Tangential Distance Fields for Mesh Silhouette Analysis

Matt Olson and Hao Zhang

GrUVi Lab, School of Computing Science, Simon Fraser University, Canada
{matto, haoz}@cs.sfu.ca

Abstract

We consider a tangent-space representation of surfaces which maps each point on a surface to the tangent plane of the surface at that point. Such representations are known to facilitate the solution of several visibility problems, in particular, those involving silhouette analysis. In this paper, we introduce a novel class of distance fields for a given surface defined by its tangent planes. At each point in space, we assign a scalar value which is a weighted sum of distances to these tangent planes. We call the resulting scalar field a tangential distance field *or TDF. When applied to triangle mesh models, the tangent planes become supporting planes of the mesh triangles. The weighting scheme used to construct a TDF for a given mesh and the way the TDF is utilized can be closely tailored to a specific application. At the same time, the TDFs are continuous, lending themselves to standard optimization techniques, such as greedy local search, thus leading to efficient algorithms. In this paper, we use four applications to illustrate the benefit of using TDFs: multi-origin silhouette extraction in Hough space, silhouette-based view point selection, camera path planning, and light source placement.*

Categories and Subject Descriptors (according to ACM CCS): *I.3.5 Computational Geometry and Object Modeling — Geometric algorithms, languages, and systems; Hierarchy and geometric transformations*

1. Introduction

The majority of geometric problems in computer graphics deal with surfaces. Typically, a surface is specified by points lying on it. To obtain a discrete model, a surface is first sampled and then the sample points are connected into a piecewise linear triangulation, a triangle mesh, which tessellates the surface. A lesser known surface representation is via the tangent planes associated with points of the surface. Such a tangent-space surface representation is sometimes referred to as the Blaschke image in Laguerre geometry [PPS03]. It has also been referred to as the *surface dual* [Joh04, Mor02, SRKP04] in the literature. However, to avoid confusion with the classical geometric dual transform [PDB*01], we do not adopt that terminology. When dealing with triangle meshes, the tangent planes are replaced by the supporting planes of the mesh triangles.

Certain geometric properties are easier to deal with via tangent-space representations. For example, two faces on a mesh which are geodesically far away from each other can have the same or similar supporting planes, making them geometrically close in the tangent space. The importance of supporting planes can be realized when studying visibility since these very planes, when crossed by the viewpoint, trigger visibility events; see Figure 1 for an illustration.

Tangent-space surface representations are not easy to visualize or utilize directly. However, one may derive useful information from them to benefit specific applications. For example, with a designated origin, each point P on a surface can be mapped to a point $H(P)$ which is the orthogonal projection of the origin onto the tangent plane at P; see Figure 1(d) for an illustration in the 2D case. This is called the *Hough transform* [Hou59] and the image of the mapping is called a *pedal surface* [Joh04]. The Hough transform, or its close relative, the dual transform, has been exploited to facilitate silhouette extraction [HZ00, PDB*01, OZ06] or back face culling [KMGL96] over polygonal models.

In this paper, we explore beyond low-level representations derived from tangent-space surface representations, such as a point cloud resulting from Hough transform. We introduce a novel class of scalar fields, constructed as a weighted sum of signed distances to the set of tangent planes or supporting planes, in the case of triangle mesh models. Referred

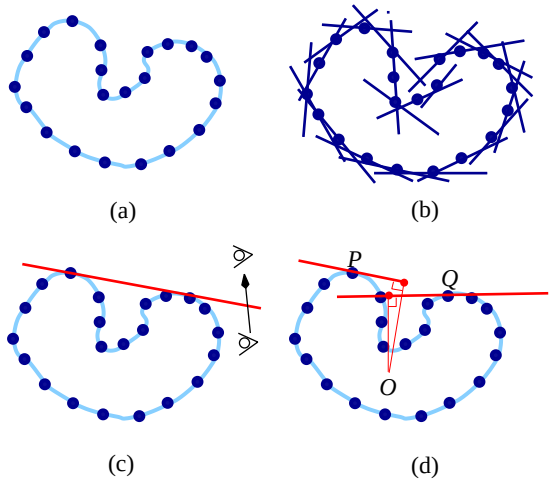


Figure 1: *A 2D shape given by points along its boundary (a) and its representation using (a subset of) tangent planes in (b). In (c), if the viewpoint crosses a bi-tangent plane (red line), there is a visibility event — part of the concave region becomes visible. In (d), two points P and Q geodesically distant from each other have similar tangent planes (red lines). Thus their Hough transform (red dots) are close to each other, where O is an arbitrarily chosen origin.*

to as *tangential distance fields*, or *TDFs*, these scalar fields capture higher-level information about the surface geometry and lend themselves to more global geometric problems, e.g., finding a viewpoint which maximizes the length of a mesh's silhouette. Also, they allow for standard optimization techniques, e.g., greedy search. One can intuitively regard a method which utilizes the TDFs as a process of *voting*. Each plane casts a vote to each point in space and the votes are weighted appropriately to serve a particular application.

All the applications considered in our work are related to mesh silhouette analysis. Section 3 explains the close tie between mesh silhouettes and supporting planes, as well as more details on tangent-space surface representations and TDFs. A point selection scheme based on TDFs is described in Section 4, outlining the algorithmic aspect of our general approach. How different voting schemes can be designed to fit specific applications is the subject of Section 5 and these specific applications are covered subsequently.

First, we develop a method which assigns the faces of a mesh to *optimized origins* in order to accelerate Hough-space silhouette extraction. The key observation is that optimizing for the number and positioning of multiple Hough-space origins can drastically improve silhouette extraction performance. We accomplish this by minimizing mesh silhouettes from the origins, while maintaining a sufficiently compact point distribution, as explained in Section 6. These criteria can be concisely expressed by a TDF and the problem is then reduced to iterative peak detection.

