions of all of the above problems. The tedious analysis of sixteen cases involving combinations of up to three faces of various dimensions is reduced to that of three cases. We are able to describe how to give an exhaustive listing of singularities up to a fixed degree in the dual space, and give examples of what they correspond to in the primal space.

As a practical substitute for the asymptotically efficient algorithms in treating all three problems, we also propose a simple algorithm for sampling points on the convex hull. An intuitive measure of the precision of sampling density is shown to have a simple geometric counterpart in the dual space. Its output may be further processed to produce a planar graph describing the exact boundary of the convex hull, the potential energy function, or the width function, subject to "small" missing features as determined by the precision measure.

In practice many industrial parts can be described solely in terms of quadratic surface patches cut out by planes. We show that in this case the algebraic overhead in these problems is significantly lower than estimated in the general algorithm. Systems of nine quadratic equations in nine variables are replaced by systems of three quadratic equations in three variables, resulting in a reduction of the total degree from 512 to 8.

1.2 An Outline

We first revisit the already-solved two-dimensional version of the convex hull problem in Chapter 2. We introduce concepts which will be elaborated later on in this simpler context and show how they significantly simplify the logic of the optimal algorithms.

The mathematical foundations of the dual approach begin with a review of several forms of traditional duality in Chapter 3. Their relationships are examined and their strengths combined in Chapter 4 to give a definition of *inverted pedal*, which retains the connection between the dual and the convex hull and at the same time is constructively defined from a computational point of view. The special case of quadratic hypersurface patches cut by hyperplanes is closely studied in Chapter 5.

The results here are essential to the significant reductions in the algebraic aspect of algorithms dealing with typical industrial parts.

The main algorithm for convex hull construction and several other algorithms directly related to the convex hull are presented in Chapter 6. The first application is studied in Chapter 7, where Kriegman's discussions and results on stable poses and capture regions are improved upon. We then investigate the second application of convex hull construction—the width functions and tight bounding boxes in Chapter 8. The contributions and future extensions of this thesis are summarized in Chapter 9.

Appendix A are the implementation notes of our `C++` program. Also presented here is an elegant design of the matrix class which provides an efficient and neat interface for submatrix manipulations without the need of keeping reference counts.

Chapter 2

A New Look at the Planar Problems

To motivated and introduce our approach, in this chapter we first consider the simple 2-D convex hull problem of ordinary and curvilinear simple polygons. The point set version was already studied in [37, 22, 34] and the curved version in [43, 13, 5]. In both cases the problems have long been known to have time complexity $\Theta(N)$, in the latter case with an additional overhead given by the algebraic complexity of the bounding curve segments.

The ideas introduced in our versions of the solutions lead to significant simplifications of the control flow over the existing algorithms, especially in the curvilinear case. They also lead to the use of and guide in modifying the classical duality definitions and properties, and eventually enable us to provide a general algorithm in higher dimensions. We also use these algorithms to introduce a few notations and give examples of the algebraic issues raised by non-linearity, which we discuss in details in later chapters.

2.1 Simple Polygons Bounded by Line Segments

We begin by the rectilinear version of the convex hull problem. Let $K = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N\}$ be the set of vertices of a simple polygon given in counterclockwise direction around its interior. Figure 2.1 shows a `C++` implementation of our algorithm for finding its convex hull. We employ an auxiliary data structure `PoinDStack` for storing the vertices in a stack. To save memory usage as well as typing, it is actually the *indices* to the vertices plus a *reference* to the array of vertices (class `Vector<PointD>`) which are stored in the stack. Given an index possibly out of

```
int right_turn(PointD const & a, PointD const & b, PointD const & c)
    { return (b-a).cross(c-b) < 0; }

class PointDStack : public Stack<int> {
protected:
    Vector<PointD> const & host;
    PointD v(int i) const { return host[(i+host.nelts()) % host.nelts()]; }
public:
    PointDStack(Vector<PointD> const & h) : host(h) {}
    int snared_beneath(int i) const
        { return v(top()).cross(v(i)) <= 0; }
    int prev_concave(int i) const { return head->next &&
        ::right_turn(v(head->next->data), v(top()), v(i)); }
    int right_turn(int i) const
        { return ::right_turn(v(i-2), v(i-1), v(i)); }
    int snared_in_pouch(int i) const { return ! prev_concave(i); }
};

PointDStack PolygonHull(Vector<PointD> & vertex)
{
    ...
    PointDStack CH(vertex);
    CH.push(i = x.max); // starting from an extremal vertex
    CH.push(stop = ++i %= N); // and ending at the same vertex
    while (++i % N != stop) {
        if (CH.right_turn(i))
            while (CH.snared_in_pouch(i)) ++i;
        else
            while (CH.snared_beneath(i)) ++i;
        while (CH.prev_concave(i)) CH.pop();
        CH.push(i);
    }
    CH.pop(); // remove the starting point, which was entered twice
    return CH;
}
```

Figure 2.1: Convex hull of a (planar) simple polygon

array bound (but not "too negative"), its member function `v()` retrieves the actual coordinates of the vertex by regarding the array as circular.

2.1.1 Preprocessing

To find $\texttt{conv}\,K$, the convex hull of K, we first choose a convenient coordinate system in which the origin $\mathcal{O}$ lies in $\texttt{int}\,\texttt{conv}\,K$, the interior of $\texttt{conv}\,K$. One obvious choice is to use $\mathbf{c}_K = \frac{1}{N}\sum_i \mathbf{p}_i$ as the new origin. It is also easy to show that $\mathbf{b}_K$ the center of the tight bounding box B of K whose sides parallel the coordinate axes lies in the interior or (with probability 0) on the boundary of $\texttt{conv}\,K$. Though not guaranteed to lie in the interior, the latter alternative does not suffer from skew point distributions. We choose $\frac{1}{2}(\mathbf{c}_K + \mathbf{b}_K)$ as the new origin as it is easy to show that it lies in the box concentric with B and half of its size, and that it lies in $\texttt{int conv}\,K$ unless all vertices are collinear.

We will assume without loss of generality that $\mathbf{p}_1$ is an extremal vertex, which is guaranteed to appear on the boundary of $\texttt{conv}\,K$. For ease of presentation we further assume that the coordinate axes are so rotated that $\mathbf{p}_1$ lies on the positive x-axis, although in reality we will not have to actually carry out this rotation. Finally let the polar angles of the vertices be normalized to fall within $[0, 2\pi)$.

2.1.2 The Main Loop

The algorithm (Figure 2.1) begins with a stack CH containing $\mathbf{p}_1$ and $\mathbf{p}_2$ with $\mathbf{p}_2$ on top, and proceeds in the given order of the vertices. We now state the invariants of main loop. At each iteration on entrance to the body of the main loop let $\mathbf{p}_k$ be the vertex on the top of CH. For every predecessor $\mathbf{p}_i$ of $\mathbf{p}_k$ (that is, $i \leq k$), exactly one of the following two statements is true:

- $\mathbf{p}_i$ is known to lie in $\texttt{int conv}\,K$ even with our limited knowledge up to this point; or

- $\mathbf{p}_i$ lies on the boundary of a convex "sector" beginning with $\overline{\mathcal{O}\mathbf{p}_1}$ and ending with $\overline{\mathcal{O}\mathbf{p}_k}$, *and* it is contained in CH in sorted order by its polar angle.

This is obviously true just prior to the first iteration of the loop.

To show that the above is indeed invariant, let us consider what happens until the end of one iteration of the outer loop. For now we ignore the possibility of having a radial segment by just mentioning that it can be avoided with probability one if we perturb the origin. Then the following cases cover all possibilities at this point: (See also Figure 2.2)

1. $\overline{\mathbf{p}_{k-1}\mathbf{p}_k}$ is counter-clockwise (a forward edge).
   (a) $\overline{\mathbf{p}_k\mathbf{p}_{k+1}}$ makes too sharp a right turn that it enters the most recent pouch.
   (b) $\overline{\mathbf{p}_k\mathbf{p}_{k+1}}$ makes too sharp a left turn that it goes clockwise (backward) and gets "snared" beneath $\overline{\mathbf{p}_{k-1}\mathbf{p}_k}$.
   (c) otherwise.
2. $\overline{\mathbf{p}_{k-1}\mathbf{p}_k}$ is clockwise (a backward edge).
   (a) $\overline{\mathbf{p}_k\mathbf{p}_{k+1}}$ makes too sharp a right turn that it enters the most recent pouch.
   (b) otherwise.

Note that there is nothing in case 2 which corresponds to case 1(b). On the other hand both cases 1(c) and 2(b) entitle $\mathbf{p}_{k+1}$ to be collected into the stack. It is straightforward to verify that the `if` part in the main loop processes cases 1(a) and 2(a) alike, whereas the `else` part processes case 1(b), both discarding succeeding vertices which cannot possibly lie on the hull. Neither of cases 1(c) and 2(b) is affected by either branch. Thus at the end of the `if` statement, all interior vertices are discarded in each of the 5 cases, and the current vertex is one which we would like to push on to the stack. Finally we proceed to remove the vertices which are now made concave by the addition of the current vertex. When this stops all vertices known to be interior to the hull up to now are removed and the invariant holds again.

2.1.3 Correctness and Complexity

The correctness of this algorithm follows from the established loop invariant and the fact that the iterations end when $\mathbf{p}_1$ has for a second time been pushed onto CH. At this moment the invariant can be rephrased as saying that vertices in CH are exactly the hull vertices in sorted order, with $\mathbf{p}_1$ occurring twice. During the algorithm, discarding vertices and stack operations are sufficient to account for all (inner and outer) loop iterations. This number is obviously $O(N)$ since each vertex contributes at most two. The algorithm is therefore $\Theta(N)$ since each testing operation takes constant time.

We note that this algorithm proceeds *around the origin*, unlike [22] which proceeds *along some straight line*. Thus there is no need to construct the upper and lower hulls separately. Moreover, the earlier assumption that $\mathbf{p}_1$ is a hull vertex and that $\overline{\mathcal{O}\mathbf{p}_1}$ is the polar axis can in reality be removed. Simple modular arithmetic on the vertex indices as performed by the member function `v()` solves all of these three problems simultaneously.

2.2 Simple Polygons Bounded by Algebraic Curve Segments

We term a simply-connected planar region bounded by a finite number of segments of algebraic curves as *a curved polygon*. The previous algorithm can be generalized to find the convex hulls of curved polygons whose edges are given in a circular order around their interiors. Assume that each curve segment is given as an implicit equation of total degree at most D in x and y. Again we shall reduce the number of possible cases which arise in the algorithm and simplify the logic without introducing global flags whose values are altered by function calls and by which the control flow is determined.

Figure 2.3 shows the main loop of a `C++` implementation. Here the input is an array of `Edge`s instead of `Vertex`es, and the auxiliary data structure is now a stack of `Edge`s. We leave out the details of the `Edge` class and only remind the user that it provides enough information for the geometric computations requested in the main loop, which we shall discuss in details promptly.

2.2.1 Preprocessing

Continuing along the line of thought in the rectilinear version, we shall proceed *around the origin*. To this end, we first break each edge into pieces at inflection points, so that each resulting piece is either entirely convex (from the origin towards outside) or entirely concave. We further break each piece into yet shorter pieces such that along each, the points have either monotonically increasing or monotonically decreasing polar angles. We finally break the pieces again if the polar angles of the points on it have a span greater than or equal to π. This last preprocessing ensures an unambiguous definition of the ordering of two points by their polar angles, where the difference between two angles is normalized to be within $(-\pi, \pi)$.

By how much did we increase the number of input curve segments in doing so? Suppose one input curve segment γ is given as the zero set of a degree D polynomial $f(x, y) = 0$. The first step breaks it at no more than D pieces. The second step further breaks γ at the *turning points/* on γ, where the tangent line to γ passes $\mathcal{O}$ and satisfies $x\frac{\partial}{\partial x}f + y\frac{\partial}{\partial y}f$, an equation of degree at most $D+1$. By Bezout's theorem, there are at most $D(D+1)$ such points. We can tighten the bound, however, if we consider f as a projective curve and perform a projective transformation which brings $\mathcal{O}$ to any point at infinity, say the one on the y-axis. In the new coordinate system the turning points correspond to the extremal points, or *folds* along the x-axis, which can be no more than D, the maximum number of points of intersection between γ and an arbitrary straight line. For the last step of breaking pieces longer than π, we dispose of the new coordinate system used only in the previous proof and go back to break γ at all of its intersections with the polar axis (in the original coordinate system). This creates at most D more breaking points. In practice we break γ into monotonic segments of angular span no more than $\pi/4$. In summary the number of segments does not increase by more than a factor of $4D$. We make one last surgery on the input. An extremal point is found and inserted as one more vertex which is guaranteed to appear on the hull. To simplify notations, we will thus assume from now on that the input curved polygon K consists of N segments which have already been pre-processed.

In the algorithm recorded in each segment `edge[i]` are `start` and `stop`, its two end points whose order is determined by tracing the curve in the order of the edges prescribed by the input. It is then meaningful to speak of an edge as being `forward` or `backward`, depending on whether the normalized difference between the polar angles of `start` and `stop` is less than or greater than zero. We rotate the indices of the edges so that the extremal vertex serves as the `start` vertex of the first edge and `stop` vertex of the last edge, both of which are certainly `forward`.

2.2.2 The Main Loop

As before we will proceed counter-clockwise around the interior of the curved polygon. In the present case, however, it is necessary to push the edges, not just their end points, to the stack. Moreover, new flat edges, namely segments of the common supporting lines between adjacent hull edges, may emerge. On the other hand, we shall see that concave edges can be treated exactly the same way as the convex or flat ones are as far as the control flow of the main loop is concerned.

We first push `edge[1]` onto the stack, noting that even if it gets popped off the stack later on, the replacing edge (which is necessarily a supporting line segment) will always have the same `start` vertex. If it happens to be concave, we push instead the lid which covers it, without modifying the concave edge itself.[1] Since we are pushing an entire edge on to the stack, we have already processed two vertices and the second edge can be treated as the rest of the edges as described below.

At each iteration prior to entering the body of the main loop, let $\hat{\gamma}$ be the edge on the top of CH. Note that $\hat{\gamma}$ may or may not be one of the input edges—it may just be a common supporting line of an input edge and a vertex. In any case let `edge[k]` be the edge in the original input which generated the `stop` vertex of $\hat{\gamma}$. In other words, `edge[k]` at least shares the same `stop` vertex with $\hat{\gamma}$ if they do not in fact coincide. We claim a loop invariant similar to that in the linear case with an additional property. For every predecessor γ_i of γ_k $(i < k)$, exactly one of the following two statements is true:

- γ_i is known to lie in $\texttt{int conv}\, K$ even with our limited knowledge up to this point; or
- at least part of γ_i lies on the boundary of a convex "sector" beginning with the ray from $\mathcal{O}$ to `edge[1].start` and ending with the ray from $\mathcal{O}$ to `edge[k].stop`.

Furthermore, all edges on the stack are either portions of some `forward` convex edges in the input, or "lids" of `forward` concave edges, or common supporting line segments of the input edges.

[1] It is very important that we should not replace the concave edge by its lid because later on we need to know the tangent direction at its end points for determining whether the next edge goes "above" or "below" it.

Supposing that the invariant holds upon entry of the loop body at some iteration, we describe how the algorithm manages to maintain its validity at the end of the iteration.

1. `edge[k+1]` is a `forward` edge.

2. `edge[k+1]` is `backward` and goes beneath `edge[k]`. (Figure 2.4, left.)

3. `edge[k+1]` is `backward` and goes above `edge[k]`. (Figure 2.4, right.)

Case 1 does not require any processing yet at this point. In case 2, we have a snare situation like the linear case. Note that the beneath test is performed against `edge[k]` and not $\hat{CurveSeg}$ since the latter may just be a lid whereas it is the tangent of the former at `edge[k].stop` which defines beneath and above. Also note that the convexity of `edge[k]` is irrelevant in performing the beneath test. The snare situation is "harder" to break than the linear case because we consider a `backward` edge to be snared regardless of its polar angle relative to `edge[k].stop.angle()`. As always, a `backward` edge will eventually be covered by a `forward` one and hence can be safely discarded. In case 3, `edge[k+1]` may or may not get snared into a pouch as in the linear case. If it does, however, it is not necessarily the most recent pouch into which `edge[k+1]` goes. Thus we skip subsequent `backward` edges instead, popping off all `forward` edges which were previously on the stack but are now completely shadowed by these `backward` edges.

At this point the current edge is a `forward` one and its starting angle is greater than the starting angle of the topmost edge on CH no matter which of the three cases we began with earlier. We compute the common supporting line segments of the current edge with the topmost edge on CH and keep popping off those edges which are completely shadowed by the common supports. When we eventually arrive at an edge which is not completely shadowed. After cutting it appropriately, we push the cover as well as the current edge onto the stack if they are not empty. The invariant is restored at this point.

2.2.3 Correctness and Complexity

Arguments similar to those in the rectilinear case show that this algorithm correct and that the combinatorial complexity is $O(N)$. The algebraic analysis earlier shows that the the combinatorial complexity is further increased by a factor of $4D$. Finally there is an extra factor incurred by the algebraic computations on the edges during preprocessing and in the main loop, of which finding the common support is the dominant one. In practice this last factor may arise in the form of numerical computations and therefore depends on the desired precision.

Note that all the comparison operators for the angles can be unambiguously defined because each edge has a small span of polar angles. The subroutines not shown in the algorithm are purely functional and algebraic, and do not require more information than the explicitly specified parameters. In other words, they have no effects on and are not affected by the control flow of the program.

2.3 Hints and Heuristics in the Design of Algorithms for Higher Dimensions

In comparison to the existing algorithms, we observe that our versions achieve a considerably simplified flow control mostly because the latter proceed around the origin. To put the comparison in a more common perspective, we can also say that the traditional algorithms proceed "around a point at infinity," much as saying that an orthographic projection is in fact a perspective projection with a point at infinity as its perspective. Consequently we notice the following distinctive features in our algorithms:

1. The coordinate system has to be chosen in such a way that the origin falls in the interior of the convex hull in order for the algorithms to work. It is, however, irrelevant whether the origin falls in the interior of the polygon itself. The traditional algorithms are not affected by the choice of the straight line with respect to which the upper hull and lower hull are defined.

2. Interesting (or rather, annoying) events occur at points on the curve segments where the tangent to the curves pass the origin. These are therefore places where we break the edges in order to ensure the monotonicity of the polar angles of points on each edge. With the traditional algorithms, the corresponding events occur at the folds, or where the tangents are perpendicular to

the straight line separating the upper and lower hull. The idea of monotonicity with respect to a straight line is either implicit or stated in these approaches.

3. In our algorithms as well as some traditional algorithms the edges are also broken at the inflection points.

We notice that these features are also present in one or another dual construct in mathematics. It is thus our goal in the next chapter to set out reviewing these concepts as they are related to our problem.

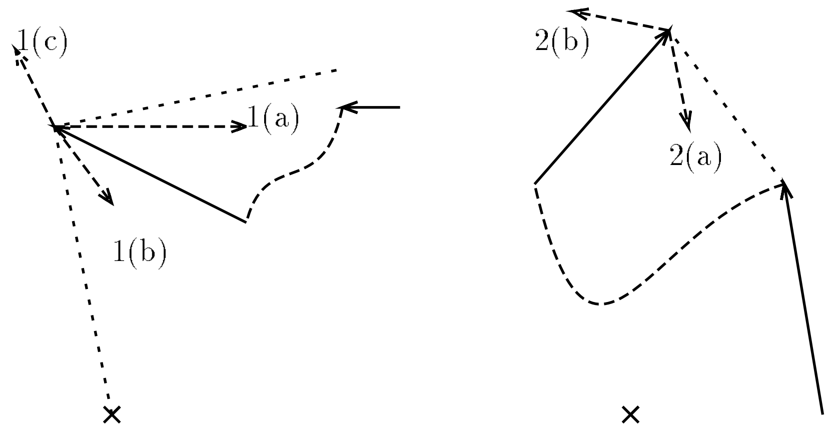

Figure 2.2: The cases of the next edge to be examined. The dashed lines indicate the (possible) edges on the polygon; the dotted lines separate the cases. The origin is marked by an "x."

```
EdgeStack CurvePolygonHull(Vector<Edge> & edge)
{
    EdgeStack CH;
    CH.push(edge[stop = i = x.max].cover());
    while ((++i %= N) != stop) {
        if (edge[i].backward())
            if (edge[i].beneath(edge[(i-1+N)%N]))
                for (; edge[i].stop.angle() < CH.top().stop.angle() &&
                    edge[i].start.angle() > CH.top().stop.angle()+M_PI ||
                    edge[i].backward(); ++i %= N);
            else
                for (; edge[i].backward(); ++i %= N)
                    while (edge[i].stop.angle() < CH.top().start.angle())
                        CH.pop();
        assert(! edge[i].backward() &&
            edge[i].start.angle() > CH.top().start.angle());
        while (!CH.isEmpty()) {
            cover = edge[i].support(CH.top());
            if (cover.start.angle() > CH.top().start.angle()) break;
            CH.pop();
        }
        CH.push(CH.pop().cut_after(cover.start.angle()));
        if (cover.start.angle() < cover.stop.angle()) CH.push(cover);
        if (edge[i].stop.angle() > cover.stop.angle())
            CH.push(edge[i].cut_before(cover.stop.angle()));
    }
    CH.pop();
    return CH;
}
```

Figure 2.3: Convex hull of a curved polygon

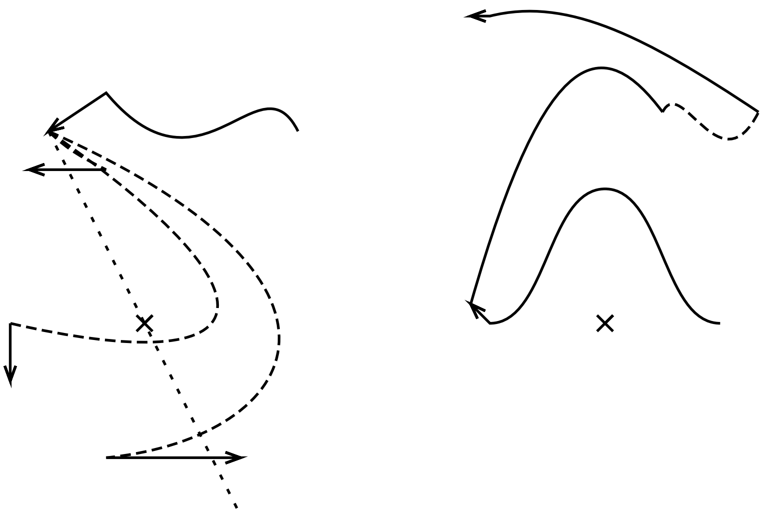

Figure 2.4: Left: backward edges going beneath—ways to break the snare situation. Right: backward edges going above.

Chapter 3

Classical Forms of Duality and Math Preliminaries

The term "dual" in mathematics has a wide variety of definitions in different contexts. In this chapter we review several of them upon which our algorithm is based, and observe the relationships among them. The foundation of our algorithm is established by giving an alternative description of the topologically derived sets of the polar set. Finally we review terminologies from combinatorial topology and specialize towards the algorithmic needs which will arise in later chapters.

3.1 Duality in Projective Geometry

3.1.1 Notations and Terminology

Conventions for Overloading Notations

The d-dimensional projective space P^d consists of *equivalence classes* of $(d+1)$-tuples which have at least non-vanishing entry. Two such tuples are equivalent if one is a non-zero scalar multiple of the other. By bracketing the entries of a tuple and displaying it as a column vector, e.g. $[\xi_0, \xi_1, \ldots, \xi_d]^T$., we denote the equivalence class to which the tuple belongs, or the projective point it represents. To avoid introducing an overwhelming number of symbols in later discussions and proofs, we will agree upon the convention of overloading notations as follows. In general let A be a $d+1$ by k matrix enclosed in brackets where $k < d+1$. The same symbol

A will also be used to denote the linear subspace spanned by its column vectors in contexts where sets are required.