The problem of *viewpoint selection* can also be phrased with respect to properties of mesh supporting planes. In this case, we attempt to maximize both the length of the silhouette and the number of faces orthogonal to the viewpoint, as explained in Section 7. Again, we can approximate these criteria by a well-chosen search space and TDF. We demonstrate results which compare favorably to those presented in recent works on viewpoint selection. We also study a closely related problem, that of *camera path planning*, and show how the same TDF can be utilized.

Finally, given a viewpoint, we can use an appropriate TDF to intelligently place light sources, as explained in Section 8. The light source positions chosen tend to minimize unlit areas facing the viewpoint, accomplished through minimizing the length of the silhouette from the light source. At the same time, the light sources attempt to maximize variation in intensity by choosing a sharp angle of incidence.

Our contributions can be summarized as follows:

- We introduce the novel concept of tangential distance fields (TDFs) to further exploit the usefulness of tangent-space surface representations for visibility problems.
- We show how TDFs can be tuned appropriately to tackle a diversity of problems, including multi-origin Hough-space silhouette extraction, viewpoint selection, camera path planning, and light source placement. We expect this to only be a partial list of potential applications of TDF.

2. Related Work

Tangent-space representation of smooth surfaces has been studied in the field of algebraic geometry [Har92]. A related concept from convex geometry is that of the *support function* of a surface, defined as the dot product $u(P) = P \cdot N(P)$, where P is a point on the surface and $N(P)$ is the unit surface normal at P. This has been adopted by Sir *et al.* [SJGar] to generate surface patches with elegant properties, including seamless support for surface offset.

In computer vision, more specifically for recognition of curved objects, a close relationship between the tangent-space representation of a surface and the tangent-space representation of its silhouette in weak-perspective images has been exploited to construct shape signatures [Joh04]. Hoppe et al. [HDD*92] use a set of tangent planes to approximate the surface that is to be reconstructed from a given point cloud. A scalar distance field which approximates the distance transform with respect to the target surface is computed to enable the use of marching cubes. The distance transform, measuring the closest distances from points to a given surface, is the most well-known scalar field defined for surfaces and it has many applications, including visibility analysis [TCO*05]. A thorough coverage on this subject can be found in the recent survey by Jones et al. [JBS06].

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.

Most relevant to our work in geometry processing are those which exploit the close tie between the supporting planes of a mesh and certain visibility events, e.g., silhouette change. This is the idea behind the use of Hough and dual transforms for silhouette extraction [HZ00, PDB*01, OZ06]. Other uses of 3D Hough transforms include linear approximation of signed distance fields [WK03], billboard cloud generation [DDSD03], recognition and reconstruction of surfaces [PPS03], and mesh retrieval [ZP01]. The dual transform has also been utilized for back-face culling [KMGL96].

While Hough and dual transforms are discrete representations derived from tangent-space surface representations, we are not aware of any other derived constructs, e.g., a scalar distance field, as introduced in this paper. Next, we discuss previous works relevant to the applications we study.

Silhouette extraction: Object-space silhouette detection under perspective projection is well studied. Many methods make use of mesh supporting planes under a plane-point transform to accelerate processing. Hertzmann and Zorin [HZ00] implement a dual-space approach in 4D homogeneous space. Pop et al. [PDB*01] use the classical point-plane duality in 3D projective space for efficient silhouette updates. In previous work [OZ06], we advocate the use of 3D Hough transform of mesh faces, organized into a *single* octree, for efficient update and initial extraction of silhouette. Compared to the dual transform [PDB*01], the Hough transform typically leads to more uniform point distributions and thus much improved performance. In this paper, we extend our past work by optimizing for the number and positioning of multiple origins in Hough-space mesh representations. Experimentally, we obtain between 2-3 times speed-up over [OZ06] in initial silhouette extraction.

Viewpoint selection and camera path planning: The viewpoint selection problem has applications ranging from image-based modelling [VFSH01] to object recognition [YSY*06] and is also studied in robotics as the sensor placement problem [TL95]. Most approaches define a view quality metric and evaluate it over a set of candidate viewpoints, essentially via an exhaustive search, choosing the candidate with the highest score as the best view. An overview of these approaches is provided by Polonsky *et al.* [PPB*05]. Recently, reflective symmetry information has been used to select single high-quality viewpoints [PSG*06]. Our approach using TDFs provides comparable or better quality viewpoints and at reduced costs, as a TDF allows for more efficient viewpoint search.

Camera-path planning is a related problem. It has been studied in the context of motion planning and path finding [AVF04], as well as in digital cinematography [CO06]. We consider camera-path planning in the same context as viewpoint selection – selecting a camera path which conveys as much information as possible about a given model based again on silhouettes. Our method for solving the viewpoint selection problem leads to an elegant and straightforward method for planning such a camera path.

Light source placement: Light source placement is generally studied alongside the viewpoint selection problem, and has applications ranging from edge-based object recognition [CM93] to perceptual scene enhancement [Gum02]. As with the viewpoint selection problem, light source placement algorithms generally define a quality metric and evaluate it over a set of candidate positions. Quality metrics can be quite similar to those for viewpoint selection. For example, [Gum02] and [Vaz06] use an entropy measure similar to [VFSH01]. Unlike viewpoint selection, however, the amount of information conveyed by a given light source placement can be affected by the light's illumination parameters (diffuse and specular values, and shininess exponent), and placing multiple light sources can give drastically different results than each light taken alone.

3. Tangential distance fields

Consider a smooth surface Σ embedded in $\mathbb{R}^3$. At any point P on Σ, we compute a tangent plane $\pi(P)$ of Σ. The set of tangent planes $\mathcal{T}(\Sigma) = \{\pi(P) : P \in \Sigma\}$ is called the tangent-space representation of Σ; in fact, $\mathcal{T}(\Sigma)$ form a surface in 4-D space. In the reminder of this paper, we shall only consider tangent-space representations of triangle meshes and their derived constructs. Naturally, the tangent planes are now replaced by supporting planes of the mesh faces (triangles). The mapping from mesh faces to supporting planes is not injective in general: many faces can share the same tangent plane. This is in fact one of the strengths of the tangent-space representation for geometry processing. For example, it allows us to identify points on the same silhouette, even though they may be arbitrarily distant on the surface.

There is an obvious tie between mesh silhouettes and supporting planes. Recall that a mesh edge is on the silhouette if and only one of its adjacent faces is front-facing and the other is back-facing. It follows that the silhouette status of the edge changes precisely when the viewpoint crosses one of the face supporting planes. Thus it is not surprising that constructs derived from the supporting planes, e.g., Hough transform, can be utilized for silhouette analysis.

We define a scalar field in which every point in space is given a value as a weighted distance to the supporting planes. We call this a *tangential distance field* or TDF. Specifically, let M be a triangle mesh whose set of supporting planes is given by $\mathcal{T}(M)$, then for $P \in \mathbb{R}^3$, the TDF value at P is,

$$\mathcal{D}(P, \mathcal{T}(M)) = \sum_{i=1}^{|\mathcal{T}(M)|} f\big(\text{dist}(\pi_i, P)\big), \qquad (1)$$

where $\pi_i \in \mathcal{T}(M)$, f is a weight function which we refer to as the *support distance function* or *SDF*, and $\text{dist}(\pi_i, P)$ is the *signed* point-to-plane distance between P and π_i. The SDF

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.

f is critical to the meaning of the TDF. It specifies the voting scheme and should be chosen on an application-specific basis. Once we have defined the TDF, we can use it to select points of interest around a mesh.

4. Point selection scheme

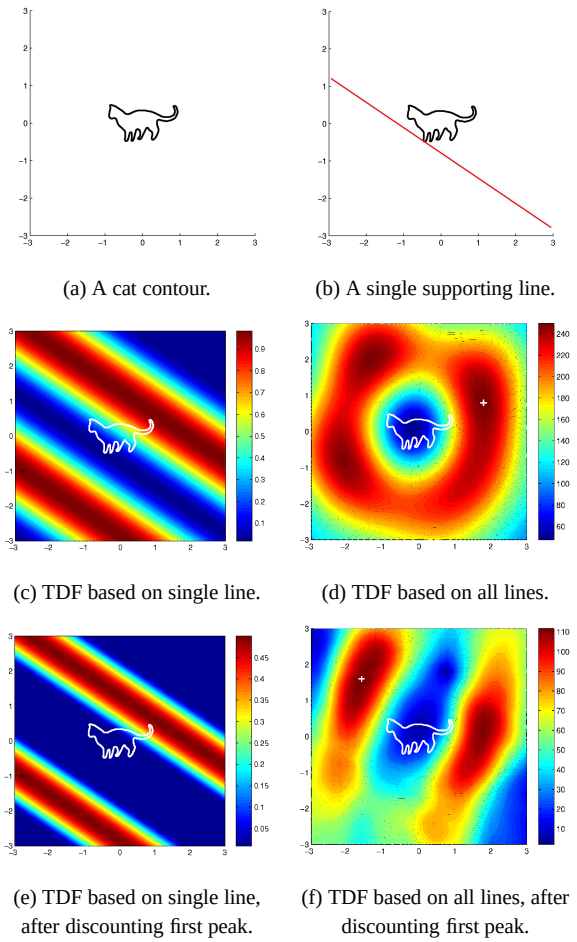
In this section we describe the use of the TDF for selecting single and multiple points of interest. Depending upon the application, these points could be camera positions for generating thumb nails of an object, origins for an optimized partition of a mesh's Hough transform, or points of other types. This should be encoded in the support distance function and the search domain. which we cover in Section 5.

4.1. Selecting a single or first point

Selecting a single, best point amounts to finding the peak value in the TDF. This is illustrated using a simple 2D example shown in Figure 2; see (a)-(d). We have chosen a particularly simple peak selection scheme by coarsely sampling the field's domain, and then recursively subsampling and searching around the highest-valued sample until a quality criterion is met, e.g., until the difference between consecutively detected peaks falls below a user-specified threshold. More sophisticated schemes can certainly be applied as well.

To ensure good results via sampling, the sampled scalar field should be sufficiently smooth and slow-varying or prefiltered to be so. Smoothness of the SDFs is ensured by our choice in the applications; Section 5 provides the details. As the TDF is a sum of SDFs, it is smooth itself. In practice, we have found that the SDFs we use are generally sufficiently well-behaved so that a relatively coarse initial sampling works sufficiently. We use a 17^3 initial sampling grid throughout. Since the smoothness of the TDF depends upon the smoothness of the summed SDFs, rather than characteristics of the input mesh, sampling resolution can be kept constant for a given application. However, more abrupt SDF changes than those encountered in this paper may generate steeper gradients in the TDF and will require a denser sampling pattern. An additional smoothing step may also be applied prior to sampling and peak detection.

In our experiments, we have not observed any meshes which produced multiple equal-valued maxima; however, it is possible that meshes with rotational or reflective symmetry may do so. In such a case, any of the maximal-valued points can be chosen as the first candidate. Any other maxima which are optimal for planes not contributing to the chosen point will be chosen later according to the weighting scheme described in Section 4.2. If these symmetric maxima are unsuitable for the given problem, an asymmetric SDF can be used to slightly penalize points across symmetry axes or planes; Section 7 provides an example.



(a) A cat contour. (b) A single supporting line.

(c) TDF based on single line. (d) TDF based on all lines.

(e) TDF based on single line, after discounting first peak. (f) TDF based on all lines, after discounting first peak.