Embedding and Improper Points

There are an infinite number of ways in which the Euclidean d-space E^d may be embedded in P^d so that many of the nice properties in P^d carry over to E^d. One natural choice is to identify each $[\xi_0, \xi_1, \ldots, \xi_d]^T \in P^d$ for which $\xi_0 \neq 0$ with $(\frac{\xi_1}{\xi_0}, \frac{\xi_2}{\xi_0}, \ldots, \frac{\xi_d}{\xi_0})^T \in E^d$. A point identified with some point in E^d is called a *proper point*; and those which left unidentified with any Euclidean point are called *improper points*. In every embedding of E^d in P^d, the set of improper points is a $(d-1)$-dimensional projective subspace of P^d. This set is called the *improper hyperplane* with respect to the particular embedding.

Hyperplanes

The hyperplane

$$H = \{[\xi_0, \xi_1, \ldots, \xi_d] \ : \ u_0\xi_0 + u_1\xi_1 + \cdots + u_d\xi_d = 0\}$$

is uniquely determined by the $d+1$ coefficients and also uniquely determines these coefficients up multiplication by a non-zero scalar. Thus H may be uniquely associated with the projective point $[\xi_0, \xi_1, \ldots, \xi_d]$. We denote H by $[u_0, u_1, \ldots, u_d]$, the equivalence class of the $(d+1)$-tuple displayed as a row vector. The point $[u_0, u_1, \ldots, u_d]^T$ and the hyperplane $[u_0, u_1, \ldots, u_d]^T$ are *projective duals* to each other.

Duality between Linear Bundles and Linear Subspaces

Let B be an m by $d+1$ matrix where $m < d+1$. The set of hyperplanes whose $(d+1)$-tuples are linear combinations of the row vectors of B form a *linear bundle of dimension* $r-1$ where r is the rank of the matrix. Accordingly we will overload the symbol B for also representing this linear bundle in a context where a set is required. This notation will not conflict with that for linear subspaces as long as we work in a projective space of fixed dimension d. The intersection of all the hyperplanes belonging to B is the null space of B and is called the *carrier* or *kernel* of B (as a set). In general a linear bundle $B_{m,d+1}$ dualizes to the linear subspace $B^T_{d+1,m}$ and vice versa.

3.1.2 Hypersurfaces of the Second Order

A *hypersurface of the second order*, or simply quadratic hyper surface, is the set of points

$$\{[\xi_0, \xi_1, \ldots, \xi_d] \;:\; \sum_{i,j=0}^{d} q_{ij}\xi_i\xi_j = 0\}$$

for some fixed $q_{ij}, 0 \leq i,j \leq d$ with $q_{ij} = q_{ji}$. This hypersurface is uniquely determined by the real symmetric matrix whose entries are these q_{ij}'s; and it can be shown that, in cases we will ever deal with, this hypersurface also uniquely determines such a real symmetric matrix up to multiplication of all entries by a non-zero scalar. We will thus use $Q = [q_{ij}]$ the bracketed version of the matrix (q_{ij}) to denote its equivalence class, namely those which are its non-zero scalar multiples. The rank and eigenvectors are invariant within each equivalence class and are therefore used to define the *rank* and *eigenvectors* of the quadratic hypersurface itself. If the rank of the hypersurface is less than $d+1$, it is said to be *degenerate*; otherwise it is *non-degenerate*.

Our abuse of notations is completed by using the square matrix Q for also referring to the quadratic hypersurface itself in a set context. In the rest of this subsection Q will always be a real symmetric matrix which refers to some hypersurface of the second order and its representing matrix.

Tangent Hyperplanes and Polars

If $\mathbf{x}$ is a point on Q, then $(Q\mathbf{x})^T$ represents the tangent hyperplane of Q at $\mathbf{x}$. If, on the other hand, $\mathbf{x} \notin Q$, then all of the points of tangency on Q from $\mathbf{x}$ fall on the same hyperplane, namely $(Q\mathbf{x})^T$. In either case, the hyperplane $(Q\mathbf{x})^T$ is called the *polar* of the point $\mathbf{x}$, and the point $\mathbf{x}$ is called the *pole* of the hyperplane $(Q\mathbf{x})^T$, with respect to Q.

Dual

If Q is non-degenerate, or equivalently the determinant of any representative of Q is non-zero, then Q^{-1} exists and is also real symmetric. In this case Q^{-1} again represents a hypersurface of the second order, which we call the *projective dual* of

Q. The duals of the points on Q, which are themselves hyperplanes, are exactly the tangent hyperplanes of the projective dual of Q, and vice versa.

3.2 Pedal Curves and Pedal Surfaces

3.2.1 Definitions and Properties

Let $\gamma(t)$ be a planar curve. The foot of the perpendicular from $\mathcal{O}$ to the tangent line to $\gamma(t)$ at $\gamma(t)$ is given by $(\gamma(t) \cdot N(t))N(t)$ where $N(t)$ is the unit normal vector at $\gamma(t)$. As t varies, this point traces out a curve called the *pedal curve of* γ *with respect to* $\mathcal{O}$.

The derivative of the pedal curve of γ with respect to the arc length parameter of γ is

$$\delta'(t) = -\kappa((\gamma \cdot T)N + (\gamma \cdot N)T)$$

where $\kappa(t)$ and $T(t)$ are the curvature and unit tangent vector of γ at t, respectively. The second factor vanishes exactly when γ passes the origin. It is seen that the pedal curve is regular except at those points corresponding to the inflection points of the original curve.

Similarly, the pedal surface of a surface in space is the loci of the foots of perpendiculars from $\mathcal{O}$ to the tangent planes to the original surface. A parabolic curve on the original surface corresponds to a cuspidal edge on the pedal surface.

3.2.2 Direct Generalizations and a Variant

Pedals in Higher Dimensions and in Degenerate Cases

On the one hand, we may generalize the idea of pedal curve and surface to higher dimensions. Define the pedal hypersurface of a smooth hypersurface to be the loci of the foots of perpendiculars from $\mathcal{O}$ to the tangent hyperplanes to the original hypersurface.

On the other hand we may also agree upon the convention that in case the pedal fails to be a hypersurface at all, we shall still refer to this construct as the *pedal* *representation* of the original hypersurface. In particular the pedal representation of a hyperplane

$$H = \{\mathbf{x} \;:\; \mathbf{n} \cdot \mathbf{x} = c\}$$

is a single point

$$\mathtt{ped}\, H = \frac{c\mathbf{n}}{|\mathbf{n}|^2}.$$

Pedal Representation of a Set of Hyperplanes

Let $\mathcal{H}$ be a set of hyperplanes. We shall use the same notation $\mathtt{ped}\, \mathcal{H}$ to denote the set

$$\{\mathtt{ped}\, H \;:\; H \in \mathcal{H}\},$$

knowing that the the context will make the meaning of this notation unambiguous.

Inverted Pedal of Hyperplanes

The pedal representation $\frac{c\mathbf{n}}{|\mathbf{n}|^2}$ of a hyperplane is a bit awkward to work with. If c is non-zero, we may scale the equation so that $c = 1$. In this case we may further apply the *inversion* map:

$$\begin{aligned} \mathtt{inv} : E^d - \{0\} &\longrightarrow E^d - \{0\} \\ \mathbf{x} &\longmapsto \frac{1}{|\mathbf{x}|^2}\mathbf{x} \end{aligned}$$

to the perpendicular foot and obtain the vector $\mathbf{n}$. We shall refer to this representation of $H = \{\mathbf{x} \;:\; \mathbf{x} \cdot \mathbf{x} = 1\}$ by $\mathbf{n}$ as the *inverted pedal representation* of H. This form of duality is employed in linear programming and also in computing the convex hulls of finite point sets.

Consider E^d as embedded in P^d in the usual way. We observe that $[-1, n_1, n_2, \ldots, n_d]$, *the projective dual of H, is related to the projective coordinates of the inverted pedal of H by multiplication by the matrix*

$$J_{d+1} \quad = \quad \left(\begin{array}{c|c} -1 & 0 \\ \hline 0 & I_d \end{array} \right)$$

3.3 Polar Sets

Let K be a non-empty subset of E^d. The *polar set* of K is

$$K^* \equiv \{ \mathbf{x} \in E^d \ : \ \mathbf{x} \cdot \mathbf{y} \leq 1 \ \text{ for all } \ \mathbf{y} \in K \} \ .$$

For example:

- if K consists of a single point $\mathbf{p}$, and $\mathbf{p} \neq \mathcal{O}$, then K^* is the closed half-space $\{\mathbf{x} \ : \ \mathbf{p} \cdot \mathbf{x} \leq 1\}$;
- if K is a closed ball of radius r centered at $\mathcal{O}$, then K^* is the closed ball of radius $\frac{1}{r}$ centered at $\mathcal{O}$;
- if K is the unit cube bounded by $x_1 = \pm 1$, $x_2 = \pm 1$, and $x_3 = \pm 1$, then K^* is the octahedron with vertices $(\pm 1, 0, 0)$, $(0, \pm 1, 0)$, and $(0, 0, \pm 1)$.

The following general properties of polar sets are useful in our approach to the convex hull problem. Let K, K_1, K_2, and $K_\alpha (\alpha \in \mathcal{A})$ be nonempty sets. Then:

1. K^* is a closed convex set containing $\mathcal{O}$.
2. If $K_1 \subseteq K_2$ then $K_2^* \subseteq K_1^*$.
3. $(\cup_{\alpha \in \mathcal{A}} K_\alpha)^* = \cap_{\alpha \in \mathcal{A}} (K_\alpha)^*$. In words, the polar set of the union is the intersection of the polar sets.
4. If $\lambda > 0$, then $(\lambda K)^* = (1/\lambda) K^*$.

5. If K is compact and convex, and contains $\mathcal{O}$ as an interior point, then K^* is compact and contains $\mathcal{O}$ as an interior point.

6. If K is closed and convex, and contains $\mathcal{O}$, then $(K^*)^* = K$.

Observation 1 *Let $K \subset E^d$ be non-empty. Then $K^* \subseteq (\texttt{conv}\, K)^*$.*

proof:

Suppose $\mathbf{y} \in K^*$. Let $\mathbf{x} \in \texttt{conv}K$ be arbitrarily given. Then there exist $k \in N$, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k \in K$, and $0 \leq \lambda_1, \lambda_2, \ldots, \lambda_k \leq 1$ such that $\mathbf{x} = \sum_{i=1}^{k} \lambda_i \mathbf{x}_i$. Thus

$$\begin{aligned} \mathbf{y} \cdot \mathbf{x} &= \sum_{i=1}^{k} \lambda_i \mathbf{y} \cdot \mathbf{x}_i \\ &\leq \sum_{i=1}^{k} \lambda_i \\ &= 1. \end{aligned}$$

Hence $\mathbf{y} \in (\texttt{conv}\, K)^*$. □

On the other hand by applying (2) to the fact that $K \subseteq \texttt{conv}\, K$, we see that $K^* \supseteq \texttt{conv}\, K$. Thus we conclude that

Observation 2 *Let $K \subset E^d$ be non-empty. Then $K^* = (\texttt{conv}\, K)^*$ and $K^{**} = \texttt{conv}\, K$.*

3.4 Topologically Derived Sets of the Polar Set

3.4.1 Partitioning Hyperplanes w.r.t. a Set

Let $K \subset E^d$ be compact and non-empty. Let us consider the set of all hyperplanes $\mathcal{H}$ in E^d. They can be partitioned into three equivalence classes with respect to K as follows. Let $H = \{\mathbf{x} \,:\, f(\mathbf{x}) = 0\}$ be a given hyperplane.

1. if $f(\mathbf{x}) < 0$ for all $\mathbf{x} \in K$ or $f(\mathbf{x}) > 0$ for all $\mathbf{x} \in K$, then H is said to *strictly bound* K;

2. if f takes on a positive value for some $\mathbf{x} \in K$ and takes on a negative value for some $\mathbf{y} \in K$, then H is said to *cut* K;

3. Otherwise the sign of f does not change but does become zero for some choice of $\mathbf{x} \in K$ and it is said to *support* K.

We will denote the set of strictly bounding hyperplanes, the set of cutting hyperplanes, and the set of supporting hyperplanes of K by $\mathcal{H}_{\mathrm{b}}(K)$, $\mathcal{H}_{\mathrm{c}}(K)$, and $\mathcal{H}_{\mathrm{s}}(K)$, respectively.

It is straightforward to verify that $\mathtt{conv}\, K$ induces exactly the same partitioning of $\mathcal{H}$ as K does, i.e.,

$$\begin{aligned} \mathcal{H}_{\mathrm{b}}(\mathtt{conv}\, K) &= \mathcal{H}_{\mathrm{b}}(K) \\ \mathcal{H}_{\mathrm{c}}(\mathtt{conv}\, K) &= \mathcal{H}_{\mathrm{c}}(K) \\ \mathcal{H}_{\mathrm{s}}(\mathtt{conv}\, K) &= \mathcal{H}_{\mathrm{s}}(K) \end{aligned}$$

3.4.2 Topological Properties of the Partitioning

We now show that the pedal representations of the hyperplanes in each class is topologically derived from those of the other two.

Lemma 1 *Let $K \subset E^d - \{\mathcal{O}\}$ be a set containing at least two points. Then* $\mathtt{ped}\,(\mathcal{H}_{\mathrm{c}} K)$ *is open, in the relative topology of $E^d - \{\mathcal{O}\}$.*

proof:

To show that $\mathtt{ped}\,(\mathcal{H}_{\mathrm{c}} K)$ is open, consider an arbitrary $H \in \mathcal{H}_{\mathrm{c}}(K)$. We will construct an open neighborhood of $\mathtt{ped}\, H$ which is entirely contained in $\mathtt{ped}\,(\mathcal{H}_{\mathrm{c}} K)$. We leave out the technical details giving the precise formulas in the construction of an open ball and instead give an intuitive and pictorial account (Figure 3.1). Let $\mathbf{p}, \mathbf{q} \in K$ be such that $\mathbf{p}$ and $\mathbf{q}$ fall on different sides of H. Every hyperplane passing any point on the line segment $\overline{\mathbf{pq}} \subseteq K$ cuts K. It is straightforward to verify that their perpendicular foots form the shaded area in the figure. The point $\mathtt{ped}\, H$ is in the interior of that area and hence in the interior of $\mathtt{ped}\, \mathcal{H}_{\mathrm{c}} K$. $\square$

Lemma 2 *Let $K \subset E^d$ be a closed set such that $\mathcal{O} \in \mathtt{int}(\mathtt{conv}K)$. Then* $\mathtt{ped}(\mathcal{H}_\mathrm{b}K) \subseteq \mathtt{ext}\,\mathtt{ped}\,(\mathcal{H}_\mathrm{c}K)$*, in the relative topology of $E^d - \{\mathcal{O}\}$.*

proof:

If K does not have any strictly bounding hyperplane, then neither can it have any supporting hyperplane. Hence $\mathcal{H}_\mathrm{c}$ is the entirety of $E^d - \{\mathcal{O}\}$ and $\mathtt{ped}\,(\mathcal{H}_\mathrm{b}K)$ is vacuously contained in the exterior of $\mathtt{ped}\,(\mathcal{H}_\mathrm{c}K)$.

So suppose that $\mathcal{H}_\mathrm{b}K$ is non-empty and let $H = \{\mathbf{x} \,:\, f(\mathbf{x}) = 0\} \in \mathcal{H}_\mathrm{b}(K)$ be arbitrarily given and its sign chosen in such a way that $\mathbf{x} < 0$ for all $\mathbf{x} \in K$. By assumption $f(\mathcal{O}) < 0$. Let $\mathbf{p} \notin K$ be a point such that $f(\mathcal{O}) < f(\mathbf{p}) < 0$. (Such $\mathbf{p}$ always exists; otherwise K wouldn't be closed.) Also let $\mathbf{q}$ be a point such that $f(\mathbf{q}) > 0$. Repeating an argument similar to that in the previous proof, we conclude that there is an open neighborhood of $\mathtt{ped}\,H$ which is entirely in the exterior of $\mathtt{ped}\,(\mathcal{H}_\mathrm{c}K)$. □

Lemma 3 *Let $K \subset E^d$ be a closed set such that $\mathcal{O} \in \mathtt{int}(\mathtt{conv}K)$. Then* $\mathtt{ped}(\mathcal{H}_\mathrm{s}K) \subseteq \partial\mathtt{ped}\,(\mathcal{H}_\mathrm{c}K)$*.*

proof:

If K does not have any supporting hyperplane then the statement is vacuously true. So suppose that $\mathcal{H}_\mathrm{s}K$ is non-empty and let $H = \{\mathbf{x} \;:\; f(\mathbf{x}) = 0\} \in \mathcal{H}_\mathrm{s}(K)$ be arbitrarily given and its sign chosen in such a way that $\mathbf{x} \leq 0$ for all $\mathbf{x} \in K$. By assumption $f(\mathcal{O}) < 0$. Figure 3.2 shows that every open neighborhood of $\mathtt{ped}\,H$ must contain points in $\mathtt{ped}\,(\mathcal{H}_\mathrm{c}K)$ (those in the open line segment $\overline{\mathcal{O}\mathtt{ped}\,H}$), as well as points in the complement of $\mathtt{ped}(\mathcal{H}_\mathrm{c}K)$ (those in the closed ray beginning at $\mathtt{ped}\,H$ and pointing away from $\mathcal{O}$). □

Putting all these together, we conclude that if $K \subseteq E^d$ is closed with $\mathcal{O}$ in the interior of its convex hull, then $\mathtt{ped}\,(\mathcal{H}_\mathrm{c}K)$ is open and $\mathtt{ped}\,(\mathcal{H}_\mathrm{b}K)$ and $\mathtt{ped}\,(\mathcal{H}_\mathrm{s}K)$ *are* its exterior and boundary, respectively. This follows from the fact that the three sets of hyperplanes form a partition of $\mathcal{H}$; while the three topologically derived sets form a partition of $E^d - \{\mathcal{O}\}$, and hence the containment relationships in the last two lemmas are really equalities.

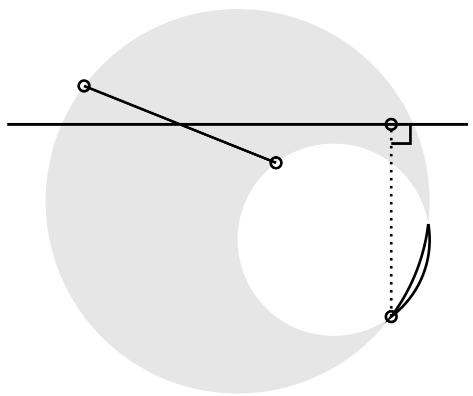

Figure 3.1: Shaded area represent loci of the perpendicular foots of the straight lines (hyperplanes in E^2) which necessarily cut K.

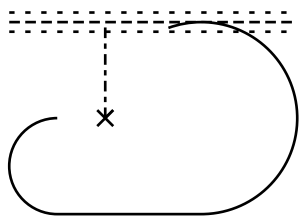

Figure 3.2: $\texttt{ped}\, H$ is on the boundary of the pedals of the cutting hyperplanes of K.

3.4.3 Topologically Derived Sets

Lemma 4 *Let $K \subseteq E^d$ be such that $\mathcal{O} \in \mathtt{intconv} K$. Then $K^* - \{\mathcal{O}\} = \mathtt{invped}((\mathcal{H}_{\mathrm{b}} K) \cup (\mathcal{H}_{\mathrm{s}} K))$.*

proof:

Let $\mathbf{p} \in E^d - \{\mathcal{O}\}$. Then

$$\begin{array}{lll} & \mathbf{p} \in K & \\ \Longleftrightarrow & \mathbf{p} \cdot \mathbf{x} \leq 1 \quad \text{for all} \quad \mathbf{x} \in K & \text{definition} \\ \Longleftrightarrow & H = \{\mathbf{x} \ : \ \mathbf{p} \cdot \mathbf{x} = 1\} \quad \text{bounds} \quad K & \text{since} \quad \mathcal{O} \in \mathtt{int}\,\mathtt{conv}\, K \\ \Longleftrightarrow & \mathbf{p} = \mathtt{inv}\,\mathtt{ped}\,((\mathcal{H}_{\mathrm{b}} K) \cup (\mathcal{H}_{\mathrm{s}} K)) & \text{inverted pedal} \end{array}$$

□

Earlier we have seen that the pedal representations of the three sets of hyperplanes induced by K form topologically derived sets of one another. This is still the case after the inversion map is applied to these sets since inversion is a homeomorphism of $E^d - \{\mathcal{O}\}$ onto itself. Now this lemma shows that the polar set of K is in fact the inverted image of $(\mathcal{H}_{\mathrm{b}} K) \cup (\mathcal{H}_{\mathrm{s}} K)$. We thus conclude with the following

Theorem 3.1 *Let $K \subset E^d$ be closed with $\mathcal{O} \in \mathtt{int}\,\mathtt{conv}\, K$. Then*

$$\begin{aligned} \mathtt{int}\,(K - \{\mathcal{O}\}) &= \mathtt{inv}\,\mathtt{ped}\,(\mathcal{H}_{\mathrm{b}} K) \\ \partial(K - \{\mathcal{O}\}) &= \mathtt{inv}\,\mathtt{ped}\,(\mathcal{H}_{\mathrm{s}} K) \\ \mathtt{ext}\,(K - \{\mathcal{O}\}) &= \mathtt{inv}\,\mathtt{ped}\,(\mathcal{H}_{\mathrm{c}} K) \ . \end{aligned}$$

3.5 Curvilinear Complexes

Linear Complexes and Polyhedra

Let $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_k$ be a linearly independent set of points in E^d, $k \leq d$. The collection of their convex combinations $s^k \equiv \{\sum_{i=0}^{k} t_i \mathbf{p}_i \ : \ \sum_{i=0}^{k} t_i = 1 \ \text{ and } \ t_i \geq$

0 for $0 = 0, 1, \ldots, k\}$ defines a *closed k-simplex*. Its relative interior is called an *open k-simplex*. When we use the term *k-simplex* by itself we shall mean open k-simplex. The simplex formed by a proper subset of $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_k$ is called a *boundary simplex* of s^k.

An *m-complex* is a collection of disjoint simplexes whose boundary simplexes are also in the collection and in which there is at least one m-simplex. The *k-skeleton* of an m-complex K is the set of all k-simplexes in K, $0 \leq k \leq m$. The point-set union of the simplexes in a complex K is called a *polyhedron* and denoted $|K|$. Correspondingly, K is called a *triangulation* of $|K|$.

Curvilinear Complexes

A *topological k-simplex* is a space homeomorphic to a k-simplex in a Euclidean space. *Topological complexes, topological polyhedra*, and *triangulation* and be defined in terms of their rectilinear counterparts in a natural way.

We are interested in topological complexes which are also embedded in a Euclidean space and whose simplexes are diffeomorphic, rather than just homeomorphic, to the corresponding rectilinear simplexes. We shall use the terms *curvilinear simplexes, curvilinear complexes*, and *curvilinear polyhedra* to refer to such objects and the related concepts.

Chapter 4

Duals of Curved Polyhedra

4.1 Object Model

Input objects we consider in this thesis will be curvilinear polyhedra with additional properties to be stated below. Being topological polyhedra, they are compact. On the other hand, they are not necessarily connected, nor are they necessarily manifolds, although each of their triangulating simplexes is indeed a simply connected manifold without boundary.

For practical considerations, we make some additional amendments on our desired curvilinear polyhedron K regarding how they are expected in the input into the algorithms.

1. All diffeomorphisms between the curvilinear simplexes and their defining rectilinear counterparts are restrictions of some algebraic functions to the simplexes.

2. Instead of requiring that the input be a description of the triangulating simplexes of the polyhedron, we will accept descriptions by *faces*. A k-*face* is the union of some adjacent k-simplexes $s_1^k, s_2^k, \ldots, s_m^k$ of K plus some of their boundary simplexes such that

   - the entire union is homeomorphic to an open k-ball,
   - the homeomorphisms of all faces participating in the union are the restrictions of the same algebraic function f,
   - this algebraic function has a full rank Jacobian (k) in the entire union, and

- the union can be described as the intersection of algebraic halfspaces with f.