Figure 2: *A 2D example of TDF construction and point selection. (a): A cat contour composed of line segments, analogous to a mesh composed of triangles. (b): One supporting line (in red), analogous to a mesh supporting plane. (c): Plot of TDF based on the supporting line shown in (b); the SDF for Hough-space silhouettes (see Figure 13) is used. (d): TDF plot based on all supporting lines. The highest peak is indicated by the white cross. In the analogous 3D case, this would be the first origin chosen for an optimal partitioning of the model's Hough points. (e): TDF based on the same supporting line in (b), discounted by the score it gave to the first peak. (f): TDF based on all supporting lines, after discounting contributions made to the first peak.*

4.2. Selecting the next point

Having selected the peak value from the initial TDF, we may wish to find the next-highest peak, and the next, and so on. Each selected point will benefit some planes more than others. Depending on the application, this might mean placing some faces far from the silhouette, or maximizing the perceptual contribution of those faces. When selecting subse-

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.

quent points, we would like to give priority to planes that have not yet contributed to a selected point, but we should not ignore planes that have already done so.

To provide a better intuition, let us use the voting analogy to describe our approach. Each plane assigns a vote to each point in space, weighted by the SDF. The sum of these weighted votes over all planes is the initial TDF. To select subsequent points, we bias the votes cast by each plane when selecting point k by the preference that plane has for the first $k-1$ points. Exactly how this is to be accomplished depends upon the application, but in general we wish to find the plane's highest weighted vote among the already-selected points and compare that to the plane's vote for the point currently under consideration:

$$w_{\text{prev}}(\pi) = \max_{i \in 1..k-1} f(\text{dist}(L[i], \pi)),$$

where π is the plane casting its vote and L is the list of previously-selected points of interest. Now, when finding the weight for π's vote on a point P, we calculate $w = f(\text{dist}(P, \pi))$ as usual, but cast a vote with weight

$$w_k(\pi) = \begin{cases} 0 & \text{if } w < w_{\text{prev}}(\pi) \\ w - w_{\text{prev}}(\pi) & \text{otherwise.} \end{cases}$$

Thus planes which have not yet elected a point which they favour cast heavily weighted votes, while planes which have already elected a suitable point of interest can still contribute to an even better point while not overwhelming the others. Figures 2 (e) and (f) give an example of this process.

4.3. Complexity considerations

Whenever we sample the TDF, we must compute the influence of each of the n triangles in the mesh, an $O(n)$ operation. Any performance improvements over the brute-force solution must therefore come from efficiency in the sampling. These improvements are a result of tradeoffs between the precision with which an application-dependent SDF represents the features we need and the sampling density required to find peaks in the TDFs generated by the SDFs.

A more detailed analysis of the relationship between the properties of SDFs and the number of samples required to find peaks in the TDF to a given accuracy may provide some insight into the structure of the method. However, we have obtained consistently good results by subsampling around peaks using a 17^3 lattice of increasing resolution, needing no more than three subsampling iterations.

Due to the linear complexity of evaluating the TDF and the not insignificant number of samples required to find a peak, our method is not at the moment suitable for real-time application. It is, however, quite practical as a preprocessing tool, and all of the applications given in this paper fit this role. As an illustration, we provide timing results for the Hough space optimization problem (the first of our applications described in Section 6) in Table 1.

Model	Size (tris)	Origins	Time
Hand	12379	2	1m30.88s
Horse	39699	3	8m52.12s
Bone	65001	2	7m26.82s
Bunny	69452	3	16m49.0s
Igea	268686	4	76m32.32s
Dragon	200000	3	43m12.57s

Table 1: *Timing results for Hough space origin optimization (see Section 6). These times show that our method is presently unsuitable for real-time processing, but quite practical as a preprocessing step.*

5. A voting scheme cookbook

In every case we assume that the input mesh has been normalized to fit within the unit sphere. For the given applications, we assume that the mesh is largely smooth; however, since the voting scheme acts as a filter over the whole mesh, a small number of local irregularities, boundaries, and even non-manifold edges are easily tolerated.

5.1. Domain restrictions

In many cases, the application itself will restrict the domain of the TDF. In the viewpoint selection problem, for example, it makes sense to place cameras within a spherical shell centered around the mesh, maximizing the size of the model in the rendered image while not allowing the view frustum to clip the mesh. However, we must understand how different restricted domains affect the interpretation of SDFs.

5.1.1. Unbounded domain

Given a domain that extends to infinity, we cannot infer anything new from the distance of a point to a plane. In such a domain, therefore, the SDF will only tell us whether a point is close to, or far from, the plane in question. However, when addressing problems involving the dual or 3D Hough [OZ06] transforms, this may be sufficient: these transforms are defined in terms of distances from the chosen origin to a set of planes.

5.1.2. Bounded domain

Most domains will be bounded, if for no other reason than that it is inconvenient to sample an infinite structure at a given resolution. When the domain is bounded on the outside, the maximum widths of the silhouette wedges of each edge on the mesh are likewise bounded. These wedges are the regions defined by the supporting planes of the wedge's adjacent triangles from which the edge is on the silhouette; see Figure 10 and [PDB*01, OZ06] for more details.

If we know (or assume) the mesh to be smooth, we can infer silhouette information from point-plane distance: any viewpoint a certain distance away from a given plane cannot

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.

silhouette edges in that plane (i.e., make edges in that plane silhouette edges). The converse is not necessarily true: near an edge, a viewpoint may be arbitrarily close to a supporting plane on that edge while not silhouetting it. Summing over every plane in the mesh, we can identify viewpoints with small silhouettes; if we have reason to prefer viewpoints far from mesh edges, we can identify viewpoints with large silhouettes as well.

5.1.3. Spherical shell domain

If the domain of the TDF can be restricted to a spherical shell around the mesh, we can infer further information from the SDF. Since we can guarantee a minimum distance from each edge on the mesh, we can reliably identify viewpoints that silhouette edges on a plane – or in general, viewpoints which see large silhouettes.

We can also infer the viewing angle from a point to the triangle that generated a given supporting plane. Since we are restricted to a spherical shell some distance from the mesh itself, we can interpret the point-plane distance as roughly representative of the sine of the viewing angle. We should take care not to expect great precision from this interpretation – however, since our SDFs must be smooth, we can identify viewpoints with "small" and "large" average viewing angles with some confidence. Note that this can be seen as a generalization of silhouette identification.

5.2. Support distance functions (SDFs)

To maintain the desirable sampling properties of the TDF, we must avoid abrupt changes in the support distance function or SDF. Any function that steps smoothly from 0 to 1 with zero derivatives at each extreme will be suitable. In our work, we have chosen the cubic function $f(x) = 3x^2 - 2x^3$ for efficiency. In every case described below, we can invert the meaning of f by taking $g = 1 - f$.

5.2.1. Triangle facing

The input argument to the SDF is the *signed* point-plane distance. This allows us to account for plane orientation (and therefore triangle facing) in our functions. We must maintain smoothness. We cannot, for example, select for all front-facing triangles with a step function. We must instead insert a smooth step at the origin, even though this gives a small weight to back-facing triangles (see Figure 3).

5.2.2. Point-plane distance

It is straightforward to build an SDF that votes for near or far points; see Figure 4. As with the plane-orientation function shown in Figure 3, we cannot select exactly those points closer or further than a set distance d; we accept a certain imprecision in our SDFs to maintain its desirable sampling properties.

Figure 3: *Support distance functions (SDFs) to identify front-facing planes. The left-hand function is discontinuous, and cannot be used as an SDF. The right-hand function is an acceptable substitute.*

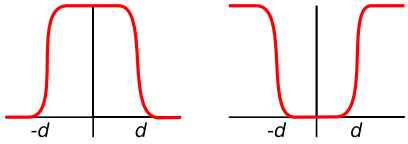


Figure 4: *SDFs for plane distance. The left-hand function votes for points closer than d units to each plane; the right-hand function votes for points further than d units away.*

5.2.3. Silhouette identification

As discussed above, with a restricted domain it is possible to identify viewpoints which see large (or small) silhouettes by their aggregate distance from the mesh's supporting planes. We can generally assume a smooth mesh and restrict the TDF domain to the near vicinity of the mesh, thus identifying viewpoints with large (or small) silhouettes with respect to a given plane usually amounts to voting for (or against) points that are very close to the plane; see Figure 5.

5.2.4. Viewing angles

By further restricting the domain, we can treat distance as roughly indicative of the sine of the viewing angle, as mentioned in Section 5.1.3. The relationships between point-plane distances and approximate viewing angles (with appropriate SDFs) are shown in Figure 6.

5.2.5. Combining SDFs

Most applications will require more detailed SDFs than the single-feature functions given above. Hough space origin optimization, for example, combines the off-silhouette func-

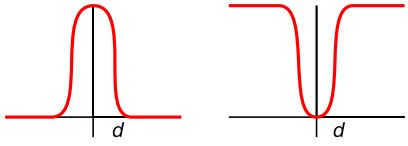


Figure 5: *SDFs for silhouette problems. The left-hand function votes for viewpoints likely to produce silhouettes on the given plane; the right-hand function votes against.*

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.

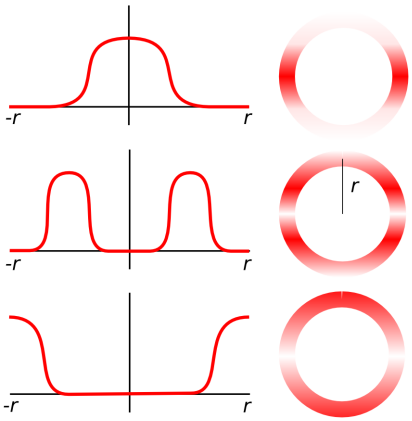


Figure 6: *SDFs for viewing angles. From top to bottom: SDFs selecting small (near zero), moderate (near $\pi/6$), and large (near $\pi/2$) viewing angles. The right-hand column shows a 2D analogue of the weights of the votes cast.*

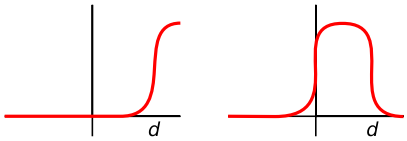


Figure 7: *SDFs combining several of the basic properties described earlier. The left-hand function votes for points far in front of each plane; the right-hand function votes for points near, but still in front of, each plane.*

tion in Figure 5 with an inverted far-from-plane function from Figure 4; see Section 6.

When combining SDFs, we select the *features* we need from each of the source functions. For example, to build a function that votes for points far from, and in front of, each plane, we can take the far-from-plane function from Figure 4 and remove the negative (left-hand) lobe. However, to build a function that votes for points *near* and in front of each plane, we combine the smooth step up at the origin from the front-facing function in Figure 3 and the smooth step down at an appropriate distance from the near-plane function in Figure 4. See Figure 7 for an illustration.

5.2.6. Choosing function scales

The above examples of SDFs are given without scales. In general, the output of the function ranges between 0 and 1, although some applications may wish to weight these values, e.g., by the area of the triangle generating the supporting plane. The width of the function is more variable, and should be determined on an application-specific basis. We consider this problem in more detail in each of the following sections.

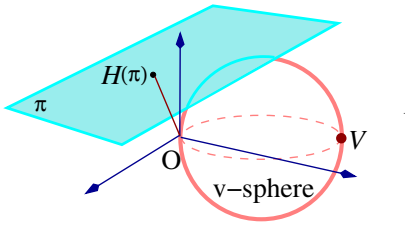


Figure 8: *The Hough transform of a plane π with origin O is the orthogonal projection $H(\pi)$. The Hough transform of a viewpoint V is the* v-sphere *(red) with antipodes V and O.*

6. Hough-space origin optimization

To date, none of the dual-space or Hough-space representations used for silhouette computations [HZ00, PDB*01, OZ06] has been optimized. We have observed that the location of the origin for Hough transform can drastically influence the resulting point distribution, which in turn, can strongly affect algorithm performance. Furthermore, the utilization of *multiple origins* and appropriate *grouping* of mesh data can lead to additional performance gains.