3. Each k-face has at least $(k+1)$ *subfaces* of dimension $k-1$. A subface of the k-face is a face of K composed of the boundary simplexes of some of $s_1^k, s_2^k, \ldots, s_m^k$ which do not vanish as a glue.

More specifically, each k-face φ^k is expected to be described as

$$\varphi^k = \{\mathbf{x} \ : \ f_1(\mathbf{x}) = f_2(\mathbf{x}) = \cdots = f_{d-k}(\mathbf{x}) = 0,$$
$$g_1(\mathbf{x}) < 0, g_2(\mathbf{x}) < 0, \ldots, g_m(\mathbf{x}) < 0\}$$

where $m > k$ and where the Jacobian of $f_1, f_2, \ldots, f_{d-k}$ has a full rank $(d-k)$ everywhere in φ^k. If K satisfies the above requirements, we shall call it a *curved polyhedron*. For computational convenience, we also assume that one sample point on each face is given along with its algebraic description even when the algebraic description alone is sufficient to single out just one simply connected component.

We will refer to the algebraic variety defined by $f_1, f_2, \ldots, f_{d-k}$ as the *host variety* of φ^k. A 0-face is also referred to as a *vertex*, a 1-face as an *edge*, and a $(d-1)$-face as a *facet*.

4.2 Duals of Faces

The polar set duality is directly related to the convex hull of a set K in that one obtains `conv` K by taking its polar set twice. Its definition, however, does not provide an operational recipe for actually writing a program to compute the convex hull of, say, an industrial part, which we model by a curved polyhedron. The pedal and projective dual of *surfaces*, on the other hand, are algebraic varieties whose equations can indeed be algorithmically computed. Yet we will have to answer the following questions before we can employ these ideas in solving the convex hull problem.

- What is a useful definition for the dual of a surface patch as opposed to that of an entire algebraic surface?

- What is a useful definition for the dual of a face of a lower dimension?
- What is a useful definition for the dual of a curved polyhedron once we have definitions for its faces?

Here the usefulness of a definition depends on how we will eventually be able to relate the "dual" of a curved polyhedron to its polar set and construct the latter from the former.

4.2.1 Duals of Vertices and Facets

If $\varphi^0 = \{\mathbf{p}\}$ is a 0-dimensional face on a curved polyhedron K, we define the *inverted pedal*, or simply *dual* of φ^0, to be

$$\tilde{\varphi}^0 = \{\mathbf{x} \ : \ \mathbf{x} \cdot \mathbf{p} = 1\}$$

This is exactly the dual hyperplane in the linear programming literature, and coincides with the boundary hyperplane of the polar set of φ^0.

If φ^{d-1} is a $(d-1)$-dimensional face on a curved polyhedron $K \subset E^d$ described by

$$\varphi^{d-1} = \{\mathbf{x} \ : \ f(\mathbf{x}) = 0, g_1(\mathbf{x}) < 0, g_2(\mathbf{x}) < 0, \ldots, g_m(\mathbf{x}) < 0\} \quad ,$$

we define its dual to be

$$\tilde{\varphi}^{d-1} = \{\texttt{inv ped}\, H_{\mathbf{X}} \ : \ \mathbf{x} \in \varphi\}$$

where $H_{\mathbf{X}}$ is the tangent hyperplane of the host variety of φ at $\mathbf{x}$. Note that $H_{\mathbf{X}}$ is well-defined because of the assumption about the rank of ∇f. Neglecting the boundary constraints imposed by g_i's, this is just the inversion of the pedal hypersurface of the host variety of φ (to the extent that the host variety is a hypersurface).

Figure 4.1 shows a cycloid and its dual. The dual of the cycloid itself (the *open* 1-face) is the curve with two asymptotes which begins at $\mathbf{u}_1$, goes through two improper points, comes back from the third quadrant, and finally ends at $\mathbf{u}_2$. The two end points of the cycloid, being 0-faces themselves, dualize to the two straight lines $\tilde{\mathbf{v}}_1$ and $\tilde{\mathbf{v}}_2$. Note that they are tangent to the dual of the curve at the end points $\mathbf{u}_1$ and $\mathbf{u}_2$, respectively. Figure 4.2 shows an s-shaped figure. The tangent to the curve at one of the end points $\mathbf{v}_1$ happen to pass the origin. In the dual space this is reflected by an asymptotic branch of the dual curve having approaching an improper point from only one side, and $\tilde{\mathbf{v}}_1$ being its asymptote.

4.2.2 Duals of Faces of General Dimensions

In general, let

$$\varphi^k = \{\mathbf{x} \;:\; f_1(\mathbf{x}) = f_2(\mathbf{x}) = \cdots = f_{d-k}(\mathbf{x}) = 0,$$
$$g_1(\mathbf{x}) < 0, g_2(\mathbf{x}) < 0, \ldots, g_m(\mathbf{x}) < 0\}.$$

where $0 < k < d-1$. Its dual is defined to be the set

$$\tilde{\varphi}^k = \{\texttt{inv}\,\texttt{ped}\,H \;:\; H \text{ has a contact of order 1 or higher at } \mathbf{x} \text{ with } \varphi \text{ for some } \mathbf{x} \in \varphi\}.$$

This definition subsumes the facet case where $k = d-1$. If we consider every hyperplane passing a vertex to have a contact of order 1 with the vertex, then the vertex case where $k = 0$ may also be subsumed. In any case we observe that a supporting hyperplane of a face φ^k necessarily have a contact of order at least one with φ^k because of the smoothness assumption for $k = 1, 2, \ldots, d-1$ and because of the convention for $k = 0$.

4.3 The Cell Theorem

Let K be a curved polyhedron. The *inverted pedal*, or simply *dual* of K is defined to be

$$\tilde{K} = \{\tilde{\varphi} \;:\; \varphi \in K\}.$$

The point set union of $\tilde{K}$ consists of the inversions of the pedals of those hyperplanes which have a contact of order 1 or higher with at least one of K's faces.

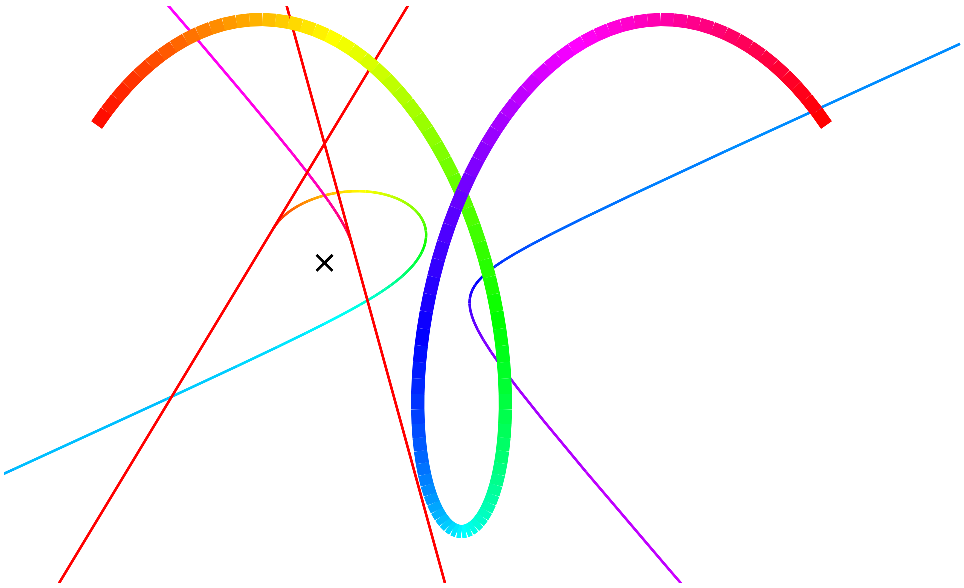

Figure 4.1: A cycloid and its dual It can be decomposed into three smooth open cells—the two end points and the rest of the curve. The two straight lines are images of the vertices. The dual of the rest is the a curve with two asymptotes, which correspond to the two supporting lines from the origin to the curve.

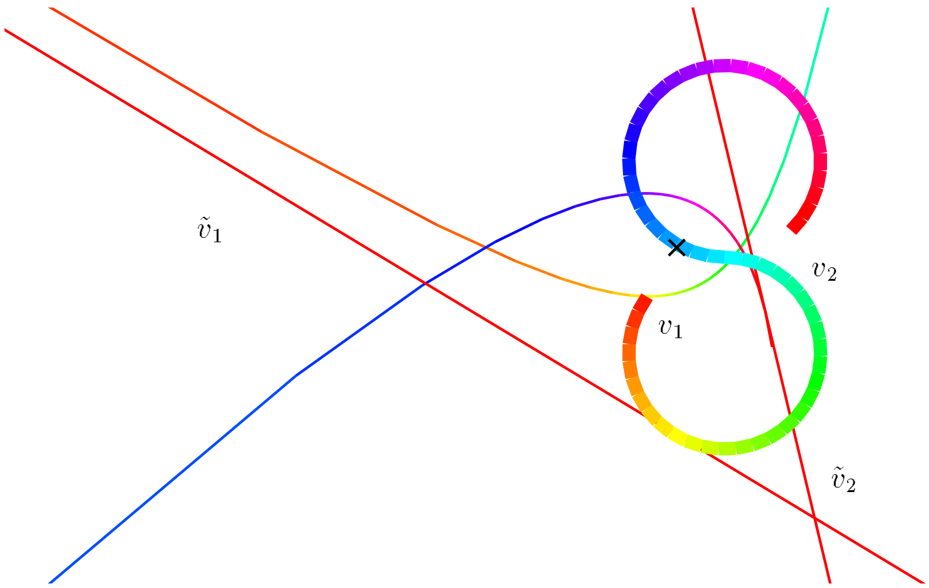

Figure 4.2: An S-shaped figure and its dual.

On the one hand, each of such hyperplanes has a non-empty intersection with K, namely a set containing at least the point of contact. They then therefore necessarily belong to $\mathcal{H}_c K$ or $\mathcal{H}_s K$ and can never strictly bound K. On the other hand, let us consider an arbitrarily given hyperplane $H \in \mathcal{H}_s K$. Let $\mathbf{x}$ be *a* point of support of K by H and φ^k be the k-face which $\mathbf{x}$ belongs to. If $k = 0$ then by definition all hyperplanes containing $\mathbf{x}$, including H, belong to $\tilde{K}$. If $0 < k < d$ then our object model ensures that φ has a k-dimensional tangent space $H^k_{\mathbf{X}}$ at $\mathbf{x}$. Moreover, $\mathbf{x}$ is in the relative interior of φ. Thus H must contain $H^k_{\mathbf{X}}$, or equivalently H must have a contact of order at least 1 with φ at $\mathbf{x}$, or else it would be a cutting hyperplane. Therefore in this case H also belongs to $\tilde{K}$. Finally, k cannot be d since any common point of H and φ would be an interior point and cannot be a point of support. We have shown that:

$$\mathtt{inv\,ped}\,(\mathcal{H}_s K) \subseteq \tilde{K} \subseteq \mathtt{inv\,ped}\,(\mathcal{H}_s K) \cup \mathtt{inv\,ped}\,(\mathcal{H}_c K).$$

Combined with the alternative description of the topologically derived set of K^* (Section 3.4.2), we conclude that

Theorem 4.1 *Let $K \subset E^d$ be a curved polyhedron. Then the polar set of K is the closed d-cell containing $\mathcal{O}$ in its interior in the arrangement of all hypersurface patches in $\tilde{K}$.*

In particular, $\partial(K^*)$ coincides with the boundary of this special cell.

4.4 Computations and General Properties of Dual

In this section we give formulæ, properties, and examples of duals, for later reference. We are mainly interested in computing the duals of the host algebraic varieties since the extent of the original faces may be directly modified to serve as the extent of the duals (see below). Thus in the following we will sometimes refer to the "dual of an algebraic variety φ," which really means the host of the dual of a face hosted by the algebraic variety φ. Many of the properties follow from the definition. Some others are direct corollaries of their counterparts in projective geometry, linear programming, polar set, or pedal curves/surfaces, preserved by the transformations involved. For example, the cusp-inflection duality is a property derived from the corresponding property of a pedal curve, and preserved by inversion, which is a homeomorphism from $E^d - \{\mathcal{O}\}$ onto itself and is continuously differentiable up to any order. We will give proofs only in the remaining cases.

4.4.1 General Properties

- Common supporting hyperplanes of two faces dualize to the intersection of their duals.
- The intersection of some finitely many hyperplanes dualizes to the affine hull of the duals of these hyperplanes.
- Let φ be a k-dimensional linear subspace of Ed whose pedal is $\mathbf{n}$. Then its dual $\tilde{\varphi}$ is the unique $(d-k-1)$-dimensional linear subspace whose pedal is $\texttt{inv}\,\mathbf{n}$ and which is orthogonal to the spanning vectors of φ. In particular, vertices dualize to hyperplane and in E^3 straight line edges dualize to straight lines.
- In E^2, ordinary cusps and ordinary inflection points dualize to each other. (Figure 4.3)
- In E^3, a parabolic curve on a surface dualizes to a cuspidal edge on the dual surface. (Figures 4.4 and 4.5, equation taken from [32].)
- The envelope of a k-parameter family of hyperplanes dualizes to a k-face. In particular, let φ be a general developable surface in E^3 and γ its edge of regression. Then $\tilde{\gamma}$ is a developable surface and its edge of regression is $\tilde{\varphi}$.
- In E^3, a ruled surface dualizes to a ruled surface since tangents to a ruling form a family of planes sharing a straight line (namely the ruling), which dualize to a straight line.

4.4.2 Implicit Faces

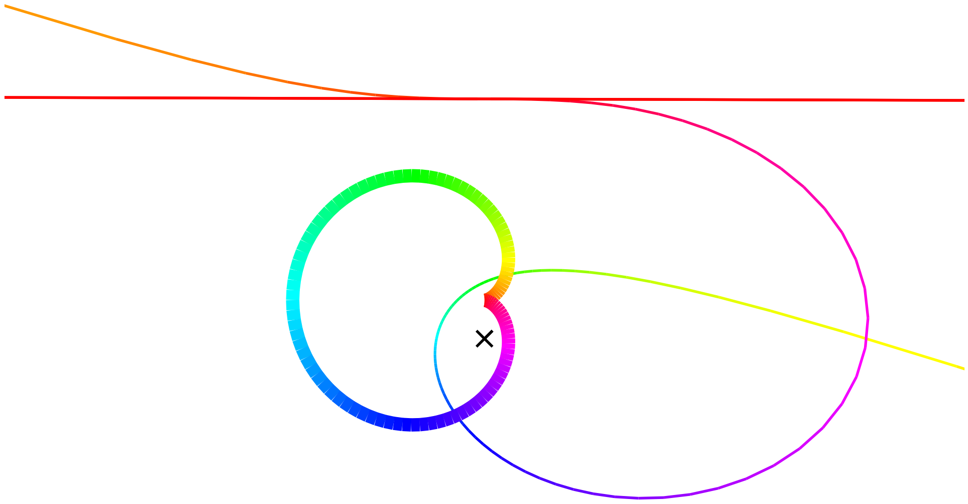

Figure 4.3: A cardioid and its dual. It need be decomposed into at least two smooth open cells—the cusp the the rest of the curve. The horizontal line is the dual of the cusp. The dual curve begins and ends at the inflection point, at which the horizontal line is the tangent.

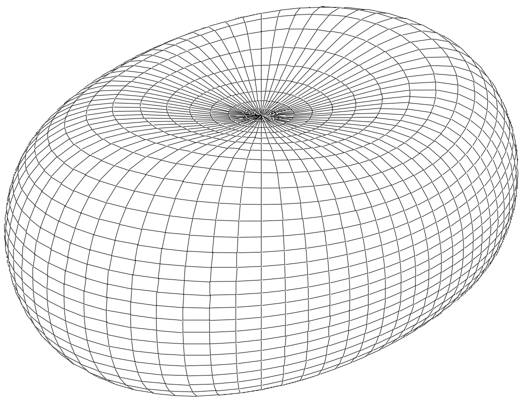

Figure 4.4: A dimpled surface—the zero set of $(4x^2 + 3y^2)^2 - 4x^2 - 5y^2 + 4z^2 - 1$.

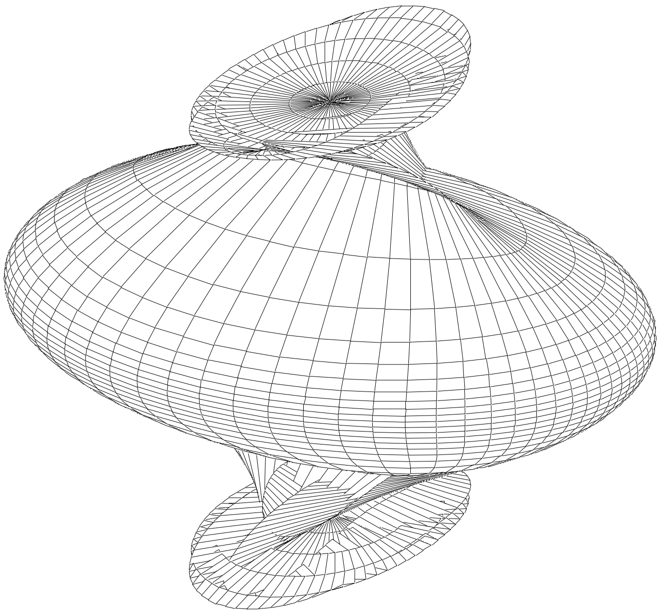

Figure 4.5: Dual of the dimpled surface. The two "umbrellas" turned inside out correspond to tangent planes to points inside the two dimples. The apparent sharp cuspidal edges of the umbrellas are duals of two parabolic curves inside the dimples.

Lemma 5 *Let φ be a k-face whose host variety is defined by the set of $d-k$ equations*

$$f_i(\mathbf{x}) = 0, \quad i = 1, 2, \ldots, d-k$$

and let

$$\mathbf{d}_i(\mathbf{y}) = \frac{\nabla f_i(\mathbf{y})}{\nabla f_i(\mathbf{y}) \cdot \mathbf{y}}, \quad i = 1, 2, \ldots d-k.$$

Let $A(\mathbf{y})$ be the $(d-k-1) \times d$ matrix whose i-th row vector is $[\mathbf{d}_i(\mathbf{y}) - \mathbf{d}_{d-k}(\mathbf{y})]^T$ and let $\mathbf{c}_i(\mathbf{y})$ for $i = 1, 2, \ldots, k+1$ be a basis for its null space. Then a set of implicit equations in the variables $\mathbf{x}$ representing the host variety of $\tilde{\varphi}$ is given by eliminating $\mathbf{y}$ from

$$\begin{aligned} (\mathbf{x} - \mathbf{d}_{n-k}(\mathbf{y})) \cdot \mathbf{c}_i(\mathbf{y}) = 0, &\qquad i = 1, 2, \ldots, k+1 \\ f_i(\mathbf{y}) = 0, &\qquad i = 1, 2, \ldots, d-k. \end{aligned}$$

For example, let φ be the surface in E^3

$$\varphi = \{(x_1, x_2, x_3) \; : \; f(x_1, x_2, x_3) = 0\}.$$

Then its dual is

$$\tilde{\varphi} = \left\{ \frac{\nabla f(x_1, x_2, x_3)}{\nabla f(x_1, x_2, x_3) \cdot (x_1, x_2, x_3)} \; : \; f(x, y, z) = 0 \right\}.$$

The extent of the face φ can be determined as follows. Let $\mathbf{x}$ be an arbitrarily given point known to lie on the host of the dual of the face φ. Then the dual hyperplane of $\mathbf{x}$ is tangent to the host of φ at some point $\mathbf{y}$. Find all such tangent points and conclude that $\mathbf{x} \in \tilde{\varphi}$ if and only if any such point of tangency lies within the extent defining φ on its host surface. In a later chapter we discuss in more details how this procedure can be carried out with less numerical problems.

4.4.3 Parametric Faces

Formulæ for parametric faces can be similarly derived from the definitions. The extent of the the dual faces is especially easy in this case since the dual is also a parametric face. All of the k parameters in the primal k-face are also parameters for the dual and retain their extent. The additional $(d-k)$ parameters of the dual extend over the entirety of R^{d-k}.

1. Let $\gamma \subset E^2$ be the edge $\{\mathbf{r}(t) \ : \ t \in I\}$ where I is an open interval. Then $\tilde{\gamma}$ is the parametric curve:

$$\left\{ \frac{|\mathbf{r}'|^2\mathbf{r} - (\mathbf{r}\cdot\mathbf{r}')\mathbf{r}'}{|\mathbf{r}|^2|\mathbf{r}'|^2 - (\mathbf{r}\cdot\mathbf{r}')^2}(t) \ : \ t \in I \right\}.$$

2. Let φ be a parameterized surface in E^3 given by $\{\mathbf{x}(u,v) \ : \ (u,v) \in D\}$ where D is open and simply connected in E^2.

$$\tilde{\varphi} = \left\{ \frac{\mathbf{x}_u \times \mathbf{x}_v}{|\mathbf{x}_u \mathbf{x}_v \mathbf{x}|} \ : \ (u,v) \in D \right\}.$$

3. Let γ be a parameterized curve in E^3 given by $\{\mathbf{x}(t) \ : \ t \in D\}$ where I is an open interval. Then $\tilde{\gamma}$ is the developable surface

$$\left\{ \frac{\mathbf{x}(t)}{|\mathbf{x}(t) \times \mathbf{x}'(t)|} + s\mathbf{x}(t) \times \mathbf{x}'(t) \ : \ t \in D \ \text{ and } \ s \in R \right\}.$$

Chapter 5

Quadratic Hypersurface Patches

In the previous chapter we have seen that computing the dual of a hypersurface in general involves variable elimination. Such algebraic computations are expensive and often requires the use of exact arithmetic. Since many industrial parts in reality are modeled by low degree surfaces, we would like to investigate if there are less expensive alternatives in these special cases. Fortunately it is the case with quadratic surface patches bounded by planar conic sections, for which the dual constructions can be largely reduced to linear algebra.

Although our main focus in this chapter is on quadratic hypersurfaces, the results we obtain are useful in the more general setting where the hosts of the faces are of higher degrees. In the last section we will be able to show that in a practical sense the severity of degeneracy of the duals is determined completely by investigating the partial derivatives of the original algebraic variety.

5.1 Non-Degenerate Hosts and the Extent of the Dual

Let $H = \{\mathbf{x} \in E^d \ : \ \mathbf{n}\cdot\mathbf{x} = 1\}$ be a hyperplane in E^d where $\mathbf{n} = (n_1, n_2, \ldots, n_d) \in E^d$ was observed to be the inversion of its pedal. We observed in an earlier chapter that if we write $\mathbf{n} = [1, n_1, n_2, \ldots, n_d]$ as a projective point in P^d, then the projective dual $\mathbf{n}'$ of H is related to $\mathbf{n}$ by

$$\mathbf{n}' = J_{d+1}\mathbf{n}$$

where J_{d+1} is the I_{d+1} with the first diagonal entry replaced by -1. Now consider a non-degenerate quadratic hypersurface

$$\{\mathbf{x} \in P^d \ : \ \mathbf{x}^T Q \mathbf{x} = 0\}$$

where $Q = [q_{ij}]$, $0 \leq i, j, \leq d$ has non-vanishing determinant. It is straightforward to follow the proofs of the theorems in projective geometry and conclude that $\tilde{Q}$ the inverted pedal of Q is the proper portion of

$$\{\mathbf{x} \in P^d \ : \ \mathbf{x}^T J_{d+1} Q^{-1} J_{d+1} \mathbf{x} = 0\}.$$

In our notation, we will use $JQ^{-1}J$ to also denote the dual surface in a set context.