In this section, we describe a method to find these optimized origins using the voting approach described in the previous section. Substantial performance gains can be obtained for static silhouette extraction without degrading the performance of silhouette updates in any way. Note that while quick silhouette updates is crucial in an interactive setting, frame-to-frame coherence can be taken advantage of at the same time. In this sense, the initial extraction of silhouettes might offer more of a computational challenge. When the viewpoint is teleported or snapshots of a scene are taken, e.g., for object recognition, one needs to extract mesh silhouettes from scratch efficiently.

6.1. Silhouette extraction in Hough space

The *3D Hough transform* maps planes to points and points to spheres. Given a plane $\pi : ax + by + cz - d = 0$, where $a^2 + b^2 + c^2 = 1$, the Hough transform $H(\pi)$ of the plane π is the point (ad, bd, cd), which is the orthogonal projection of the origin onto the plane π; see Figure 8. Given a point $p = (a, b, c)$, the Hough transform $H(p)$ of the point p is the sphere with antipodal points at p and the origin. The Hough transform of a polygon mesh is the collection of points given by the Hough transform of its faces. Obviously, the orientation of the mesh faces is lost during this process. The viewpoint in the original space is transformed to a sphere, called the *v-sphere*. We will refer to the Hough transform of a plane (or equivalently a face's supporting plane) as the *Hough point* of that plane (or face).

The silhouette extraction algorithm relies upon an augmented octree over the mesh's Hough transform. Each node in this octree maintains two bounding boxes: the *point*

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.

bounding-box (*PBV*) and the *edge-bounding-box* (*EBV*). Each node's PBV tightly bounds the points contained within the octree node; this bounding volume is smaller than the octant of its parent which defines the node. The EBV at a node bounds all points contained in the node, as well as points corresponding to faces that are *adjacent to* faces contained in the node. This allows us to detect the containment of faces with respect to the v-sphere, as well as the intersection of mesh edges with respect to the v-sphere.

To find the mesh silhouette for an arbitrary viewpoint v, we use a theorem proven in [OZ06]: if an edge e with adjacent faces π_1 and π_2 is *not* on the silhouette from the origin, it is on the silhouette from v if and only if exactly one of the Hough transforms, $H(\pi_1)$ or $H(\pi_2)$, is contained in the v-sphere corresponding to v. Thus in preprocessing, we compute the set of mesh edges that are on the silhouette from the origin; we call this the *Silhouette-From-Origin* (*SFO*) set. To compute the silhouette from v, we first check each edge in the SFO set explicitly and then use the augmented octree to identify the non-SFO edges that cross the v-sphere — they are also on the mesh silhouette.

A full description of the Hough-space silhouette extraction algorithm can be found in [OZ06]. The crucial point to note is that more uniform point description in Hough space lead to more balanced search tree data structures, which in turn, results in performance gains. Thus we select multiple Hough-space origins with this goal in mind.

6.2. Optimizing for initial silhouette extraction

Silhouette extraction incurs most of its cost in bounding-box checks; we must check against both PBVs and EBVs. Direct optimization for origin count and positions should measure these operation counts in silhouette computation. However, modeling that problem is unrealistic as it is difficult, if not impossible, to mathematically relate operation counts with mesh geometry. We thus resort to a heuristic, which is based on an observation that there is a correlation between the number of SFO edges and the number of unnecessary EBV checks, and between the average distance of Hough points to their origins and general performance. These issues are elaborated in the next section and the discussion motivates the use of the TDF and our voting scheme.

It follows from the definition of the Hough transform that we can change the Hough transform of a set of planes by changing the origin. We can substantially increase initial silhouette extraction performance if we select a small set of origins and assign each triangle on a mesh to an appropriate origin. These performance gains come from minimizing the number of extraneous bounding-box incurred.

Some visual results from our approach are shown in Figure 9. As we can see, good origins can be some distance away from the mesh and the corresponding grouping of the mesh faces generally do not resemble results from known mesh segmentation algorithms. Naively placing origins at the centroids of different parts of an object generally leads to poor performance for silhouette extraction.

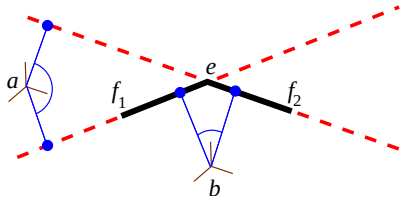


Figure 10: *An edge e, its neighboring faces f_1 and f_2, and two candidate origins a and b. The edge is on the silhouette with respect to origin a, but not with respect to b. Note the angles formed by the Hough transforms of f_1 and f_2 from origins a and b — the smaller angle generated from b is preferred.*

6.3. Reducing SFO edges and use of TDF

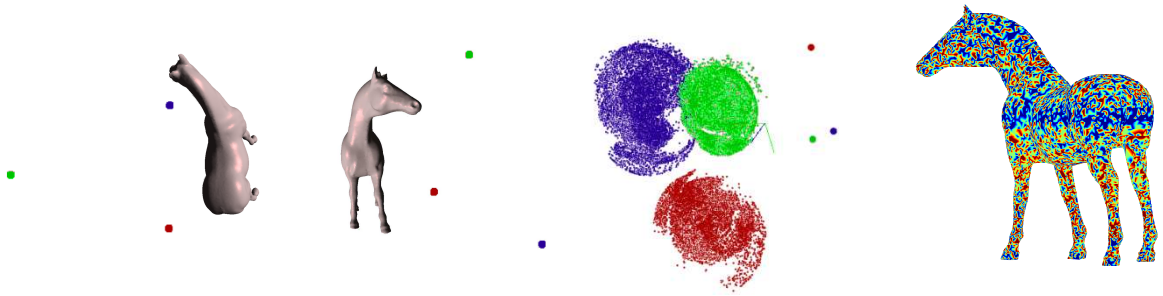
As argued in Section 6.1, we must check every face adjacent to an SFO edge explicitly when performing initial silhouette extraction. By selecting an appropriate set of origins we can substantially reduce the number of SFO edges in the mesh. If the mesh is smooth, that is, if most dihedral angles between adjacent faces are close to π, eliminating SFO edges has the additional and significant benefit of reducing the number of spurious EBV intersections, as we show below.