The extent of the dual face on the dual host surface can be determined implicitly as in the case for general dual surfaces. In the quadratic case, it can also be determined directly in the dual space. Let H be a hyperplane bounding the face φ whose host is Q. The portion of the boundary of $\tilde{\varphi}$ as determined by H consists of the duals of the tangent hyperplanes taken along the cut. These tangent hyperplanes all intersect at one point, namely $\mathbf{p}$ the pole of H with respect to Q. Thus their duals are just the intersection of $\tilde{\mathbf{p}}$ with $\tilde{\varphi}$.

5.2 Generalized Projective Duality

The projective duality

$$\mathbf{x}^T Q \mathbf{x} = 0 \iff \mathbf{n}^T Q^{-1} \mathbf{n} = 0$$

of quadratic hypersurfaces was derived based on the hyperplane-point duality:

$$\{\mathbf{x} \ : \ \mathbf{n} \cdot \mathbf{x} = 0\} \iff \mathbf{n}^T.$$

When we replaced the above hyperplane-point duality by the pedal duality:

$$\{\mathbf{x} \ : \ \mathbf{n}\mathbf{x} = 0\} \iff (\mathbf{n} J_{d+1})^T$$

we obtain the conclusion in the previous subsection. In general if we begin with a hyperplane-point duality defined to be

$$\{\mathbf{x} \,:\, \mathbf{n}\mathbf{x} = 0\} \iff R\mathbf{n}^T$$

where R is a non-singular real symmetric $(d+1)$ by $(d+1)$ matrix, then we obtain the duality

$$\mathbf{x}^T Q \mathbf{x} = 0 \iff \mathbf{n}^T (RQR)^{-1} \mathbf{n} = 0. \tag{5.1}$$

in like manner. In this case a linear bundle $B_{m,d+1}$ dualizes to the linear subspace RB^T.

It can be interpreted as dualizing the points, hyperplanes, and non-degenerate quadratic hypersurfaces with respect to the fixed non-degenerate quadratic hypersurface R. The pedal dual is a special case when $R = J_{d+1}$, the unit hypersphere. A second interpretation is that we are still taking the ordinary projective dual, except that we are carrying out all the computations in a new coordinate system in which the imaginary quadratic hypersurface

$$\{[\xi_0, \xi_1, \ldots, \xi_d] \,:\, \xi_0^2 + \xi_1^2 + \cdots + \xi_d^2 = 0\}$$

takes on the new form

$$\{[\xi_0', \xi_1', \ldots, \xi_d'] \,:\, \sum_{i,j=0}^{d} r_{ij} \xi_i' \xi_j' = 0\}.$$

For we can let $\mathbf{x} = R\mathbf{y}$ and see that the duality (5.1) taking place in the old coordinate system translates to the "standard" dual in the new coordinate system expressed in $\mathbf{y}$.

5.3 Degenerate Hosts

When Q has a vanishing determinant, the projective dual is undefined and the tangential dual need be computed directly in this case.

Let Q be a degenerate quadratic hypersurface of rank $k < d+1$. Write Q as $U\Lambda U^T$ where $\mathbf{u}_{*j}$ $j = 0, 1, \ldots, d$ the column vectors of U are eigenvectors of Q and where Λ is the diagonal matrix composed of Q's eigenvalues so arranged that $\lambda_i \neq 0$ for $i = 0, 1, \ldots, k$ and $\lambda_i = 0$ for $i = k+1, \ldots, d$. Now consider the non-degenerate quadratic hypersurface $Q' = U\Lambda' U^T$ where Λ' is diagonal with $\lambda'_i = \lambda_i$ for $i = 0, 1, \ldots, k$ and $\lambda'_i = 1$ for $i = k+1, \ldots, d$.

Suppose first that an arbitrarily given hyperplane $Fvecn$ is tangent to Q. Then $\mathbf{n}Q\mathbf{x}_0 = 0$ for some $\mathbf{x}_0 \in Q$. Perform a coordinate transformation $\mathbf{x} = U\mathbf{y}$ and consider the new coordinate $\mathbf{y}_0 = U^T\mathbf{x}_0$ of $\mathbf{x}_0$. In the new coordinate system, the quadratic hypersurface takes on the form $\mathbf{y}^T\Lambda\mathbf{y} = 0$ and the given tangent hyperplane is $\mathbf{n}U$. It is straightforward to verify that the tangent hyperplane of Λ (that is, of Q in the old coordinate system) at each $\mathbf{y}$ whose first $k+1$ coordinates are the same as $\mathbf{y}_0$, is also $\mathbf{n}U$. Necessarily the last $d-k$ coordinates of $\mathbf{n}U$ vanish. This implies that $\mathbf{n}U$ is tangent to Λ' (or, to Q' in the old coordinate system) as well as passes $\mathbf{e}_i$ for $i = k+1, \ldots, d+1$ where $\mathbf{e}_i$ has all but one non-zero entry at the i-th position.

Conversely suppose we are given a hyperplane $\mathbf{n}'$ which is tangent to Q', and the last $d-k$ of whose coordinates $\mathbf{n}'U$ in the new coordinate system vanish. Let $\mathbf{x}'_0$ be the point of tangency on Q', i.e., $\mathbf{n}'Q'\mathbf{x}'_0 = 0$. Then it is not difficult to find a point $\mathbf{x}_0$ such that $\mathbf{n}'$ is tangent to Q at $\mathbf{x}_0$. Specifically let $\mathbf{x}_0$ be the vector whose coordinates $U^T\mathbf{x}_0$ in the new coordinate system are the same as $U^T\mathbf{x}'_0$ except with the last $d-k$ entries replaced by 0's.

Thus the tangent hyperplanes of Λ can be characterized as the those of Q' which also pass $\mathbf{u}_{*,k+1}, \ldots, \mathbf{u}_{*,d}$. The inverted pedal of Q is therefore the intersection of $\tilde{\mathbf{u}}_{*,k+1}, \ldots, \tilde{\mathbf{u}}_{*,d}$, and $\tilde{Q}'$. Since Q' is non-singular, $\tilde{Q}' = J_{d+1}(Q')^{-1}J_{d+1}$. The $\mathbf{u}_{*j}$'s are linearly independent and dualize to $\mathbf{u}_{*j}^T J_{d+1}$. Hence the intersection of these hyperplanes is a k-dimensional linear subspace $\pi \subset P^d$. Furthermore, π is not tangent to $\tilde{Q}'$. because $\mathbf{u}_{*,k+1}, \ldots, \mathbf{u}_{*,d} \notin Q'$. Thus $\tilde{Q}$, or equivalently the intersection of π with $\tilde{Q}'$, is a quadratic hypersurface of P^k embedded in P^d. In other words, the dual of a rank k quadratic hypersurface is $(k-2)$-dimensional.

5.4 Lower Dimensional Hosts

Let φ be the host of a $(d-k-1)$-dimensional face $(1 \le k < d)$ of a (degenerate or non-degenerate) quadratic hypersurface patch whose host is Q. Let

$$H_j = \{[\xi_0, \xi_1, \ldots, \xi_d] \;:\; \sum_{i=0}^{d} n_{ij}\xi_i = 0\} \quad j = 0, 1, \ldots, k-1$$

be a set of linearly independent hyperplanes whose intersection with Q defines φ. Let N be the $(d+1) \times k$ matrix whose columns are the vectors $\mathbf{n}_{*j}$, which after transposing represents the linear bundle spanned by these hyperplanes according to our convention. Perform QR factorization on N and write it as

$$N \;=\; (U_1 U_2) \begin{pmatrix} R \\ O \end{pmatrix}$$

where

- The dimensions of $U_1, U_2, R,$ and the zero matrix O are $(d+1) \times k, (d+1) \times (d+1-k), k \times k,$ and $(d+1-k) \times (d+1-k)$, respectively;
- $U = (U_1 U_2)$ is orthogonal;
- R is upper triangular and non-singular.

Write Q as

$$Q \;=\; (U_1 U_2) \begin{pmatrix} A & B \\ B^T & C \end{pmatrix} \begin{pmatrix} U_1^T \\ U_2^T \end{pmatrix}$$

where A of dimension $k \times k$ and C of dimension $(d-k+1) \times (d-k+1)$ are real symmetric. Then it is easy to verify that the following is another characterization of φ equivalent to the original one:

$$\{\mathbf{x} \;:\; U_1^T \mathbf{x} = 0 \;\text{ and }\; \mathbf{x}^T U_2 C U_2^T \mathbf{x} = 0\}.$$

Denote the degenerate quadratic hypersurface $U_2 C U_2^T$ of rank at most $d-k+1$ by Q'.

Theorem 5.1 *Let φ, U_2, and C be defined as above. If C is non-singular, then $\tilde{\varphi} = J_{d+1,d+1} U_2 C^{-1} U_2^T J_{d+1,d+1}$ is a quadratic hypersurface of rank $d-k+1$.*

proof:

We will prove a slightly different result in order to keep the notations manageable. In this proof only we modify the definition of tangential dual to be taken with respect to the imaginary quadratic hypersurface represented by the identity matrix, namely the ordinary projective dual, instead of with respect to the unit sphere $J_{d+1,d+1}$. Thus let $S = U_2 C^{-1} U_2^T$ be the quadratic hypersurface which we desire to show to be $\tilde{\varphi}$. Once we identify S with $\tilde{\varphi}$, we may apply the interpretations in a previous section by actually taking the dual with respect to $J_{d+1,d+1}$ and obtain the exact statement in the theorem.

We first show that $\tilde{\varphi} \subseteq S$. Let $\mathbf{x}_0 \in \varphi$ be arbitrarily chosen and fixed. The tangent hyperplanes of φ at $\mathbf{x}_0$ form a linear bundle of dimension k whose carrier (kernel) is the intersection of the linear subspace N (or equivalently U_1 since as matrices their transposes have the same null space) with the hyperplane $\mathbf{x}_0^T U_2 C U_2^T$. Thus in the dual space these tangent hyperplanes form the linear subspace spanned by the column vectors of U_1 and the vector $U_2 C U_2^T \mathbf{x}_0$. For each arbitrary k-vector $\mathbf{w}$ and each arbitrary scalar α, let $\mathbf{y} = U_1 \mathbf{w} + \alpha U_2 C U_2^T \mathbf{x}_0$. It is easy to verify that

$$\mathbf{y}^T U_2 C^{-1} U_2^T \mathbf{y} \quad = \quad 0$$

and hence conclude that $\tilde{\varphi} \subseteq S$.

Conversely, suppose $\mathbf{y}_0 \in S$. We will consider

$$L = \{\alpha \mathbf{y}_0 + U_1 \mathbf{v} \ : \ \alpha \in R \ \text{ and } \ \mathbf{v} \in R^k\}.$$

in the dual space all at once. It can be verified that $L \subset S$ and is a linear bundle of dimension k in the primal space. Using the column vectors of U_1 and U_2 as a basis, it is easy to find the carrier of L in the primal space to be

$$\{U_2 \mathbf{z} \ : \ \mathbf{y}_0^T U_2^T \mathbf{z} = 0\}.$$

It can also be verified that $\mathbf{z}_0 = C^{-1}U_2^T\mathbf{y}_0$ is a valid choice of $\mathbf{z}$ in the above. Furthermore the vector $\mathbf{x}_0 = U_2\mathbf{z}_0$ lies in Q'. The tangent hyperplanes to Q' at $\mathbf{x}_0$ dualize to the span of the vector $U_2CU_2^T\mathbf{x}_0$ and of the column vectors U_1. Substituting the formula for $\mathbf{x}_0$ and $\mathbf{z}_0$ into the linear combinations, we see that this linear bundle of dimension k is

$$L' = \{\alpha U_2U_2^T\mathbf{y}_0 + U_1\mathbf{w}' \ : \ \alpha' \in R \ \text{ and } \ \mathbf{w}' \in R^k\}\}.$$

We would like to show that $L = L'$. The elements of L for which $\alpha = 0$ coincide with the elements of L' for which $\alpha' = 0$. For the other elements in both sets we scale so that $\alpha = \alpha' = 1$. It now suffices to show that

$$\mathbf{y}_0 - U_2U_2^T\mathbf{y}_0 \quad \in \quad \{U_1\mathbf{w} \ : \ \mathbf{w} \in R^k\}.$$

But the span of U_1 is just the null space of U_2 and it is easily verified that the above claim is indeed true. $\square$

Finally we mention that if C is singular, we may diagonalize C as $V\Lambda V^T$ where the first m diagonal entries vanish, $0 < m < d-k+1$. Let Λ' be the same as Λ with the vanishing diagonal elements replaced by 1's. If we write Q as

$$(U_1U_2)\begin{pmatrix} I & 0 \\ 0 & V \end{pmatrix}\begin{pmatrix} 0 & 0 \\ 0 & \Lambda \end{pmatrix}\begin{pmatrix} I & 0 \\ 0 & V^T \end{pmatrix}\begin{pmatrix} U_1^T \\ U_2^T \end{pmatrix},$$

then it is not difficult to see that a combination of the arguments in the degenerate host and lower dimensional host leads to the conclusion that $\tilde{\varphi}$ is the intersection of $JU_2V(\Lambda')^{-1}V^TU_2^TJ$ with the hyperplanes represented by the first m rows of $V^TU_2^T$. For $d = 3$ the only case in which $k > 0$ and $m > 0$ corresponds to straight lines and points. Hence we may simply use the linear duals instead and will not have to invoke this result in practice.

5.5 Degeneracies and Singularities

In this section we study the singularities brought about by dualizing a smooth hypersurface of higher degrees. It turns out to be more convenient to work in the Euclidean space than in the projective space.

We first give a formula for computing the dual after a translation of coordinate systems. The duals are to be taken with respect to the unit sphere in the original coordinate system, however. Let $\mathbf{p} \in E^d$ be the origin of the new coordinate system expressed in the old coordinates. Then the duality

$$\{(x_1, x_2, \ldots, x_d)^T \; : \; \sum_{i=1}^{d} n_i x_i = 1\} \iff (n_1, n_2, \ldots, n_d)^T$$

in old coordinates translates to

$$\{(X_1, X_2, \ldots, X_d)^T \; : \; \sum_{i=1}^{d} n_i(X_i + p_i) = 1\} \iff (n_1 - p_1, n_2 - p_2, \ldots, n_d - p_d)^T\}$$

in new coordinates. Now suppose that we are given $H = \{(X_1, X_2, \ldots, X_d)^T \; : \; \sum_{i=1}^{d} n_i X_i = s\}$ in the new coordinate system to begin with, assuming that H does not contain the old origin. Let $\lambda = \frac{1}{s + \sum_{i=1}^{d} n_i p_i}$. Then the dual is $\lambda \mathbf{n} - \mathbf{p}$ in the new coordinate system.

Given an algebraic hypersurface φ and a regular point $\mathbf{p}$ in φ, we now ask under what conditions does the point dualize to a singular point, and how bad the singularity is when it does so. More precisely we would like to know the rank of the Jacobian matrix of the dual with respect to any regular parameterization of φ nearby the given point $\mathbf{p}$. To simplify the computations, we will show the derivation for $d = 3$, although the idea is the same in higher dimensions.

First we rotate the coordinate axes so that the tangent plane at $\mathbf{p}$ becomes parallel to the $x_1 x_2$-plane. Obviously the geometry of neither φ nor the dual is not affected and thus without loss of generality we will assume that this is the coordinate system initially given to us. Next we translate the origin to $\mathbf{p}$. Nearby the new origin (the given point), the surface can be written as $(x_1, x_2, f(x_1, x_2))^T$

for some (not necessarily algebraic) function f. The tangent plane corresponding to the parameters (x_1, x_2) is

$$\{(y_1, y_2, y_3) \; : \; (y_1 - x_1)f_1 + (y_2 - x_2)f_2 = y3 - x3\},$$

where $f_i = \frac{\partial}{\partial x_i} f$ are evaluated at the corresponding parameters. We use the formula previously derived to compute its dual $g(x_1, x_2)$ and find its partial derivative with respect to x_1 to be

$$\frac{(f_1 1 C - f_1 C_1, f_1 2 C - f_2 C_1, C_1)^T}{C^2}$$

where $C = x_1 f_1 + x_2 f_2 - f + p_1 f_1 + p_2 f_2 - p_3$ and $C_1 = \frac{\partial}{\partial x_1} C$. We note that $f_1|_{(0,0)} = f_2|_{(0,0)} = 0$. Thus the above partial derivative evaluates to $(-f_1 1 p_3, -f_1 2 p_3, p_1 f_1 1 + p_2 f_1 2)^T / C^2$ at the corresponding point. Similarly

$$\left[\frac{\partial}{\partial x_2} g(x_1, x_2)\right]|_{0,0} \;\; = \;\; \frac{1}{C^2}(-f_1 2 p_3, -f_2 2 p_3, p_1 f_1 2 + p_2 f_2 2)^T.$$

Note that the third row of the Jacobian matrix

$$\begin{pmatrix} -f_1 1 p_3 & -f_1 2 p_3 \\ -f_1 2 p_3 & -f_2 2 p_3 \\ p_1 f_1 1 + p_2 f_1 2 & p_1 f_1 2 + p_2 f_2 2 \end{pmatrix}$$

is always a linear combination of the first two (recall that $p_3 \neq 0$) and hence its rank is exactly that of the upper square matrix, which is completely determined by the partial derivatives of up to the second order.

It is well known that the pedal of a surface has a cuspidal edge corresponding to the parabolic curves on the original surface. It would have been easy to reach the same conclusion for the inverted pedal by noting that the inversion map is continuously differentiable up to any order and hence preserves the dimension of such singularity. However, the result for the quadratic hypersurfaces together with a generalization of the above calculations enable us to conclude more generally that the dimension of the dual at a point having k non-Morse terms is $d - k - 1$. Note that we do not claim to have categorized different types of singularities. In terms of the computation of the convex hull, the above observation is sufficient for us to specify the algebraic preprocessing of the surfaces for the algorithm.

For hosts of lower dimensions (less than $d-1$), we observe that eliminating d variables from a system of $(d-k)+(k+1)$ equations in general leaves us with a single equation. We thus expect that the inverted pedals of most cells, *regardless of their dimensions*, to be $(d-1)$-dimensional. The only degenerate case in E^3 is a straight line edge, which dualizes to a straight line edge. We will therefore not discuss the details of lower dimensional hosts since our focus is on 3-D algorithms and since such edges can be easily detected and discarded.

Chapter 6

Convex Hull Algorithms

In this chapter we present an asymptotically efficient algorithm for constructing the convex hull of a set of algebraic surface patches and other algorithms directly related to convex hulls. We recall that in the 3-D rectilinear case, the presence of the adjacency information on the simple polyhedron does not admit a more efficient algorithm to be developed than a simple one without taking advantage of it. Similarly, our algorithm will accept as input an unordered set of surface patches (up to their boundaries) and therefore does not rely on the connectedness or homogeneity of the faces. It turns out that it is in fact nearly optimal.

The algorithm works in three major steps. After preprocessing, the surface patches are first either discarded (if they are hyperbolic, parabolic, or elliptic but concave) or dualized. Equally important, and quite often probably even more so, is to dualize the boundary and dangling curves and the vertices. In the second step we begin with an arrangement of surface patches in the dual space. A number of cell construction algorithms may be chosen from and applied to create the convex cell that contains the origin in its interior. For the asymptotic analysis we choose a reduction to the construction of the lower envelope. Finally the boundary of the convex hull need to be recovered efficiently from that of the polar set.

6.1 Dualizing the Input

6.1.1 Choosing the Coordinate System

Application of Theorem 4.1 requires that the convex hull of the object K contain the origin in its interior. To this end, we may arbitrarily pick four or more points which lie on the input surface patches and verify that they are linearly independent. Their average is then in the interior of the convex hull and may be used as the origin of the new coordinate system. It may happen that we can not find any quadruples of linearly independent points because all of the input surface patches lie on a linear subspace In this case we perform an appropriate coordinate transformation which brings the affine hull of K to one of the coordinate planes and invoke the corresponding algorithm in a lower dimension.

In practice we would like not only the aforementioned property to be true, but also that the origin is as far from the boundary of the convex hull as possible. It is easy to see that if a sphere of radius $r > 0$ centered at the origin is entirely contained in the convex hull of K, then K^* is entirely contained in a sphere of radius $\frac{1}{r}$ centered at the origin in the dual space. Thus we may perform the above choice of quadruples of points several times and compute the ratio of the volume to the area of the tetrahedron they define. This ratio is proportional to the radius of the inscribing sphere of the tetrahedron. We choose the quadruple which gives the largest ratio and use the center of its inscribing circle as the new origin. Let us denote this chosen tetrahedron by T_0 and from now on we will refer to the coordinates and equations of the input surface patches with respect to this new coordinate system.

6.1.2 Breaking the Faces at Singularities

Most of the subsequent algorithms we will invoke assume that they are given "nice" inputs where each face is smooth. We recall that we make the same assumption about the input to the convex hull algorithm. However, in dualizing the faces we may introduce singularities as shown in the previous chapter. We are thus responsible for breaking the dual surfaces at these singularities in order for subsequent algorithms to work.

It is relatively straightforward to break each 2-face φ along the curve of support $\gamma \subset \varphi$ from the origin. This curve corresponds to an improper curve in the dual space. We also have to process the 1-faces in the primal space similarly by breaking them at points at which the tangent pass the origin (which do not exist if we make the general position assumption). Such singularities correspond to the fold, the "least singular" of all singularities. The study of more interesting singularities is a major subject in itself. [7, 18] Fortunately we don't have to deal with it in its entirety in order to describe the algorithm. Specifically, we have to break each dual surface at singularities into subfaces of lower dimensions, each of which is guaranteed to be regular for its dimension, but we need not worry about how these pieces fit together topologically except up to the subface relationship.

In Section 5.5 we see that a point on the dual surface is regular as long as its corresponding point in the primal space has a non-degenerate second fundamental form. More generally, the dimension of the dual surface nearby a point $\tilde{\mathbf{p}}$ is at least (and usually exactly) as high as the number of non-vanishing eigenvalues of the second fundamental form at the corresponding point $\mathbf{p}$ in the primal space. Here we are using the term "dimension" in a computational context where we require a parameterization of the surface be explicitly established such that its Jacobian matrix has the rank equal to the claimed dimension.

Thus we introduce artificial vertices in the dual space which correspond in the primal space to flat points [19] where the second fundamental form vanishes completely. We further introduce artificial edges in the dual space which correspond in the primal space to the parabolic curves, where the determinant of the second fundamental form vanishes. In general the original surface is now divided into two or more regions each of which is either hyperbolic or elliptic. We may discard all hyperbolic regions since their contribution to the convex hull, if any, will be manifested by their boundary curves. If, on the other hand, the original surface happens to be a developable surface, then the entire surface is parabolic and can be discarded for the same reason as hyperbolic regions in ordinary surfaces.

Note that we need not introduce these extra lower dimensional faces in the primal space. Specifically, a flat point in the primal space does not contribute more than one tangent plane, if at all, to the convex hull. It does not create a plane in the dual space but is rather simply used to delimit the artificial edges in the dual space.

Similarly, a parabolic curve in the primal space contributes to the convex hull no more than the envelope of the tangent planes to the host surface taken along that curve, which forms a developable surface. Thus its dual is not a developable surface but merely a portion of a space curve which delimits the elliptic or hyperbolic regions.

We finally observe that one type of singularity is absent in the dual space, namely the folds with respect to the origin. More precisely, a dual surface patch φ cannot contain any point at which the tangent passes the origin, for such tangents would correspond to improper points in the primal space.

6.1.3 Bounding the Work Space in the Dual Space

Unlike the polar set dual, the compact object K in the primal space may give rise to unbounded portions in the (inverted pedal) dual space even though we have chosen our coordinate system carefully. We will thus have to further cut the dual surfaces and construct a reasonably small work space.

For each coordinate axis, for example the x-axis, find its intersection with the boundary of the tetrahedron T_0. Clearly the two intersections $x = x_{\text{min}}$ and $x = x_{\text{max}}$ must be such that $x_{\text{min}} \leq -r$ and $x_{\text{max}} \geq r$ where r is the radius of the inscribing circle of T_0. These six intersection points define an octahedron entirely contained in the convex hull. The dual of this octahedron is a rectangular box $\mathcal{B}$ whose faces are parallel to the coordinate axes and which contain the polar set. We thus clip each dual surface by this solid.

6.2 Constructing the Polar Set

Cells in an arrangement of hypersurfaces can be constructed by any of a variety of well known algorithms. For example, a cylindrical decomposition would suffice for this construction. (This approach is similar to the one taken by Kriegman in [32].) However, more specialized and more efficient algorithms are also available for this particular problem [23].

As we are only interested in one particular cell (which is also *convex*), we may obtain an asymptotically even more efficient algorithm by way of reduction to the computation of the lower envelope, a problem studied in [46]. This randomized algorithm takes as input Σ a set of N algebraic surface patches of bounded degree without folds, each with a bounded number of similarly constrained subfaces. It then produces the *minimization map*. This is basically a planar graph with additional geometric information describing the lower envelope of Σ, or the set of points right above the plane. The algorithm runs in expected time $O(N^{2+\epsilon})$ for all $\epsilon > 0$.

Let $\mathbf{e}$ be the unit vector pointing in the direction of positive z-axis Consider the following function defined on the halfspace $z > 0$:

$$\mathcal{K} : \mathbf{x} \longmapsto \alpha^{-1}\mathbf{x} + (\alpha - 1)\mathbf{e}$$

where $\alpha = \mathbf{x}\cdot\mathbf{e}$. Its effect is to take vectors along the same direction to vectors ending in the same line parallel to $\mathbf{e}$, with their order on the original direction preserved. The intuitive interpretation above also serves to show that if $\mathcal{O}$ is in the kernel of a star-shaped cell C in an arrangement of surfaces (which it is in our case), then half of $\partial C \cap C$ is mapped to the lower envelope of the new arrangement. We note that $\mathcal{K}$ and its inverse

$$\mathcal{K}^{-1} : \mathbf{y} \mapsto \alpha\mathbf{y} + \alpha\mathbf{e} - \alpha^2\mathbf{e}$$

where $\alpha = \mathbf{y}\cdot\mathbf{e}$, are both algebraic.

The transformation produces a valid input for [46]. Specifically,

- Each transformed face obviously has a bounded number of subfaces as does the original face.

- The transformation preserves the smoothness of the faces and increases the degrees of the host surfaces by a factor of at most two.

- Each transformed face φ is fold-free. For let the host φ be given as $f(x_1, x_2, x_3) = 0$ and let $(y_1, y_2, y_3) = (\frac{x_1}{x_3}, \frac{x_2}{x_3}, x_3)$. Then the normal to the transformed face can be found by chain rule to be $(y_3 f_{x_1}, y_3 f_{x_2}, y_1 f_{x_1} + y_2 f_{x_2} + f_{x_3})$. It is parallel to the xy-plane (namely it is a fold) precisely when the tangent to φ passes the origin, which is impossible according to our earlier observation.

Constructing K^* the polar set of the curved polyhedron is now relatively straightforward. We first need to bound the work space. The transformation $\mathcal{K}$ introduces undesirable poles "along the horizon"—a point with nearly zero z-coordinate is transformed to a point with a very large z-coordinate. For each face of the box $\mathcal{B}$ bounding the polar set, a pyramid is formed with the face as the base and $\mathcal{O}$ as the top. The portion of the arrangement of the dual surface patches within the pyramid is then transformed with respect to the corresponding coordinate plane. It is easy to see that the transformation of this portion is bounded and the lower envelope can therefore be computed and transformed back. Finally the six patches of $\partial\texttt{conv}\,K$ corresponding to the six pyramids are sewn together.

As a by-product of this reduction, we have shown that

Theorem 6.1 *The (structural) complexity of the convex hull of a curved polyhedron with N faces is $O(N^{2+\epsilon})$ for any fixed $\epsilon > 0$ provided that the number of subfaces for each face is bounded by a constant.*

6.3 Recovering the Convex Hull

Regardless of the algorithm we choose to construct the polar set, we can recover the convex hull from the polar set in time linear in the number of features on the polar set. For simplicity, we shall assume that the cell construction algorithm has produced a *Doubly Connected Edge List* of [39] for representing the boundary of a convex curved polyhedron embedded in E^3 as a connected planar graph G. Most existing general-purpose representations are equivalent to this simple structure as far as planar graphs are concerned, including the one generated by the reduction step described above. We could apply the same procedure to K^* which was applied to K in order to find K^{**}, namely the convex hull. However, we really need not compute the inverted pedal of the 2-faces of K^* since they are contained in the input 2-faces. Better yet, we do not even construct the arrangement of the inverted pedal of the faces of K^* since we are already dealing with a convex object. Instead, we just modify G to obtain the adjacency information of K^{**} and compute the geometric information as follows.

First a new planar graph G' is constructed whose vertices correspond to the vertices, edges, and regions of G. The vertices of G' represent the 2-faces of $\texttt{conv}\,K$, the dual of the polar set. An edge is drawn between two vertices of G' if the two corresponding entities in G are incident. These represent the 1-faces of $\texttt{conv}\, K$. Clearly G' is planar and the regions of G' represent the 0-faces of $\texttt{conv}\, K$.

Denote by v the vertex in G that corresponds to $v' \in G'$. To recover faces of $\texttt{conv}\, K$, we observe that a vertex v' in G' represents

1. part of an input surface patch in the primal space if v is a region in G. Its geometric information is already present in the input.

2. a new developable surface patch which bridge two original surfaces if v is an edge in G. Its geometric information is computed by the formulæfor inverted pedals.

3. a triple common supporting plane if v is a vertex in G. Its equation is easily found by computing the dual of the corresponding vertex.

To recover edges and vertices, we observe the following cases according to what kind of incidence in G an edge e' in G' corresponds to.

1. *Vertex–edge adjacencies.* In this case, e' represents the ruling in $\texttt{conv}\, K$ that marks the transition from a plane π to a developable surface σ. Its geometric information is easily computed as the dual of the tangent straight line of $\tilde{\sigma}$ at $\tilde{\pi}$.

2. *Edge–region adjacencies.* Here e' represents a curve in $\texttt{conv}\, K$ that marks the transition from a developable surface σ to an input face φ. This curve can be considered the intersection of σ and φ, and hence the dual of the envelope of the tangent planes of $\tilde{\varphi}$ along $\tilde{\sigma}$ (that is, their common tangent planes in the dual space). In particular if $\tilde{\sigma}$ can be parameterized then so can this new edge.

3. *Vertex–region adjacencies.* Now e' represents a vertex in $\texttt{conv}\, K$ that marks the transition point from a plane to an original surface. Its coordinates can be found by dualizing the tangent plane of the region at the vertex.

The size (number of vertices) of G' is at most six times that of G since both are planar. Computing the geometric information in each of the above case contributes only a constant factor determined by the maximum degree of the polynomials. Thus

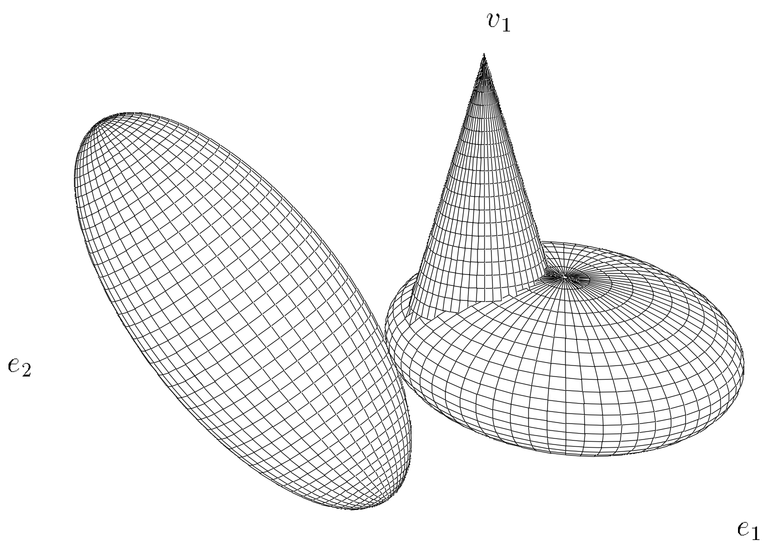

Figure 6.1: The union of two ellipsoids and a truncated cone.

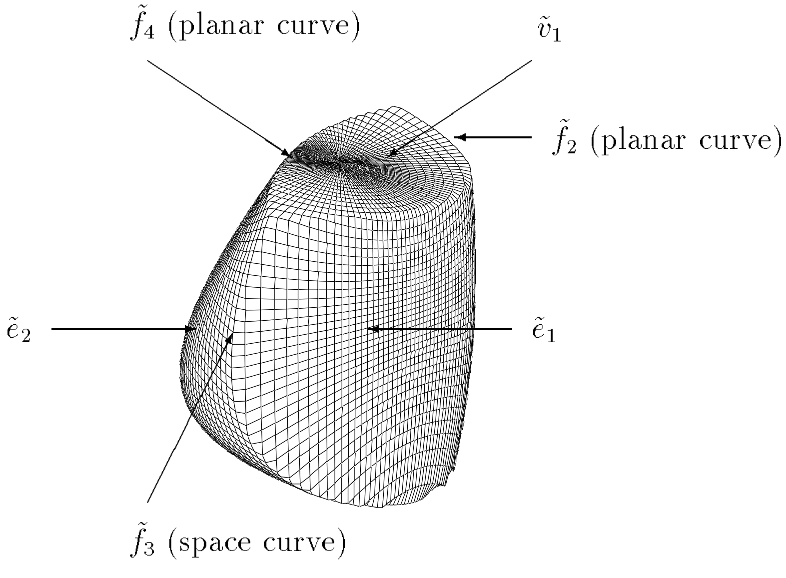

Figure 6.2: The polar set of the object, recovered as the cell containing the origin in the arrangement of dual surfaces.

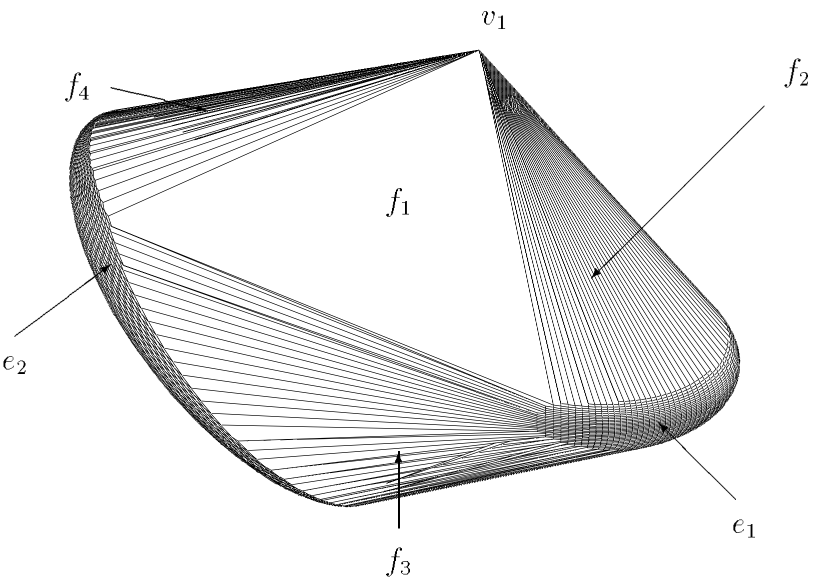

Figure 6.3: The convex hull of the original object, recovered from the dual of the polar dual (previous figure). Note that f_1 is planar since it is dual to the triple intersection (vertex) in the previous figure. Other new surfaces (f_2, f_3, and f_4) are developable, and arise from the corresponding edges of the polar set.

the combinatorial complexity of the convex hull and also that of computing $\texttt{conv}\, K$ from K^* is the same as the complexity of ∂K^*, up to a constant factor.

Figure 6.1 shows two ellipsoids and part of a cone in space. The contribution of the cone to the convex hull is limited to its vertex. The ellipsoids have implicit equations

$$\begin{aligned} e_1 &: \quad 29x^2 + 36y^2 + 180z^2 - 24xy - 48x + 144y - 36 = 0 \\ e_2 &: \quad 225x^2 + 612y^2 + 388z^2 - 768yz - 2448y + 1536z + 1548 = 0\ . \end{aligned}$$

Their inverted pedals are computed to be:

$$\begin{aligned} \tilde{e}_1 &: \quad 36x^2 + 9y^2 + 5z^2 + 24xy - 20y - 5 = 0 \\ \tilde{e}_2 &: \quad 100x^2 - 3y^2 + 153z^2 + 192yz + 100y - 25 = 0. \end{aligned}$$

Figure 6.2 shows the polar set and Figure 6.3 its convex hull.

In summary, we have shown that

Theorem 6.2 *Let K be a curved polyhedron with N algebraic faces of bounded degree each with a bounded number of subfaces. Then* $\texttt{conv}\, K$ *can be computed in randomized expected time $O(N^{2+\epsilon})$ for any $\epsilon > 0$, where the constant depends on the maximum degree of the polynomials in the input as well as on the maximum number of subfaces each face may have.*

proof:

> Each step in the series of computations involve processing faces individually (up to their subfaces). Thus each original face is processed at most only a constant number of times. The transformations and preprocessing create no more new faces than a constant number of times the input faces, as determined by the maximum degree of the input polynomials. Hence the complexity of the lower envelop algorithm dominates and we have the desired conclusion. $\square$

It is known that the complexity of the convex hull of N spheres can have complexity $\Omega(N^2)$ [6]. Generally one expects a worst case complexity of $\Omega(N^2)$ for algebraic surfaces of bounded degrees. It is also conjectured that the complexity is actually $\Theta(N\lambda_q(N))$ where q is a constant depending on the degree of the surface and $\lambda_q(N) = O(N \log^* N)$ is the length of the maximum Davenport-Schinzel sequence of order q on N symbols. See for example [1] for a short summary and further references.

6.4 General Curved Polygons in E^2

In Chapter 2 we study the convex hull algorithm which takes advantage of the fact that the input is a *simple* curved polygon. If the input is a general curved polygon which consists of dangling edges and disconnected components, we may no longer have a linear time algorithm. The special case in which the curved polygon consists of a set of N discrete points serves to give a lower bound of $\Omega(N)$ for the convex hull problem. In this section we show that this lower bound can be achieved.

Let K be a curved polygon consisting of N isolated vertices and/or algebraic edges of degree at most D. As usual, the coordinate system is so chosen that the origin lies in the interior of the convex hull. As in the 3-D case, we first break the edges at the turning points, namely the folds as seen from the origin. We also break them at the inflection points, which are the 2-D analogs of the parabolic or flat points in 3-D. Finally, we break long edges into pieces with angular spans no more than π. The same arguments used in the simple curved polygon case applies here as well and we conclude that the number of curve segments does not increase by more than a factor of $4D$.

For each piece or each isolated vertex, we compute its dual. An isolated vertex dualizes to a halfspace of interest to us, namely the one which contains the origin. A curve segment consists of two vertices and an open edge. Its dual as a whole again defines a halfspace delimited no longer by a straight line but rather by one piece of curve segment and at most two rays (fewer if one or both of the vertices in the primal space are turning points).

Now that we have $O(DN)$ halfspaces, we may perform a standard divide-and-conquer technique to find their intersection. The choice of the origin ensures that during the merging process we always have a pair of convex regions with nonempty intersection. Their intersection can be found in time linear in the number of the curve edges bounding the two convex regions.

We bound the complexity of the algorithm as follows. We first observe that the convex hull of a set of k *non-intersecting* simple curve segments in the plane has a structural complexity of $\lambda_2(k)$. In other words, there do not exist a pair of the input curve segments a and b such that their portions alternately appear in the convex hull by more than three times (counting both), for otherwise they would intersect each other. Thus the convex hull of k curve segments has a structural complexity of at most $\lambda_2(k) = 2k - 1$, whether they be algebraic and if so regardless of their degrees. It follows that our algorithm runs in time $O(N \log N)$ where the constant depends on D. To summarize, we have

Theorem 6.3 *The convex hull of a set of N smooth algebraic curve segments and/or isolated vertices consists of at most $2N - 1$ smooth pieces and can be constructed in time $\Theta(N \log N)$ with an overhead determined by the maximum degree of the curves.*

6.5 Avoiding Hull Construction

In some applications the convex hull of a curved polyhedron K may not always have to be explicitly constructed. For example one may wish to make a small number of queries as to whether an arbitrarily given point in space lies in the convex hull of a curved polyhedron but otherwise not interested in the convex hull itself. In this case a much simpler algorithm can be used without actually constructing the convex hull.

A point $\mathbf{p}$ is

- in the interior of $\mathtt{conv}\, K$ iff the dual plane $\tilde{\mathbf{p}}$ strictly bounds K^*;
- in the exterior of $\mathtt{conv}\, K$ iff the dual plane $\tilde{\mathbf{p}}$ cuts K^*; and
- on the boundary of $\mathtt{conv}\, K$ iff the dual plane $\tilde{\mathbf{p}}$ supports K^*.

The dual problem is therefore only slightly more difficult than the 2-D convex hull finding problem. Suppose the input faces have been pre-processed as in the complete algorithm for constructing the convex hull. In the dual space we intersect $\tilde{\mathbf{p}}$ with the closure of $\tilde{\varphi}$ for each face φ. We discard all surface patches which do not intersect $\tilde{\mathbf{p}}$. If none of the dual surface patches remain, we conclude that $\tilde{\mathbf{p}}$ strictly bounds K^*.

Now suppose that some dual surface patches do intersect $\tilde{\mathbf{p}}$. If $\tilde{\mathbf{p}}$ supports any of the dual surface patches, we mark $\mathbf{p}$ as being potentially on the boundary. In any case, we actually construct the convex "half plane" determined by each intersection. The intersection itself is generally a curve segment with two ends. We always make it into a closed or infinite path by adding two rays tangent to the segment at its end points.

Now we may invoke an algorithm similar to the intersection algorithm used in the construction of the general (non-simple) 2-D convex hull, except that we don't have a fixed point on $\tilde{\mathbf{p}}$ which is guaranteed to lie in the intersection. Nevertheless in the merging process, we can take linear time to decide whether an arbitrarily chosen point $\mathcal{O}'$ lies within both convex regions. If so, we just proceed to merge their boundaries as before. Otherwise we break the two circular lists of boundary curve segments into at most four sublists, each of which has monotonically increasing or decreasing polar angles with respect to $\mathcal{O}'$. The intersection of the convex regions can still be found in linear time. If the intersection of any pairs turns out to be empty during this process, then $\tilde{\mathbf{p}}$ strictly bounds K^*. Now suppose the intersection exists. The algorithm then returns that $\mathbf{p}$ lies on the boundary of $\mathtt{conv}\, K$ if it was previously marked as being potentially so; or that $\mathbf{p}$ lies outside $\mathtt{conv} K$ otherwise. In the latter case where $\mathbf{p}$ lies outside the convex hull, it is straightforward to actually construct the supporting cone from $\mathbf{p}$ to K.

Theorem 6.4 *Let K be a curved polyhedron with N faces. A point $\mathbf{p}$ in space can be classified against $\mathtt{conv}\, K$ in time $O(N \log N)$ without explicit construction of $\mathtt{conv}\, K$. If $\mathbf{p}$ lies outside the convex hull, the supporting cone from $\mathbf{p}$ to $\mathtt{conv}\, K$ has structural complexity $O(N)$ and can be constructed in the same amount of time as the classification problem.*

6.6 A Practical Approach

6.6.1 Rationale

In many applications limited precision of the output is tolerable as long as the error is predictable, whereas an exact algorithm which is prohibitively slow is to be avoided. The reduction of the convex hull problem to the lower envelope problem—although serves very well its purpose of giving us a satisfactory asymptotic bound—may incur too much algebraic and coding overhead. In this section we consider a numerical approximation for which

- an intuitive measure of error can be defined;
- the error can be estimated and controlled;
- the output can be further processed to generate the "exact" solution, subject to missing "small" features as determined by the error measure;
- the coding effort is minimal.

The algorithm works as follows. In the dual space we choose a set of nicely spaced directions, for example, ones corresponding to the grid points on the unit sphere with "equally spaced" (in the usual sense) longitudes and latitudes. For each direction we shoot a ray. This ray is intersected with each dual surface to find the intersection point nearest the origin. Thus we have a mesh of sample supporting planes along with the knowledge of the faces they support. This mesh can then be further processed depending on the application which requires the construction of the convex hull.

6.6.2 Precision and Error Measures

We choose the number of rays penetrating a unit area on the unit sphere as the measure of precision. Let us explain the intuition behind this choice by considering the two (disconnected) curved polyhedra in Figures 6.4 and 6.5, respectively. In the first case one reasonably asks the algorithm always to report the cone-like area as a new face even if the middle ellipsoid shrinks to the size of the needle. In the second case one should expect a program running on a machine with limited numerical precision to incorrectly report or leave out the "flat island" formed by the middle ellipsoid inside the new surface, even though the area of the reported or missing island may be rather large. Using our proposed measure of precision, the algorithm on input of the first example is not entitled to leave out the new surface, or else it would mean that tangent planes placed at such obvious different angles as those on the opposite sides of the cone are left out or identified. On the other hand,

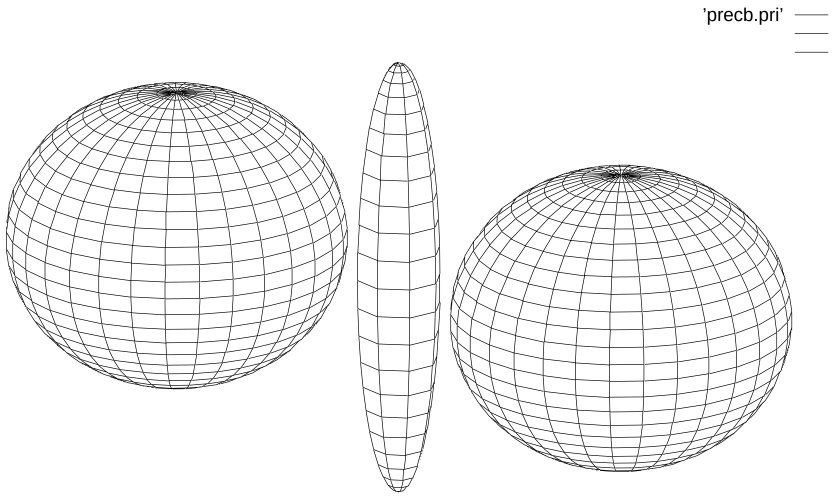

Figure 6.4: A "big" feature that should not be missed in terms of convex hull construction.

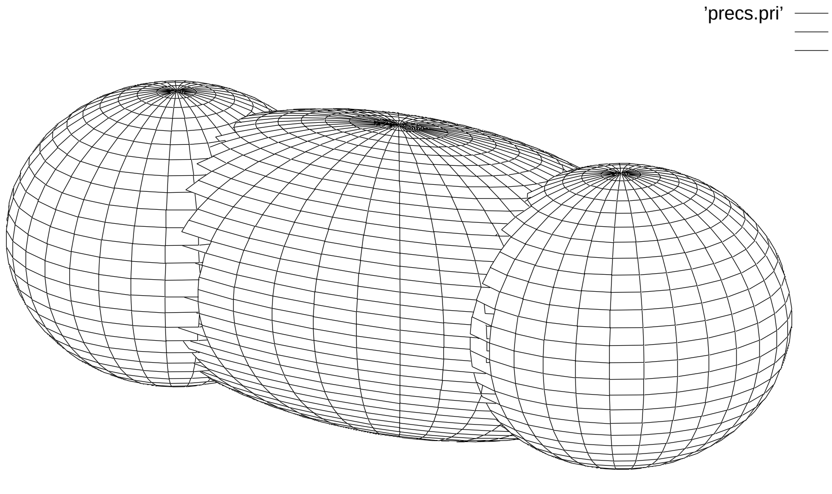

Figure 6.5: A "small" feature that could be missed if the the precision of the approximation is low.

the second example does suffer from having nearly coincident planes and should be allowed to miss the big feature in the primal space.