Recall that the Hough transform of a face is a scalar multiple of its unit normal vector, with the scalar being the distance from the origin to the supporting plane of the face. It follows that if the dihedral angle between two adjacent faces f_1 and f_2 is θ, the angle formed by their Hough points must be either θ or $\pi - \theta$ (see Figure 10). This angle is strongly correlated with the probability that a v-sphere will intersect an EBV containing f_1 and f_2, whether the edge between the two faces is on the silhouette or not.

The probability that an EBV will intersect a v-sphere is not solely determined by its volume: since all v-spheres must pass through the origin, the geometry of a v-sphere depends strongly upon the orientation of the viewpoint it represents. By minimizing the angle between two Hough points, which we subsequently refer to as their *angular extent*, we reduce the angular extent of the EBVs they generate in their octree. This in turn reduces the likelihood that a randomly-chosen viewpoint's v-sphere will intersect those EBVs.

In the worst case, a poor choice of origin for f_1 and f_2 may result in an EBV that contains the origin; see Figure 11(a). This EBV will never be discarded by an octree traversal, as all v-spheres pass through the origin. Suppose, however, that we have a sharp edge — that the dihedral angle between two faces is close to zero; see Figure 11(b). In this situation, we may easily obtain a worst-case EBV if we try to

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.



(a) Top view. (b) Front view. (c) Hough points. (d) Face grouping.

Figure 9: *A horse model (39,698 triangles) and three Hough-space origins (colored markers) selected by our algorithm are shown in two views in (a) and (b). The origins and their respective Hough-space points are shown in (c). Grouping of the mesh faces based on the origins is visualized in (d) via color coding. Extraction of the horse silhouettes takes about* 2 *milliseconds, compared to close to* 5 *milliseconds using an unoptimized single-origin Hough transform, as done in the original Hough-space silhouette extraction algorithm [OZ06].*

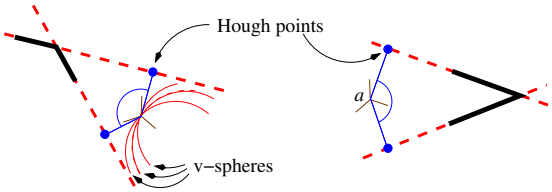


Figure 11: *Two worst-case scenarios involving poorly-chosen origins. (a) The EBV generated by two adjacent faces includes the origin and will never be discarded. (b) A sharp edge induces a large angle between the Hough transforms of its incident faces, when it is not forced onto the SFO.*

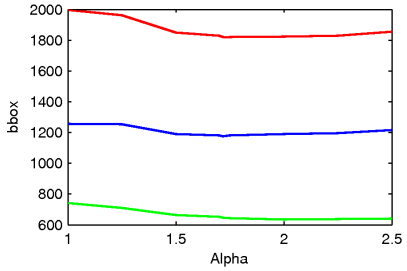


Figure 12: *Number of bounding box (bbox) checks (green: PBV; blue: EBV; red: total) vs.* α *for the hand mesh with two origins.*

eliminate the sharp edge from the SFO set. Using the methods of Section 5, we can choose origins which discourage both of these worst-case scenarios within the framework of our voting scheme by choosing a support distance function that penalizes points of interest (candidate origins) too close to supporting planes.

However, assigning a plane to its most distant origin does not by itself guarantee that the angular extent of its edges will be minimized. Furthermore, excessively distant origins tend to produce highly non-uniform point distributions in Hough space. Uniformity of point distribution is one of the greatest advantages of Hough transform for silhouette extraction. We therefore seek to keep our origins as close to the mesh as possible — but no closer.

We can choose these origins by the peak-finding method introduced in Section 4. The SDF described in Figure 13 selects those points where on average, each edge has a small likelihood of lying on the silhouette from one of the selected origins, but is not located so far away as to unbalance any origin's resulting octree (see Figure 9 (c)). We describe this method in detail below.

6.4. Domain restriction and voting scheme

We have found that restricting the domain of the voting field to a sphere of radius 3 around the normalized mesh produces excellent results. This restricted domain gives us the properties discussed in Section 5.1.2, but large enough to fully contain the peaks of the TDF.

We use the SDF shown in Figure 13. Note that this function is parameterized by a scalar α and has range $[0,1]$. An optimal value of α will capture not only the mathematically tractable property of minimizing silhouette size on a smooth mesh, but also the decidedly untidy property of generating a Hough transform whose points produce a well-balanced octree. However, experimentally, we have found that for the rather diverse set of mesh models in our test, an α value of about 1.72 generally produced the best performance. Figure 12 shows the effect of changing α on the number of bounding box checks for the hand mesh with two origins. This reveals a general trend that the algorithm performance does not vary much with α, as long as it is not too small.

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.

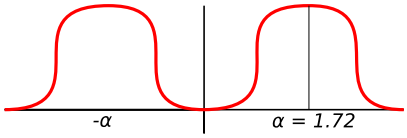


Figure 13: *The SDF for Hough space origin optimization.*

6.5. Origin selection and face grouping

In our current implementation, we select a user-specified number k of origins based on the TDF defined above. The origins are selected one at a time in a greedy fashion. After one origin is selected, the scalar field is updated to remove its influence. An obvious question that arises is how large k should be. Indeed, multiple origins ($k > 1$) tend to improve performance but only to a certain extent. A large value of k tends to reduce the SFO set, and consequently reduces the number of unnecessary leaf-level EBV checks, but increasing the number of octrees increases the overhead of bounding-box checks near the roots of the trees. We find that values of k between 2 and 4 work well for most meshes.