Such choice of precision measure can also be interpreted as weighing the sampling density (number of rays per unit area on faces in the primal space) by the Gaussian curvature of the surface since the latter is the limiting ratio between the area of the Gaussian sphere and that of the surface area in the primal space. In other words it realizes the intuition that we need more samples around surface areas which are badly curved, and don't wish to waste bandwidth on nearly flat faces.

The sampling criterion is appropriate for visualization purposes. Rendering algorithms based on the normals of the sampled facets usually prefer a "roughly constant change of normal direction" to a "roughly constant size of rendering area." This technique may be applied to curved polyhedron consisting of higher degree surface patches such as the dimpled surface in Figure 4.4, whose duals are too complicated to derive symbolically.

This measure has a geometric meaning in the dual space as well, up to a factor bounded by two positive constants. Let R_c and R_i be the radii of the smallest circumscribing sphere and largest inscribing sphere centered at the origin, respectively, of K^*. Then the sampling density (number of rays per unit area on the faces in the *dual* space) on the boundary of the polar set is easily shown to be proportional to that on the Gaussian sphere to within a factor between $\frac{R_i}{R_c^3}$ and $\frac{1}{R_i^2}$.

Figure 6.6 shows the precise correspondence between the error in the dual space and that in the primal space. Consider a small ball of radius ϵ in the dual space representing the range of error. Since inverted pedal is symmetric with respect to the origin, we may assume without loss of generality that the center of the ball lies on one of the positive coordinate axes, say the y-axis. We further simplify the discussion by looking at only a cross section of the ball where the cutting plane passes the origin, say the xy-plane.

Let the center of the boundary circle be $(0, h)$ and assume that $\epsilon \leq$. Then the matrix representation of the circle is

$$B = \begin{pmatrix} h^2-\epsilon^2 & 0 & -h \\ 0 & 1 & 0 \\ -h & 0 & 1 \end{pmatrix}.$$

Its dual

$$\tilde{B} = \begin{pmatrix} 1 & 0 & -h \\ 0 & -\epsilon^2 & 0 \\ -h & 0 & h^2-\epsilon^2 \end{pmatrix}$$

is a hyperbola centered at $(0, \frac{h}{h^2-\epsilon^2})$. Points inside B may be thought of as points on smaller balls concentric with B. Thus they dualize to tangent lines to similar-looking hyperbolas whose two sheets are closer than and sit in-between those of $\tilde{B}$. These lines all fall in the concave area in-between the two sheets of $\tilde{B}$ (Figure 6.6). Conversely, every line falling in that concave area dualizes to a point inside B.

Even though the error ball in the dual space dualizes to an unbounded volume except by the enclosing ball in the primal space, the orientation error does indeed approach zero as the error ball shrinks in size. The latter is determined by the angles between the two asymptotes of the hyperbola, namely $2\sin^{-1}(\epsilon/h)$, and becomes proportional to the radius of the error ball as $\epsilon \to 0$.

6.6.3 Other Practical Considerations

The output mesh generated by the described algorithm may be further processed to produce a planar graph describing the boundary of the exact convex hull (subject to missing features as explained earlier). We begin by explicitly constructing the planar graph represented by the mesh. For each adjacent pair of vertices V_1, V_2 in the graph which belong to two different dual surfaces, we add an intermediate vertex along the grid line $\overline{V_1 V_2}$ by computing the intersection of the two host surfaces with the plane containing V_1, V_2, and $\mathcal{O}$. Then for each grid for which any boundary grid line has changed, we use a brute-force procedure to add exact edges which correspond to the intersection of the actual dual surfaces, adding new vertices as necessary for triple intersections. Finally we remove redundant edges connecting adjacent vertices which fall in the same dual surface, removing the vertices as well when they become isolated. We may not want to carry out the last step to remove all redundant edges and vertices we could, since certain algebraic operations are simplified when the spans of the surfaces are kept relatively small. An analogous situation is demonstrated in Chapter 2 where we restrict each boundary curve to have an angular span of π or less.

In case the hull has to be constructed exactly without missing features, existing algorithms for converting CSG to boundary representations such as [38, 42] may be used in the dual space in place of the near optimal algorithm for convex cell construction. Although the asymptotic worst case behavior is worse, the similarity of these algorithms to their primal counterparts may argue in favor of their use.

In CAD and CSG applications, being able to represent the original and derived surfaces and curves analytically versus numerically means a significant saving in storage. One would even like to have a piecewise parameterized representation available for rendering and other purposes. Fortunately we can do both quite efficiently and effectively with the duals and convex hulls of curved polyhedron consisting of quadratic surface patches cut by planes. Using the fact that all dual surface patches are quadratic, it is straightforward to show that the intersection curves in the dual space can be parameterized by solving degree 4 equations. By the results in Section 4.4.3, the common supporting developable surfaces can therefore be parameterized as well.

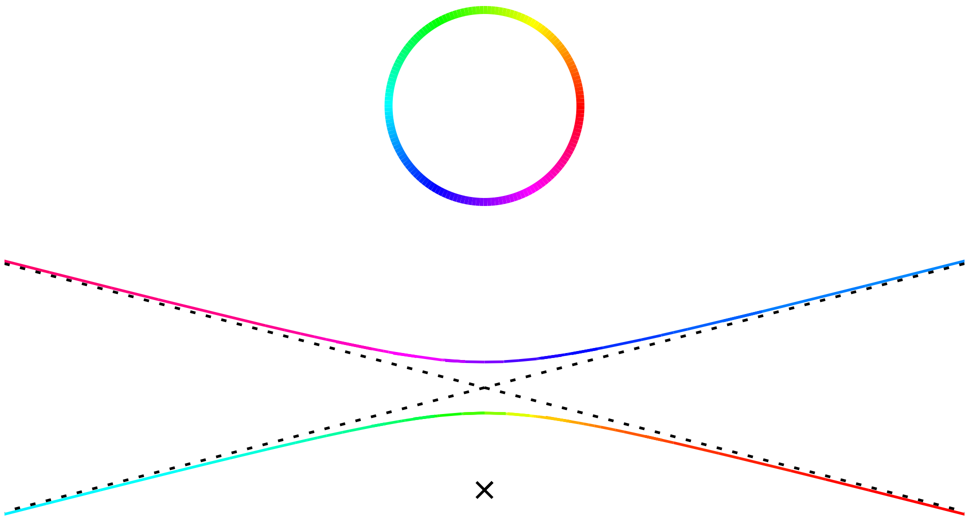

Figure 6.6: An ϵ-error circle and its dual hyperbola.

Chapter 7

Stable Poses and Capture Regions

Given a solid object K, one may wish to find what are the possible orientations at which it can rest on a flat supporting plane. One application may be in manufacturing environments where an algorithm for producing oblivious grasping plans need to take as input the possible stable poses of a part moving on a conveyor belt. [40] Another application is in model based vision algorithms where the object to be recognized rests on a horizontal flat surface such as the floor. More detailed discussions on the latter application can be found in [30].

One may ask a more general question: for each stable pose, what are the initial orientations of the part K starting from which K will eventually converge to that stable pose, assuming that the dynamics are dissipative? We apply inverted pedal to these problems and propose an alternative practical approach which in the quadratic case is far more efficient than the algorithm used in the first implementation (to the best of our knowledge) for solving these problems. [30, 31].

7.1 The Direct Approach

Kriegman pioneered the study of stable poses in [30] for piecewise smooth objects. We briefly summarize his approach here, and supplement his discussions by mentioning the difficulties of the direct approach in the verification of local minimality and in dealing with singularities. We will use a slightly different notation. In particular we place the coordinate system so that the center of mass of K coincides with the origin.

7.1.1 Summary of the Case Analysis

The conceptually simplest of all possible cases is a pose with only one contact point $\mathbf{p}$ located in the relative interior of an elliptic face. A necessary condition for the pose at $\mathbf{p}$ to be stable is that

$$\begin{cases} s(\mathbf{p}) = 0 \\ \mathbf{p} \times \nabla s(\mathbf{p}) = 0 \end{cases} \tag{7.1}$$

This condition characterizes the critical points of the potential energy function. Correspondingly, we will call poses satisfying (7.1) the *critical* poses. For a critical pose to represent a local minimum as distinguished from local maximum or a saddle point of the potential energy, $\mathbf{p}$ also has to satisfy

$$|\mathbf{p}| < \min(\frac{1}{k_1}, \frac{1}{k_2}) \tag{7.2}$$

where k_1 and k_2 are the principle curvatures of surface s at $\mathbf{p}$.

Other kinds of critical poses may be analyzed similarly. In each case a set of equations similar to (7.1) may be derived which provide the necessary and sufficient condition for a combination of contact points to determine a critical pose of K. We simply list all possibilities of the set of contact points as discussed in [30] without listing the corresponding equations:

- one regular point in a 2-face (discussed above);
- two surface points;
- one surface point and one edge point;
- two edge points;
- three surface points;
- two surface points and one edge point;
- one surface point and two edge points;
- three edge points.

To make the above a complete list we would have to include cases in which vertices are also involved. The equations involving the vertices are relatively simple and are explicitly excluded from [30]. We will not pursue this issue as our approach will require us to enumerate a different and much smaller set of possibilities.

The implementation actually takes triples and pairs of faces/edges which determine all potential critical points. The homotopy continuation method is then applied to solving each system of equations and find the contact points numerically. The running time is $O(N^3 d^9)$ for an object consisting of N features, each of which is described by algebraic equations of degree at most D. As noted in the paper, it takes about two days to compute the stable poses for a part similar to Figure 7.1 on a Sun SPARCstation 1.

7.1.2 Distinguishing Local Minima from Other Critical Poses

Once a critical pose is found, one has to check

1. whether the critical pose is a local minimum and

2. whether the corresponding supporting plane intersects any other portion of K.

The second test as described in [30] involves solving less costly systems of equations for each feature on the object (and hence does not contribute an extra factor to the complexity, assuming N is not too huge). For the first test, one need a criterion similar to the inequalities (7.2) for the other cases. In the three-point contact cases the critical pose is a local minima if and only if the supporting plane the projection of the center of mass lies within the triangle formed by the three points of contact.

The cases involving two-point contact is less straightforward. We will take the two-surface-point case as an example. Let $\mathbf{p}_1 \in s_1$ and $\mathbf{p}_2 \in s_2$ be the points of contact in a critical pose. As given in [30], $\mathbf{p}_1$ and $\mathbf{p}_2$ satisfy:

$$\begin{cases} s_i(\mathbf{p}_i) = 0 \\ \mathbf{n}_i(\mathbf{p}_i) \times \mathbf{n}_2(\mathbf{p}_2) = 0 \mathbf{n}_1(\mathbf{p}_1) \cdot (\mathbf{p}_1 - \mathbf{p}_2) = 0 \\ \mathbf{p}_1 \times \mathbf{p}_2 \cdot \mathbf{n}_1(\mathbf{p}_1) = 0 \end{cases}$$

Once we find the solution $\mathbf{p}_1 = \hat{\mathbf{p}}_1$ and $\mathbf{p}_2 = \hat{\mathbf{p}}_2$, we have to verify that a small perturbation of the orientation of the object will move it towards poses with higher

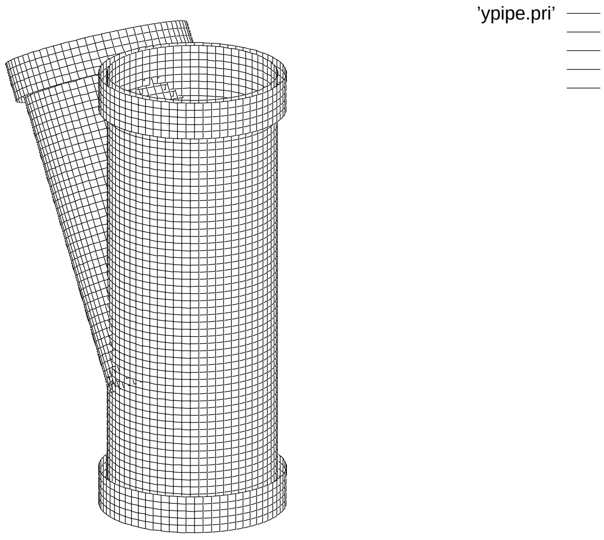

Figure 7.1: A PVC pipe fitting.

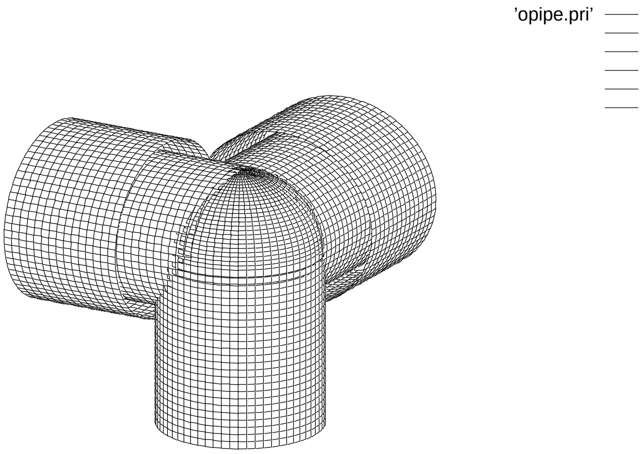

Figure 7.2: Another PVC pipe fitting.

as opposed to lower potential energy. Intuitively there are two distinct "principle" ways of perturbing K out of the critical pose—around the axis determined by $F\hat{v}ecp_1F\hat{v}ecp_2$ or around the line on the supporting plane which passes the projection of $\mathcal{O}$ and meets $\hat{\mathbf{p}}_1F\hat{v}ecp_2$ orthogonally. Other directions of perturbation are possible but it is not clear how to provide a mathematically satisfactory argument which states that it suffices to consider these two kinds of perturbations using the direct approach. Later on we shall argue, taking the dual approach, more rigorously that this "generally" true, and point out precisely when it fails to be the case.

The critical pose is stable with respect to the second kind of perturbation if and only if on the supporting plane, the projection of the center of mass falls between the two points of contact $\hat{\mathbf{p}}_1$ and $F\hat{v}ecp_2$. The other kind of perturbation corresponds to the one-dimensional solution set to (7.3) with the last equation removed. One way to verify the minimality of the solution $(\hat{\mathbf{p}}_1, \hat{\mathbf{p}}_2)$ is to choose a convenient parameterization of the solution curve (in the configuration space) and compute indirectly (using the chain rule) the first and second directional derivatives of the potential energy along the solution curve. It can then be evaluated at run time for each critical pose. The resulting formula can be formidable with a system in six variables as such. An alternative is not to compute the second derivative symbolically a priori but to approximate it numerically at run time using the first derivative.

7.1.3 Coincidences and Singularities

More often than not, industrial parts are designed with certain symmetry and coplanarity. For example, many part are designed so that when they rest on a horizontal plane, four points or rather four facets come in touch with the plane. It is hard to give a catalog of all kinds of possible singularities. The "stacked ellipsoids" in Figure 7.3 shows another example. Besides the two obvious stable poses of one-point contact, there are two other poses each making a 3-point contact with the supporting plane. In terms of equation solving, we have a one-point contact that gives a critical pose (but not stable by itself). Two such critical solutions together constitute a stable pose. However, a third such critical solution happens to give the same pose. Many common parts consist of cylinders, cones, or other surfaces of revolution, and it is not uncommon for their center of mass to be designed to lie on the axis of symmetry. The case analysis listed in [30] and reviewed above, when supplemented by the other cases involving vertices, give a catalog of cases to which other cases can be reduced. The cost is, however, expensive in terms of the number of coplanar features. For a critical pose consisting of k contact features, the implemented algorithm as described will compute $\Theta(k^3)$ sets of equations and as many number of tests on minimality, only to arrive at the same conclusion for this one critical pose. Then one faces the task of partitioning all of the computed stable poses into equivalent classes, each of which consists of numerically nearby but symbolically distinct solutions.

Agrawal studies various kinds of coincidences in his thesis [3] and proposes a general approach for dealing with them. Of particular interest to us in our application here are *coincidence by design* as exemplified above, and *coincidence by theorem*. Considering the number of cases, it is at best tedious to modify the algorithm in [30] to take advantage of the proposed solution by symbolically sharing their parameters. If all of the coefficients in the input equations representing the faces are integral (or rational, for that matter), such coincidence can be handled as if it were coincidence by theorem. In our context it means that we may use interval arithmetic to compute the solutions and detect (or reject) the coincidence based on the identity (or the lack of it) of the involved polynomials as suggested by random evaluations. Either way, such overhead will render the direct approach even less attractive in regards to running time.

Coincidence and symmetry may bring up singularities in terms of equation solving, which require the algorithm to do more than just recognizing the coincidence and solving only a representative set of equations. Consider the toy made of two balls and a connecting rod in Figure 7.4. Suppose that the center of mass lies exactly at the center of symmetry (half way between the balls, on the axis of the rod). This is not a cylinder by itself, which is treated specially in the implementation of [30] and whose symmetry is utilized to speed up computations. Such a two-point contact corresponds to a one-dimensional set of solutions in the configuration space and is likely to cause numerical problems.

We have used simple examples to illustrate the variety of possible special cases which may arise due to co-planarity and singularity. When the input surfaces are of higher degrees, more complicated and interesting phenomena may conceivably

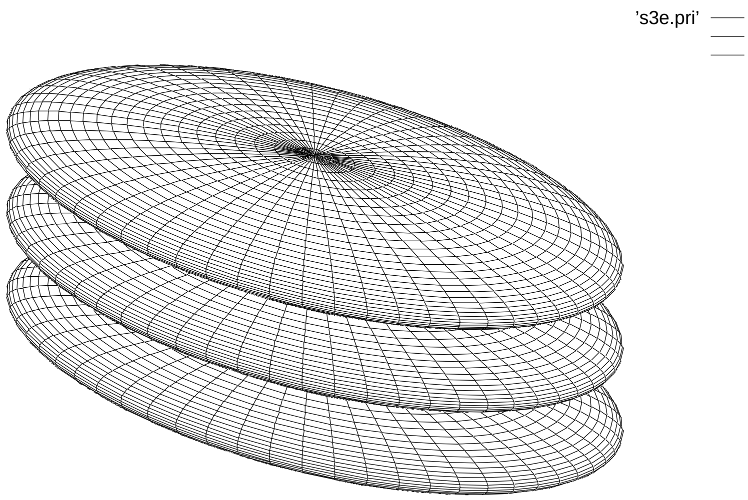

Figure 7.3: Collinear "two-point" contacts.

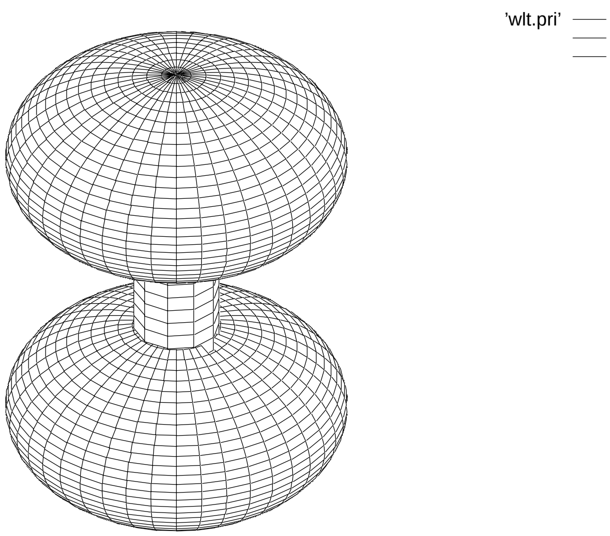

Figure 7.4: Singularity caused by symmetry.

occur. Compounded with the issue of identifying stable poses among critical poses, and with the number of different cases (recall that we left out the vertex cases), a thorough and reliable implementation based on the direct approach will quickly get out of the hand.

7.2 The Dual Approach

As noted in the paper, one weakness of the direct approach is the lengthy execution time spent in solving systems of equations. We will look at the dual approach which not only reduces the number of cases to consider but also admits much faster numerical methods when the objects under consideration are quadratic surface patches cut by planes. Moreover, the recognition of minimality, coincidences, and singularities become more intuitive. It is more straightforward to exhaustively enumerate, with respect to inputs of bounded degrees, all possible degenerate cases which may arise. It is also a less formidable task to construct a complete implementation which handles such special cases correctly.

7.2.1 Case Analysis

We begin by stating the problem in a slightly different way. Given a curved polyhedron K in space, what are the possible planes which may serve as the floor for K to rest upon stably? The candidates consist of all the supporting planes of K, which may be parameterized by their orientation. In our terms, the stable poses correspond to the local minima of the potential energy as a function of the orientation of the supporting plane. For our purpose, the potential energy is in turn equivalent to the distance of the supporting plane from the origin.

Thus we choose the center of mass to be the origin and find K^*. We shall then find the local *maxima* of the distance $|\mathbf{p}|$ from any point $\mathbf{p}$ on $\partial(K^*)$ to the origin since this distance is the reciprocal of the distance from the corresponding supporting plane $\tilde{\mathbf{p}}$ to the origin. Neglecting coincidences and singularities for the moment, there are three cases which an implementation taking the dual approach will have to consider. Two cases consist of six or more sub-cases whose distinctions are irrelevant to the stable pose algorithm but are implicitly taken care of by the dual mechanism.

1. $\mathbf{p}$ *is a vertex of* ∂K^*. It cannot be the dual of a facet present in the input since facets do not contribute to the convex hull and are not dualized at all. (Recall that by faces we usually mean open faces and that the contribution of a closed facet is accounted for by its 0-subfaces.) For similar reasons, it cannot be the intersection of a curve and a face in the dual space. It is generally the intersection of three faces, each of which may be the dual of either a vertex, an edge, or a 2-face. These cases correspond to the three-point contact cases in the direct approach.

2. $\mathbf{p}$ *is a regular edge point.* The edge is generally the intersection of two faces in the dual space, each of which may be the dual of a vertex, an edge, or a face. These cases correspond to the two-point contact cases in the direct approach.

3. $\mathbf{p}$ is a regular surface point. In this case the surface φ can only be the dual of a face. It cannot be the dual of an edge for otherwise φ would be a developable surface and it is not difficult to verify that there cannot be a local maximum of the function $x^2 + y^2 + z^2$ on such a surface.

7.2.2 Coincidences

The various difficulties in identifying the stable poses, in the co-planarity or co-linearity, and in singularities of the systems of equations are also present in the dual approach, but in a manner that admits a more systematic treatment. Let us first consider the following change of variables in the open halfspace $x > 0$ of the dual space:

$$x = r \cdot \frac{1-s^2}{1+s^2} \cdot \frac{1-t^2}{1+t^2} \tag{7.3}$$

$$y = r \cdot \frac{1-s^2}{1+s^2} \cdot \frac{2t}{1+t^2} \tag{7.4}$$

$$z = r \cdot \frac{2s}{1+s^2} \tag{7.5}$$

and

$$
\begin{aligned}
s &= \frac{y}{x + \sqrt{x^2 + y^2}} \\
t &= \frac{z}{\sqrt{x^2 + y^2} + \sqrt{x^2 + y^2 + z^2}} \\
r &= \sqrt{x^2 + y^2 + z^2}
\end{aligned}
$$

It is readily verified that this change of variable is bijective and continuously differentiable up to any order between the open halfspace $x > 0$ and the corresponding open set $\{(s, t, r) \ : \ -1 < s, t < 1 \ \text{ and } \ r > 0\}$. The family of concentric spheres centered at the origin are mapped to the iso-surfaces $r = c$ for each fixed c in the (s, t, r)-parameter space. Thus $\partial K^* \cap \{(x, y, z)^T \ : \ x > 0\}$ corresponds to the graph of a well-defined function $r = f(s, t)$ of s and t. Note that f is only piecewise smooth.

The stable poses correspond to the local maxima of the graph of f. The three cases in the dual approach remain topologically the same when considered in the parameter space as the change of variables does not alter the smoothness (or roughness for that matter) of ∂K^*. Thus the candidate critical poses correspond, in the parameter space, to vertices, to regular curve points at which the tangent is horizontal, or to regular surface points at which $\nabla f = 0$, All situations involving co-planarity of more than three contact points are accounted for by vertices of degree higher than 3. Similarly, co-linearity of more than two contact points are accounted for by points belonging to an edge which is the intersection of more than two smooth surfaces.

7.2.3 Singularities

Singularities corresponding to surfaces of revolutions manifest in the parameter space as horizontal "ridges" along which all poses are equally stable. More generally, a regular surface point at which the Hessian of f does not have a full rank corresponds to a singular pose.

- A point with vanishing Hessian corresponds to contact at an umbilical surface point $\mathbf{p}$ where the osculating spheres to any surface curves passing $\mathbf{p}$ at $\mathbf{p}$ are centered at the center of mass.

- A point with rank 1 Hessian corresponds to one of the following situations:
  - contact at a regular surface point $\mathbf{p}$ where one of the circles of curvature along the principle directions at $\mathbf{p}$ is centered at the center of mass.
  - contact at a regular edge point $\mathbf{p}$ where the circle of curvature at $\mathbf{p}$ is centered at the center of mass.

In such cases higher differentials up to the maximum degree possible, and only these, need be examined to decide whether the the corresponding critical pose is a stable pose.

Similarly, we may have singular poses corresponding to a regular edge point in the parameter space with vanishing second order directional derivative. In this case the osculating plane of the edge point is horizontal. The corresponding 2-point contact in the primal space is less intuitive as explained below. Note that each of the two contributing features may be either vertices, edges, or 2-faces, although they cannot be both vertices simultaneously. Also recall that the edge corresponds to, in the primal space, the envelope of the two features, which is a developable surface. Now in the dual space such an edge corresponds to a curve whose osculating sphere at this point is centered at the origin. There are two cases in the dual space:

Case A The osculating plane does not pass the origin. Then in the primal space there is a unique cone such that it contains the line of support as a ruling, its vertex lies on the edge of regression of the envelope surface, and its axis passes the center of the mass. This cone has a contact of order two or higher with the envelope and represents a critical pose which may or may not be stable.

Case B The osculating plane passes the origin. Then in the primal space there is a unique cylinder such that it contains the line of support as a ruling, its axis passes the center of mass (and hence its axis is horizontal), and its axis passes the center of the mass. This cylinder has a contact of order two or higher with the envelope and represents a critical pose which may or may not be stable. Figure 7.4 shows such an example.

In either case, higher order derivatives of the edge in the parameter space must be investigated to determine whether the pose is stable.

7.2.4 A Complete Algorithm

We may now sketch how an implementation can identify the stable poses among all critical poses while taking coincidences and singularities into account. A suitable coincidence resolution scheme such as [3] is first applied to resolve vertex and edge coincidences. Next the parameters corresponding to the critical poses are computed as described in the three major cases. Each critical point $\mathbf{p}$ in the parameter space may have zero or more edges incident upon it depending on the case as well as on the number of coincidences at that point. To make sure that $\mathbf{p}$ is a local maximum (in the parameter space), we test whether f is "negative definite" around $\mathbf{p}$ by iterating over all incident edges as well as all surrounding surface sectors. The test for each edge or sector is performed and singularities are examined by looking at as many higher order directional derivatives or differentials as necessary, until a conclusion is reached or up to the upper bound of the surface degree. In the surface sector case the range of the sector has to be taken into account.

7.3 Capture Regions

The technique in the previous section naturally extends to the computation of capture regions—the regions in the parameter space within each of which all points correspond to starting poses that will eventually converge to the same stable pose. Kriegman's treatment on this subject is rather detailed for both curved objects and polyhedra. [31] One unsatisfactory aspect, however, is that the analysis is based on the stratified Morse theory, and consequently the functions involved have to be Morse. For curved objects (smooth all over), this means that his analysis and implementation would fail when the potential function contains points at which the Hessian matrix is singular. Thus the singularities discussed in the previous section, which in practice may arise due to design, are explicitly excluded. Also excluded are the coincidences such as co-planarity of more than three contact points and co-linearity of more than two contact points. Finally, piecewise-smooth objects are discussed in [31] but the direct approach does not suggest an efficient algorithm for dealing with such objects.

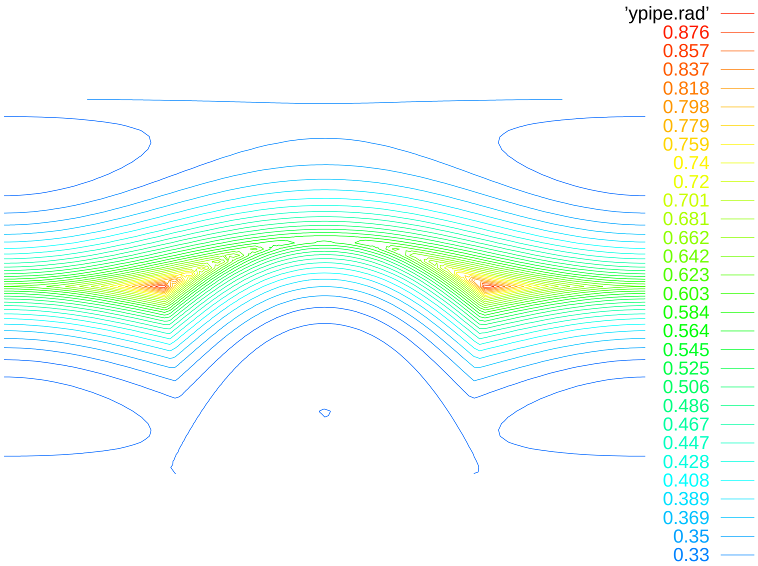

Figure 7.5: Equi-potential lines (in the parameter space) of the two-piece PVC pipe example.

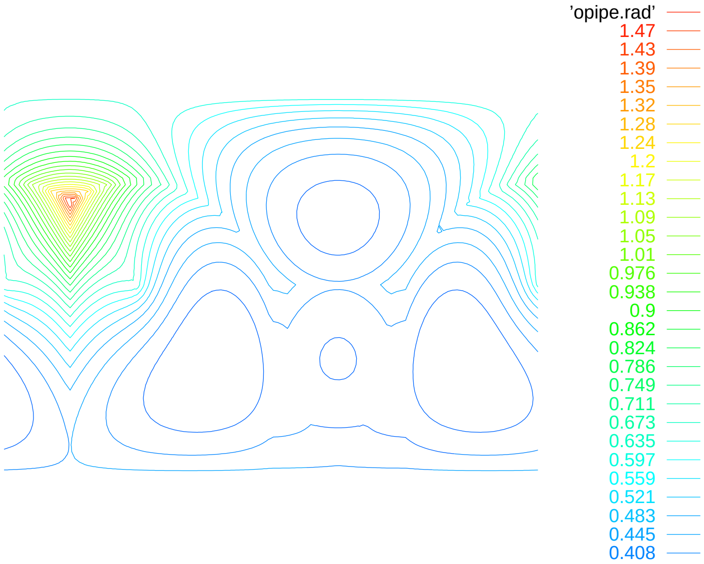

Figure 7.6: Equi-potential lines (in the parameter space) of the three-piece PVC pipe example.

Taking the dual approach, however, we are able to deal with all of the problems stated above, at least conceptually. Figures 7.5 and 7.6 show the equi-potential lines of the PVC pipe examples in Figures 7.1 and 7.2, respectively. (The parameterization used is spherical coordinates, not the algebraic change of variables in (7.3).) Finding the capture regions may be accomplished by working in the (s, t, r)-parameter space, using the algorithm in [31] with appropriate modifications to also process more singular saddles. Theoretically one may use the following asymptotically efficient algorithm which relies on the fact that all input surfaces are algebraic.

1. Find all saddle points, taking into account coincidences and singularities.

2. Find the contours passing the saddle points. Note that these are algebraic curves which intersect only at the computed saddle points and which need not be approximated or traced numerically as done in [31].

3. Apply a plane sweep algorithm to construct an arrangement consisting of these contours and possibly additional vertical lines.

4. For each peak $\mathbf{p}$ (corresponding to a stable pose) locate it in the arrangement. Do a breadth-first search of the adjacent faces starting from the face containing $\mathbf{p}$ in order to mark all faces in the arrangement which converge to $\mathbf{p}$.

If the input object consists of N features, then the number of smooth pieces which constitute the potential energy function in the parameter space is $O(N^{2+\epsilon})$ where the constant depends on the maximum degree. Thus there are at most (constant times) this many saddle points. The number of contour curves passing each saddle point is determined by the severity of coincidence and singularity. Each contour curve may consist of up to $O(N^{2+\epsilon})$ smooth pieces. It is then readily verified that the combinatorial complexity of this algorithm is $O(N^{4+\epsilon})$ since the plane sweep algorithm need not worry about intersecting contours.

7.4 Practical Considerations

We have largely left out the algebraic complexity in the previous discussions in our treatment of critical poses and capture regions. Yet in an actual implementation it is the algebraic computations that dominates the computation time as pointed out by Kriegman. The transformation introduced in Section 7.2 is algebraic in the relevant direction and does not increase the degrees of the dual surfaces by more than a factor of 5. (In fact a slightly modified variant can be used to achieve a factor of 4.) However, the dualization process may bring a degree D surface to a surface of degree up to D^4.

Since solving systems of equations is so costly, we might as well evaluate the potential energy all over the places, which can be realized by the ray-shooting technique introduced in Section 6.6. The angle between two rays in the dual space is the same as that between the two poses they determine in the primal space. The precision measure discussed there can be used as a parameter the user controls directly. If the precise poses have to be computed, the graph describing the potential energy function can be constructed from the mesh. The precise poses can then be computed using the nearby vertices as initial guesses to simple numerical techniques such as Newton's method, which is much less costly than the homotopy method. A missing feature in this context corresponds to a small capture region, which is likely what the user wouldn't want anyways.

Next let us focus on the special case in which all faces of K are quadratic surface patches bounded by planes. The edge case (typically a two-point contact) can be resolved by parameterizing two variables in the other one (as the solutions of quartic equations), substituting into $x^2+y^2+z^2$, and finally minimizing a univariate function with radicals. The vertex case (typically a three-point contact) requires us to solve a system of 3 quadratic or linear equations. It may be transformed to a root-finding problem for a univariate function as in the previous case; or one may apply Newton's method. In the latter case a good initial guess may be found by locating the octant, with respect to the normal forms of each surface, which contains the solution.

Now we consider the face case (one-point contact). We will derive the computation involved in the general case where the constant term of the quadratic equation does not vanish. The other case is similar but simpler and hence omitted. Let the host surface in question be given as

$$\{\mathbf{x} \, : \, f(\mathbf{x}) = 1 + 2\mathbf{b}^T\mathbf{x} + \mathbf{x}^T A\mathbf{x} = 0\}.$$

and let $g(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2$. Finally let

$$L(x_1, x_2, x_3, \lambda) = g(x_1, x_2, x_3) + \lambda f(x_1, x_2, x_3)$$

where λ is the Lagrange multiplier to be determined. Equating to zero the partial derivatives of L with respect to x_1, x_2, and x_3, we find that

$$(I + \lambda A)\mathbf{x} + \lambda \mathbf{b} = 0.$$

Expressing $\mathbf{x}$ in terms of λ and substituting it into the fourth partial derivative of L we obtain a polynomial of degree 6 in λ.

Thus in each of the three cases, the problem can be reduced to solving a single equation in one variable, albeit non-algebraic. Equivalently, the system in each case is algebraic of total degree at most 8, which is a significant improvement over the pessimistic estimate of 512 in [30].

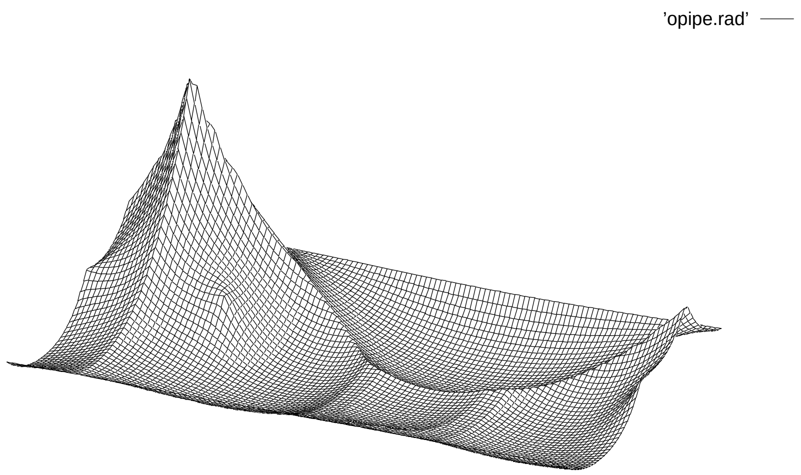

Figure 7.7: The radius function of the three-piece PVC pipe example.

Chapter 8

Width Functions and Tight Bounding Boxes

In this chapter we study two other applications of convex hulls of curved polyhedra. The ideas and techniques used in constructing the potential energy function of an object can be applied directly to constructing the width function of an object K. Once the width function of K is constructed, it may then be used to find the orientation and dimensions of the tight bounding boxes of K with minimum area, volume, or some other metric.

8.1 Width Function

Let K be a convex curved polyhedron whose interior contains the origin $\mathcal{O}$. Given a fixed direction represented by a vector $\mathbf{u}$, one may determine the distance between the pair of supporting planes of K whose normal vectors are parallel to $\mathbf{u}$. We now study the width function—this distance as a function of the given direction.

The need to compute width functions arise in a few applications. In an automated manufacturing environment where parts are fed to robotic manipulators, the width function of the planar part is essential to the construction of an oblivious grasping plan [36, 41, 11]. The special case in which the part is a planar polygon is studied in [20, 21] for use in such constructions for orienting parts using parallel jaw grippers. Another application is in the interpretation of the results obtained from a coordinate measuring machine (CMM), an instrument used to obtain the actual coordinates of sampled points on the manufactured surface designed to be flat. The distance between the closest pair of parallel planes enclosing the measured set of points serves as a good measure for "flatness" of the manufactured surface and can be compared with the tolerance.

To construct the width function of a curved polyhedron, we first find its polar set either by an ordinary cell construction algorithm or by ray shooting. Each straight line l passing the origin in the dual space represents a family of parallel planes in a fixed direction. It intersects ∂K^* at two points l_A and l_B lying on the opposite sides of the origin and representing an enclosing pair of supporting parallel planes of K. The width of K along the direction of l is therefore $1/|A_l| + 1/|B_l|$. Clearly the width function is non-smooth only at places where either A or B lies on an edge or coincides with a vertex of ∂K^*. Thus the decomposition (planar graph) of the domain of the "radius" function may be superimposed on a suitable mirror image of its own. The resulting planar graph gives a decomposition of the parameter space such that the width function is smooth and can be represented symbolically within each face. One convenient choice of such "radius" function is the potential energy function, although the choice of the origin is clearly irrelevant in constructing the width function. The two-dimensional analogs are similar and simpler. Figure 8.1 shows a general curved polygon in the plane, which is disconnected and non-convex, along with its convex hull. Its polar set is depicted in Figure 8.2. The width function is shown in Figure 8.3.

In the following table, we summarize the time complexity for constructing a planar graph to describe the width function of a curved polyhedron.

	planar, simple	planar, arbitrary	spatial
linear	$\Theta(N)$	$\Theta(N \log N)$	$O(N^2) \log N$
curved	$\Theta(N)$	$\Theta(N \log N)$	$O(N^{4+\epsilon})$

For the four planar cases, once the polar set is constructed, it takes only merging two lists whose elements signal the parameters corresponding to the non-smooth points of the width function. The time complexity is obviously optimal for the two simple sub-cases. It is also optimal for the two arbitrary sub-cases because the convex hull can be reconstructed in linear time given the width function. In the spatial linear case, constructing the polar set takes $O(N \log N)$. The complexity of the boundary of the point set of polyhedron, however, is linear. Thus the two copies of

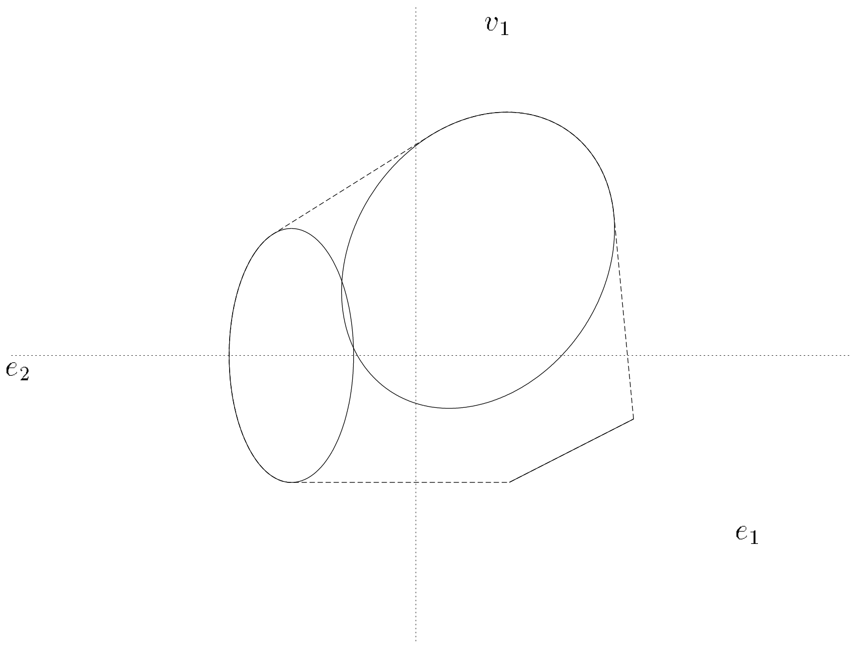

Figure 8.1: An "object" K consisting of two ellipses and a line segment; shown along with its convex hull.

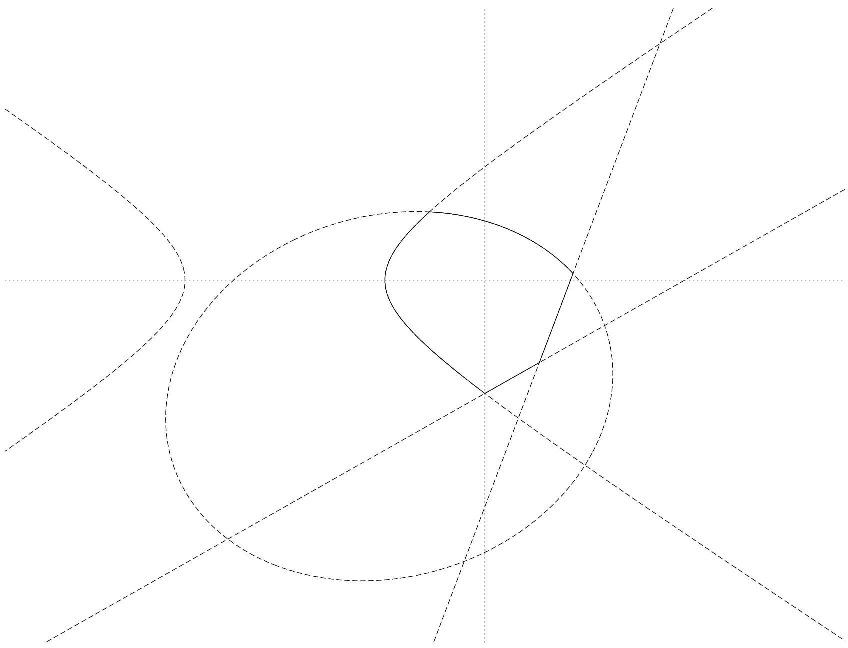

Figure 8.2: The dual of K. The solid line indicates the boundary of its polar set, K^*.

the boundary can be merged in quadratic time with a logarithmic overhead. For a curved polyhedron in space, the $O(N^{2+\epsilon})$ structural complexity results in the super-quartic running time of merging two planar subdivisions. Note that the complexity for a curved polyhedron is not completely general in that each 2-face of the input polyhedron is assumed to be bordered by a bounded number of edges. We finally mention that for an ordinary (non-curved) polyhedron, a faster algorithm exists if one only needs the overall width of the polyhedron and not a complete description of the width function.

8.2 Planar Tight Bounding Boxes

Given a curved polygon K in the plane, a parallel pair of supporting lines which enclose K can be thought of as a two-gon which bounds K tightly in a fixed direction. It is natural to extend the idea and consider the area, for example, of the box whose sides are parallel to a given direction and which tightly bounds K. The area is a piecewise smooth function of the the given direction. A complete description of this function may be constructed by superimposing four copies of the "radius" function similar to the potential energy function, each rotated by $\frac{\pi}{2}$ more than its predecessor. Figure 8.4 shows this function for the curved polygon in Figure 8.1. From it the tight bounding boxes of maximum and minimum area are found and depicted in Figure 8.5.

As is the case with the potential energy function, coincidences and singularities may happen. For example, we expect a bounding box to have generally four contact points with K, although with certain orientations the number of contact points may be more, or there may even be a line segment on K that coincides with one of the edges of the bounding box. Similarly, in order to decide whether the tight bounding box at a certain orientation is locally maximal or minimal, it may be necessary to look at higher derivatives of the area function if both the first and second derivatives vanish. We will not repeat a similar discussion that gives a catalog of the interesting situations in the primal space which correspond to coincidences or singularities in the dual space. It should be clear that by taking the dual approach, the problem of finding the smallest bounding box of a convex K with N is reduced to solving $O(N)$ instances of non-linear (smooth) optimization problems. It would be at best tedious

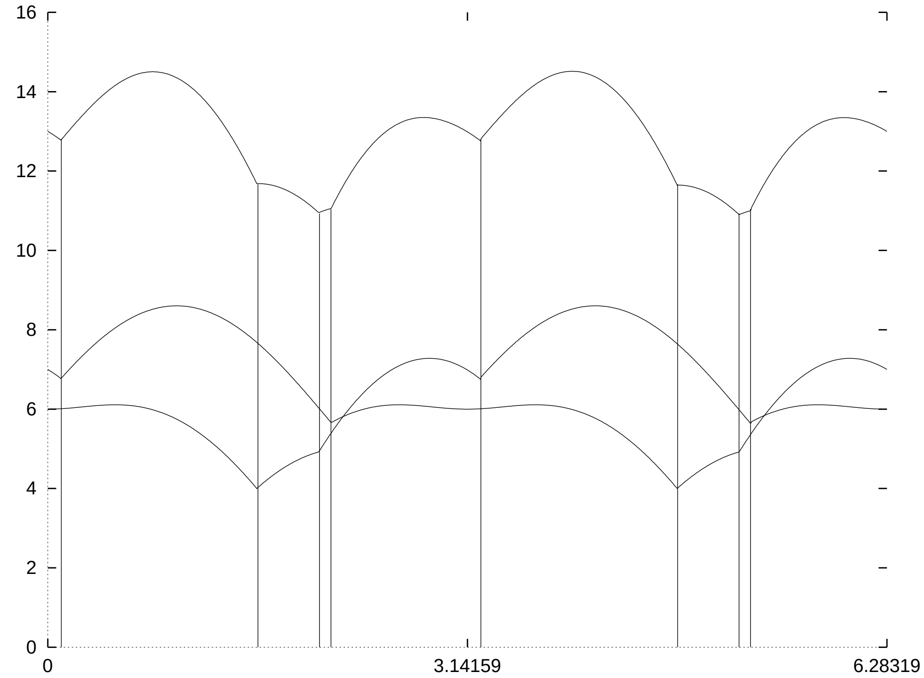

Figure 8.3: The only discontinuities in the first derivative of the width function occur at either those of the radius of K as a function of the polar angle, or those of the same function translated by π. (All three functions are shown.)

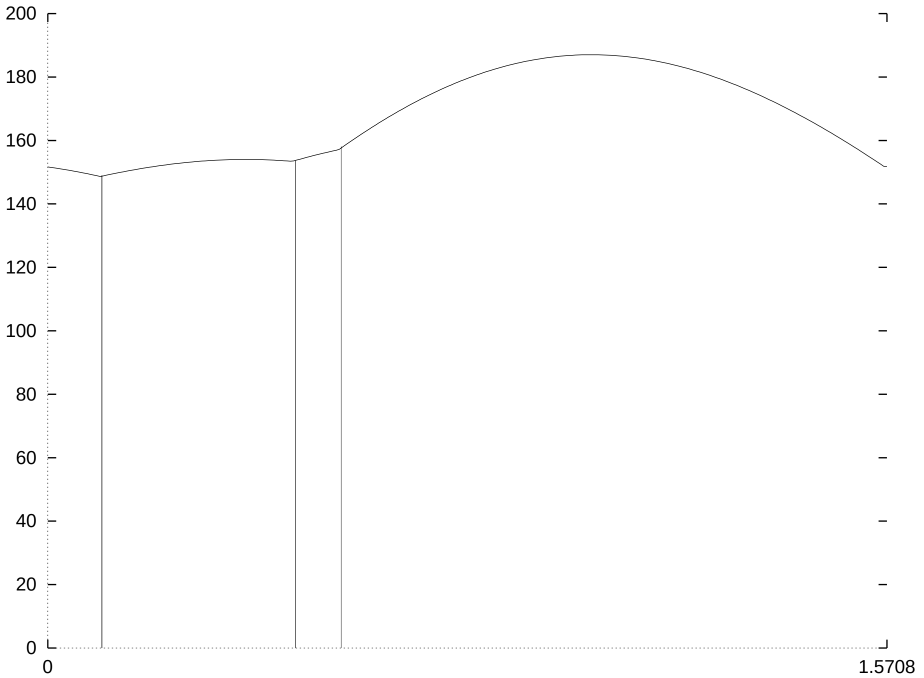

Figure 8.4: The area of the tightest bounding box as a function of the polar angle. Only one period (0 to $\pi/2$) is shown.

to enumerate all possible combinations which give the optimization constraints in the primal space.

Two generalizations and their meaningful combinations follow immediately. First, the optimality criterion may be chosen from a wide variety of possibilities such as minimal circumference, tight with maximal shortest edge, tight with minimal longest edge, etc. Each gives rise to a different optimization problem, but does not change the combinatorial structure of the problem. Secondly, the domain of optimization may be all k-gons whose inner angles in a circular order are the same as a given k-gon, or it may be a reasonable restriction of the above such as all k-gons similar to the given one. In particular minimal bounding squares can be found in the same order of time. In general for a k-gonal problem, the combinatorial complexity will be $O(N \log k)$ since the merge has to be performed $\log k$ times. Note however, that the increase in the algebraic complexity caused by considering a bounding k-gon instead of a bounding rectangle or triangle, may or may not be higher than just a factor of k, depending on the numerical method used to solve each instance of the optimization problem.

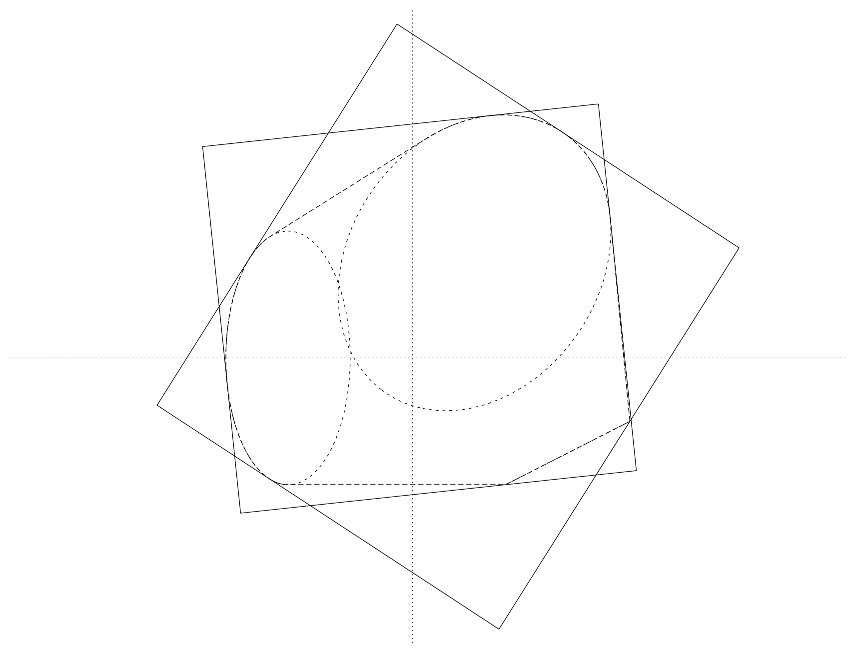

Figure 8.5: The tight bounding boxes of maximum and minimum area as found from the previous picture.

Chapter 9

Conclusions

9.1 Contributions

Combinatorial Complexity

In this thesis we have shown that the convex hull of a 3-D curved polyhedron has $O(N^{2+\epsilon})$ faces and can be constructed in randomized expected time $O(N^{2+\epsilon})$, provided each face of the curved polyhedron has a bounded number of subfaces. The graph of the potential energy function of such an object can be decomposed into $O(N^{2+\epsilon})$ smooth pieces and can be constructed in randomized expected time $O(N^{2+\epsilon})$. The capture regions of a 3-D curved polyhedron has a structural complexity of $O(N^{4+\epsilon})$ and can be constructed in deterministic time $O(N^{4+\epsilon})$. Similarly, the width function of a 3-D curved polyhedron has a structural complexity of $O(N^{4+\epsilon})$ and can be constructed in deterministic time $O(N^{4+\epsilon})$.

Two operations with respect to the convex hull of an arbitrary 3-D curved polyhedron can be performed more easily and efficiently without explicit construction of the convex hull. The classification of a point again the convex hull of a 3-D curved polyhedron takes time $O(N \log N)$. Given a curved polyhedron and a point outside its convex hull, the supporting cone from the point to its convex hull has a structural complexity of $\Theta(N)$ and can be constructed in time $O(N \log N)$.

The convex hull of a 2-D curved polygon can be found in time $\Theta(N \log N)$.[1] The minimal bounding boxes of a 2-D curved polygon can be found in time $\Theta(N)$.

[1] Recall that a curved polygon may consist of more than one connected component and may have dangling 1-faces and/or 0-faces.

Algebraic Complexity

The new faces in constructing the convex hull of a curved polyhedron made of quadratic surface patches with planar cut have closed form parameterizations. The critical poses of such a curved polyhedron can be found by solving systems of equations each of total degree at most 8.

Miscellaneous

It is illustrated by an implementation that numerical approximations of most of the above computations can be efficiently (but not optimally in the combinatorial sense) and safely carried out without resorting to sophisticated coincidence manipulation strategies. (The implementation does not suggest how to approximate capture regions efficiently.) The existing optimal algorithms for constructing the convex hull of a simple planar polygon, curved or rectilinear, are significantly simplified. The existing discussions of the stable pose and capture region problems are improved by giving a case analysis of coincidences and singularities complete up to a fixed degree. A simple `C++` interface is proposed for the safe and efficient manipulations of submatrices without maintaining reference counts.

9.2 Future Research

Some of the results concerning curved polyhedra are valid only if the curved polyhedra satisfy the extra constraint that each face has a bounded number of subfaces. The algebraic complexity of the duals of higher degree surfaces and curves is worth studying considering the dramatic difference between the naive estimate of the degree 512 system of equations versus the tight bound of 8 after careful analysis of quadratic surfaces using the dual approach. There is room either for improving the algorithmic complexity or for establishing a worst case scenario in the convex hull of a curved polyhedron and its potential energy function. This is also the case with point classification and supporting cone construction without explicit hull construction, and especially with the width function and capture region.

The prototype implementation allows us to evaluate the dual approach. Two possible extensions are either enhancing the implementation and make it interface with existing solid modelers, or incorporating a numerical solver for dealing with systems of equations.

Reference List

[1] P. K. Agarwal, M. Sharir, and P. Shor. Sharp upper and lower bounds on the length general Davenport-Schinzel sequences. *Journal of Combinatorial Theory, Series A*, 52:228–274, 1989.

[2] Pankaj K. Agarwal. *Intersection and Decomposition Algorithms for Planar Arrangements*. Cambridge University Press, 1991.

[3] Amitabh Agrawal. *A General Approach to the Design of Robust Algorithms for Geometric Modeling*. PhD thesis, Computer Science Department, University of Southern California, 1994.

[4] D. S. Arnon and B. Buchberger. *Algorithms in Real Algebraic Geometry*. Academic Press, 1988.

[5] Chanderjit Bajaj and Myung-Soo Kim. Convex hull of objects bounded by algebraic curves. *Algorithmica*, pages 533–553, 1991.

[6] J. D. Boissonnat, Cérézo, O. Devillers, J. Duquesne, and M. Yvinec. An algorithm for constructing the convex hull of a set of spheres in dimension d^*. In *Proceedings of the Fourth Canadian Conderence on Computational Geometry*, August 1992.

[7] J. W. Bruce and P. J. Giblin. *Curves and Singularities*. Cambridge University Press, 2nd edition, 1992.

[8] Bernard Chazelle. An optimal convex hull algorithm and new results on cuttings. In *Proc. 32nd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 29–38, 1991.

[9] Bernard Chazelle. An optimal algorithm for intersecting three-dimensional convex polyhedra. *SIAM Journal on Computing*, 21(4):671–696, August 1992.

[10] Bernard Chazelle, Herbert Edelsbrunner, Leonidas Guibas, and Micha Sharir. Diameter, width, closest line pair, and parameteric searching. *Discrete Computational Geometry*, pages 183–196, 1993.

[11] Yuibin Chen, Chao-Kuei Hung, and Doug Ierardi. Optimal manipulation strategies for orienting planar workpieces by pushing. *submitted to Algorithmica*, 1994.

[12] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, (ii). *Discrete Computational Geometry*, 4:387–421, 1989.

[13] David P. Dobkin and Diane L. Souvaine. Computational geometry in a curved world. *Algorithmica*, pages 421–457, 1990.

[14] J. J. Dongarra, R. Pozo, and D. W. Walker. Lapack++: A design overview of object-oriented extensions for high performance linear algebra. Technical report, University of Tennessee, 1993. software available at ftp://netlib.att.com/att/c++/lapack++/.

[15] H. Edelsbrunner, L. Guibas, J. Pach, R. Pollack, R. Seidel, and M. Sharir. Arrangements of curves in the plane: topology, combinatorics, and algorithms. *Theoretical Computer Science*, 92:319–336, 1992.

[16] Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer-Verlag, 1987.

[17] Michael C. Gemignani. *Elementary Topology.* Dover, 1990. Republication of the second edition (1972) of the work originally published in 1967 by Addison-Wesley.

[18] Robert Gilmore. *Catastrophe Theory for Scientists and Engineers.* Dover, 1981.

[19] A. Goetz. *Introduction to Differential Geometry.* Addison-Wesley, 1970.

[20] K.Y. Goldberg. A complete algorithm for orienting polygonal parts. *Algorithmica*, 1993.

[21] K.Y. Goldberg and A. Rao. Orienting generalized polygons. Technical Report 275, USC Institute for Robotics and Intelligent Systems, 1991.

[22] R. L. Graham and F. F. Yao. Finding the convex hull of a simple polygon. *Journal of Algorithms*, 4(4):324–331, 1983.

[23] Joos Heintz, Marie-Francoise Roy, and Pablo Solernó. Description of the connected components of a semialgebraic set in single exponential time. *Discrete and Computational Geometry*, pages 121–140, 1994.

[24] Roger A. Horn and Charles R. Johnson. *Matrix Analysis.* University of Cambridge, 1985.

[25] Doug Ierardi and Dexter Kozen. *Synthesis of Parallel Algorithms*, chapter 15. Morgan Kaufmann, 1993.

[26] Alfred Inselberg. Personal communication.

[27] Bill Jones, Hebert Edelsbrunner, and Guibas et al. On-line bibliography of computational geometry. anonymous ftp from `cs.usask.ca` in file `pub/geometry/geombib.tar.Z`.

[28] Andrew Klapper. A lower bound on the complexity of the convex hull problem for simple polyhedra. *Information Processing Letters*, 25:159–161, 1987.

[29] D. J. Kriegman and J. Ponce. On recognizing and positioning curved 3d objects from image contours. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, November 1990.

[30] David J. Kriegman. Computing stable poses of piecewise smooth objects. In *Computer Vision Graphics Image Processing: Image Understanding*, 1991.

[31] David J. Kriegman. Let them fall where they may: Capture regions of curved objects and polyhedra. Technical Report 9508, Yale University, June 1995. Submitted to the International Journal of Robotics Research.

[32] David J. Kriegman, Erliang Yeh, and Jean Ponce. Convex hulls of algebraic curves. In *SPIE Conference on Curves and Surfaces in Computer Vision and Graphics*, November 1992.

[33] Steven R. Lay. *Convex Sets and Their Applications.* John Wiley & Sons, Inc., 1992. original edition 1982.

[34] D. T. Lee. On finding the convex hull of a simple polygon. *International Journal of Computing and Information Science*, 12(2):87–98, 1983.

[35] Dinesh Manocha. Solving polynomial systems for curves, surface, and solid modeling. In *Proc. of ACM/SIGGRAPH*, 1993.

[36] M.T. Mason, K.Y. Goldberg, and R.H. Taylor. Planning sequences of squeeze–grasps to orient and grasp polygonal objects. Technical Report CMU–CS–88–127, Carnegie Mellon University, Computer Science Department, April 1988.

[37] D. McCallum and D. Avis. A linear algorithm for finding the convex hull of a simple polygon. *Information Processing Letter*, 9:201 – 206, 1979.

[38] James R. Miller. Incremental boundary evaluation using inference of edge classifications. *IEEE Computer Graphics and Applications*, pages 71–78, January 1993.

[39] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 1985.

[40] Anil Rao, David Kriegman, and Ken Goldberg. Complete algorithms for feeding polyhedral parts using pivot grasps. 1994. Submitted to IEEE Transactions on Robotics & Automaton.

[41] Anil S. Rao and Kenneth Y. Goldberg. Manipulating algebraic parts in the plane. In *IEEE Transactions on Robotics & Automaton*, 1993.

[42] Aristides A. G. Requicha and Herbert B. Voelcker. Boolean operations in solid modeling: Boundary evaluation and merging algorithms. In *Proceedings of the IEEE*, 1985.

[43] Alejandro A. Schäffer and Christopher J. Van Wyk. Convex hulls of piecewise-smooth jordan curves. *Journal of Algorithms*, 8:64–94, 1987.

[44] Otto Schreier and Emanuel Sperner. *Projective Geometry of n Dimensions*. Chelsea Publishing Company, New York, 1985.

[45] J.T. Schwartz and M. Sharir. On the two dimensional davenport-schinzel problem. *Journal of Symbolic Computation*, 10:371–393, 1990.

[46] Micha Sharir. Almost tight upper bounds for lower envelopes in higher dimensions. In *Symposium on Foundations of Computer Science*, pages 498–507, 1993.

Appendix A

Implementation Notes

We have shown that asymptotically efficient algorithms exist for solving many of the problems treated in this thesis. In reality, however, some of these algorithms are either too complicated to implement or simply incur so much algebraic overhead that it dominates the running time. In most cases, the algebraic complexity also forces the implementation to face coincidences and singularities seriously. An implementation based on these "efficient" algorithms thus may not be able to enumerate a small number of such special cases and explicitly treat them separately. Instead, it must adopt some coherent strategy such as that proposed in [3].

Such limitation, however, is not intrinsic to the dual approach. We now briefly describe a naive and yet very efficient ray-shooting implementation based on the dual approach. It is most useful as a tool for the visualization of the concepts discussed in this thesis, and for use in combination with existing tools which solve (smooth) optimization problems numerically given good initial guesses. We also discuss a `C++` technique we adopted in our implementation which is less relevant to the algorithmic issues but is nevertheless very interesting.

A.1 Summary

Platforms and Compiler

The program is written in `C++`. We have compiled and tested it on an IBM PC with i486DX33 processor running Linux and OS/2. The compiler used is `gcc` in both cases.

Dependency

The following libraries/packages are used directly or indirectly by the program.

LAPACK The linear algebra package is linked with the program.

Scheme This lisp dialect is used to pre-process the input. We only use the Linux version (`snow/stk`) to actually pre-process the input. Its output is used on other platforms to test the program.

Gnuplot This plotting package is used to post-process the mesh file created by our program and to create postscript pictures.

The following packages are not needed to compile or run our program but they were indispensable tools during the development of the program.

RLaB This linear algebra calculator is first used to create a prototype implementation for our algorithm and later used to verify the correctness of our matrix implementation.

PARI/gp We use this symbolic calculator mostly for polynomial computations.

A.2 Object Specification

Input to the program are surface patches specified in LISP syntax. The following is a typical description of a surface patch.

```
(
(class          2)
(scale          #(1.2 1.2 0))
(rot_type       ezyx)
(rot_angle      #(0 90 0))
(center         #(-0.1 -1.1 1.1))
(u_range        #(-180 180))
(v_range        #(0 1.5))
)
```

The meanings of these fields are explained as follows.

class Quadratic surfaces are classified according to their normal forms as in [44]. The possible values are:

- **0** Cone or plane.
- **1** Parabolic cylinder, hyperbolic paraboloid, or elliptic paraboloid.
- **2** Hyperbolic cylinder, elliptic cylinder, hyperboloid of two sheets, hyperboloid of one sheet, or ellipsoid

scale Scaling factors along the three principle axes of the surface.

rot_type Name of the rotation scheme.

rot_angle Rotation parameters to bring the surface from the standard orientation to the desired orientation.

center Translation to bring the surface from the standard position to the desired position.

u_range The range of the longitudinal parameter.

v_range The range of the latitudinal parameter.

The description scheme begins with a representative quadratic surface. In our example, a class 2 surface with two positive axis scales and one vanishing axis scale is interpreted as an infinite cone whose axis coincides with the Z-axis. It happens to be a circular cylinder with a 1.2 radius. This cylinder is first rotated using the Euler-Z-Y-X angle specification and its axis become aligned with the X-axis. Its center is then moved to $(-0.1 - 1.11.1)$. Finally a patch is cut out from the infinite cylinder as our desired input. The longitudinal parameter specifies the range, in degrees, of the angular parameter around the axis; and the latitudinal parameter specifies the extent in the Z-direction.

A.3 Functionalities and Limitations

The program uses ray shooting in the dual space to find the boundary of the polar set. Given an object description file as input, it may output any combination of the following information.

mesh of the object containing regularly sampled points on the input surface patches

mesh of the boundary of the polar set containing coordinates of points on ∂K^* which are sampled along fixed intervals of longitudes and latitudes

mesh of the radius as a function of the longitudes and latitudes

mesh of the boundary of convex hull in their Euclidean coordinates.

The resolution of the mesh can be specified, although it only has a first-order effect on the output.

Currently not all legal inputs to the program are supported. Specifically, the following features are not implemented:

1. cones, planes, hyperboloids, paraboloids, and all parabolic surfaces;
2. cut along longitudinal lines;
3. cut along non-parameter lines;
4. program-chosen origin;

All of these features can be added without much difficulty except planes and planar cut along non-parameter lines. The difficulty associated with these two cases is unrelated to the dual approach. In the primal space it is more tedious to create a satisfactory mesh for a planar patch bounded by conic sections. The same is true with surface patches cut along non-parametric lines.

The simplicity of the dual approach allows our implementation to take a naive strategy for dealing with coincidences. A macro `nearly_zero` is defined and used consistently in all situations where there may be numerical difficulties. Without compiler or manual optimization, it takes about 20 seconds on a 486DX33 running Linux to produce three (the dual, the radius, and the hull) 58 by 29 meshes for the pipe example consisting of six surface patches and ten edges.

A.4 Submatrices: to Copy or to Reference?

One of the efficiency issues most commonly raised against the use of C++ is the automatic invocation of copy constructors for function arguments as well as for their return values. Consider the following simple piece of code for matrix computation:

```
MatrixDouble A, B, C;
...
C = A * B;
```

If the matrix operators are not properly defined, there can be as many as four ephemeral matrices which are destroyed almost as soon as they are created. A good design of the class provider will result in the creation and destruction of just one temporary matrix. Such cost can be reduced to minimum in a series of matrix computations if the class provides a set of destructive operators/functions which return a reference to one of its arguments, and if the class user makes a judicious use of these operators. Thus, for example, an expression like

```
A *= B += C /= D;
```

will perform three operations without creating any ephemeral objects.

Our discussion is based on the assumption that we have agreed upon such design and use of the matrix class. The question we would like to address is: how do we deal with submatrices efficiently? Do we make a copy of the submatrix and create a new matrix object, or do we just create a reference to the original matrix? The first alternative is obviously very inefficient. It is not very pleasant to implement the second alternative, nor is it clear how one should handle user expressions like

```
A.sub(0,0,n/2,n) += A.sub(n/2+1,0,n,n);
```

in which both arguments involve references to the same matrix. The Lapack++ library [14] basically takes the first alternative but also allows the user to create matrices which represent frequently used references to the submatrices. This is not very satisfactory as the class provider still has to maintain a reference count and the user still suffers from performance penalty when dealing with submatrices which are used only once.

We propose a submatrix interface which

- avoids unnecessary copying of the matrix entries;
- does not require the class provider to maintain reference counts;
- allows nested expressions to contain references to the same matrix; and yet
- does not create garbage in the free store.

We begin with the declaration of the reference class which has little data content but provides almost all of the matrix functions:

```
class MatDblRef { // reference to submatrix
friend class MatrixDouble;
protected:
    MatrixDouble & host;
    int _sr0, _sc0, _snr, _snc;
        // location and size of the submatrix referenced by *this
    MatDblRef(MatrixDouble *h, int i, int j, int nr, int nc);
    void operator =(MatDblRef const & A) { assert(0); }
    ...
public:
    MatDblRef & pre_mul_eq(MatDblRef const & A, char t = 'n');
    MatDblRef & post_mul_eq(MatDblRef const & A, char t = 'n');
    ... // more arithmetic functions and operators
};
```

An object belonging to this class is but a reference to part of a true matrix (`host`) containing the actual array of double values. Moreover, ordinary users are not even allowed to create such an object since the only constructor is protected. However, all of the ordinary matrix operations one would want to use are defined in terms of objects of this class.

How could the user use such an object if s/he cannot even create one? The obvious solution is to create a class publicly derived from `MatDblRef`. The bulky content of an ordinary matrix—the array of doubles is also defined in this class.

```
class MatrixDouble : public Matrix<double>, public MatDblRef {
```

```
friend class MatDblRef;
public:
    // constructors
    MatrixDouble(int m, int n) : Matrix<double>(m,n),
        MatDblRef(this, 0, 0, m, n) {}
    MatrixDouble(MatDblRef const & src) : Matrix<double>(src._snr, src._snc),
        MatDblRef(this, 0, 0, src._snr, src._snc) { copy(src); }
    // submatrix access
    MatDblRef sub(int i, int j, int nr, int nc)
        { return MatDblRef(this, i,j,nr,nc); }
    MatDblRef rows(int i, int nr=1) { return sub(i,0,nr,_ncols); }
    MatDblRef cols(int j, int nc=1) { return sub(0,j,_nrows,nc); }
};
```

Now an object of class `MatrixDouble`, once created, is also a reference to the entirety of itself, forever (that is, until it dies). Thus an ordinary user may use all of the arithmetic functions/operators on a `MatrixDouble` object defined in the parent class as if there were just one ordinary matrix class.

The interesting mechanics begins to work when the user wishes to use submatrices, either as the source or as the destination of an operation. The member functions of `MatrixDouble` for accessing submatrices creates objects of class `MatDblRef` whose `host` is the `MatrixDouble` object in question. The user may use these submatrices/references freely in those arithmetic functions/operators defined for `MatDblRef` in a complicated expression without incurring more overhead than copying 4 integers and one pointer (by which the reference `host` is implemented) each time. If the user tries to save the result of the final expression, however, a copy will be created out of necessity, by the second constructor in `MatrixDouble`.

Thus the user retains the option of writing efficient expressions without compromising the integrity of the system. With our design, there are other subtleties which must be handled carefully, but they are pretty much standard `C++` pitfalls and techniques and hence omitted from our discussion.