Automatically choosing an optimal k is probably difficult; this problem bears a similar characteristic as choosing the right k in k-means clustering. Given sufficient processing time, one can always test a sequence of values for k and for each k perform the origin selection heuristic and rely on performance measurements to choose an appropriate k. In fact, this has been a well-practiced heuristic for choosing the number of clusters in clustering analysis [ELL01].

The origin selection algorithm above is concerned only with distances from origin to face supporting planes, or equivalently, distances from origin to Hough points. As previously noted, such distances alone offer no guarantee that we can minimize the angular extent of Hough point pairs contributing to EBVs. Rather than assigning faces to the origins with which they score the highest, we first consider whether each origin would put any smooth edges on that face on its silhouette, then use face distance as a tiebreaker.

We define the *SFO count* of a face f with respect to an origin o as the number of edges of f that are on the silhouette from o, not on the mesh boundary, and have dihedral angle less than $\pi/2$. Naturally, the dihedral angle at an edge is formed by its two adjacent faces. If f has a nonzero SFO count with respect to o, we expect that at least one of the edges on f will incur an EBV with a large angular extent from o. We therefore assign f to the origin that minimizes its SFO count. In the not unlikely event that f has the same (minimal) SFO count with respect to more than one origin, we assign f to the origin from which it is most distant.

6.6. Multi-origin silhouette extraction

Having found a set of k origins, we assign faces in the mesh to a single origin, then build an augmented octree for each [OZ06]. Rather than assigning faces to their favoured candidate origin, we first consider whether each origin would put any smooth edges on that face on its silhouette, then use face distance as a tie-breaker.

We can now construct an augmented octree for each origin and its associated group of faces. However, for each group C, we must take care to account for faces that are not present in C but are *adjacent to faces that are* in C. If we were to ignore these faces during octree construction for C, we would not create the proper EBVs, and would risk not identifying these "group-crossing" edges during the initial silhouette extraction step. To account for these faces, we create "ghost points" corresponding to them while creating the EBVs for C. These ghost points are not present in the octree for C and do not contribute to its PBVs, but they do affect EBVs.

To find the static mesh silhouette with respect to a viewpoint v, we must first check each SFO edge, as before. We are only obliged to do so if it is on the silhouette from the origins of the groups to which its adjacent faces have been assigned. This eliminates the vast majority of explicit SFO edge checks. We can now traverse the octree for each group independently, as in [OZ06]. Before doing so, we must transform the global viewpoint into the space of each cluster in order to calculate the correct v-sphere. Since a change of origin can be interpreted as a translation, this is a simple matter of subtracting the cluster origin from the viewpoint.

6.7. Experimental results

We tested our algorithms on a PC running Linux 2.6.18 with four Intel Xeon 3GHz processors and an NVidia GeForce 6800 Ultra graphics card. Six models, listed in Table 2, were used, with face counts ranging from 12K to 268K. Note that these same models were also used in the experiments of the original Hough-space silhouette paper [OZ06]. We intend to directly compare our results with those obtained in that work. In addition to reporting the number of bounding box checks, seen as the atomic operations of the algorithm, we also give actual timing for silhouette extraction. One should keep in mind however that our code had not been optimized and such timing results are only meant to provide a general impression of the speed of our algorithms.

We tested initial silhouette extraction performance for optimized Hough spaces by selecting 3,865 viewpoints evenly distributed over three spheres, all centered about the mesh centroid and having varying radii from 1 to 3. Recall that we have scaled the test meshes to fit within the unit sphere. Results and comparison with extraction using single-origin Hough transform [OZ06], where the origin is chosen as the mesh centroid, are summarized in Table 2. The relatively high cost for the bone and dragon meshes is due to the higher complexity of their silhouettes.

Our program requires anywhere between two minutes (for the hand) and seventy-five minutes (for the Igea) to find a

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.

set of optimal origins, depending on the size of the mesh and the number of origins required. Naturally we would like to improve upon this, but for a preprocessing step this is still acceptable and it measures up favorably against the cost of other optimization approaches, e.g., [SGG*00]. The best choice for the ideal distance α is likely model-dependent. We choose $\alpha = 1.72$ throughout this set of experiments, as discussed in Section 6.4. The optimal number of origins k varies across the test models, but remains relatively small.

7. Viewpoint selection

We approach the viewpoint selection problem from an essentially perceptual point of view: we wish to select a small set of k viewpoints — perhaps three or four — which together provide as complete a visual description as possible of the target mesh. This is important for users browsing large mesh databases, where a small number of low-resolution thumbnail images may be generated for each entry and should allow the user to identify the mesh.

7.1. Selection criteria

Silhouettes form one of the most important perceptual cues in object recognition [Koe84]. When humans sketch objects, they tend to define features by their silhouettes. Therefore, we seek to maximize the perceptual information provided by a set of viewpoints by choosing those viewpoints to maximize the length of the mesh's silhouette from each.

In general, this problem is difficult to formulate in terms of supporting planes. However, most large meshes are composed of triangles of similar size. We assume that the lengths of the input mesh's edges are consistent and close to uniform — if this is the case, we can reduce the problem of measuring silhouette length to the problem of counting silhouette edges. This problem can be straightforwardly formulated in the voting framework we have described.

In addition, we follow an observation of Vazquez *et al.* [VFSH01] that faces seen at orthogonal or nearly orthogonal viewing angles contribute greatly to view entropy, their measure of viewpoint quality. With this in mind, we also wish to select viewpoints that can see a large number of faces at large viewing angles. Again, this is simple to integrate with silhouette selection in our voting framework.

In general, we are not concerned with back-facing triangles in the viewpoint selection problem. The silhouette selection function from Section 5 will inevitably give heavily-weighted votes to back-facing planes near the silhouette, but as this reinforces points that generate a large silhouette it is not undesirable. However, a large number of back-facing planes orthogonal to the viewing direction, when combined with a similar number of front-facing orthogonal planes, tend to indicate a reflective symmetry plane. Podolak *et al.* [PSG*06] point out that symmetry information is perceptually useful for viewpoint selection. We therefore give a small weight to back-facing planes with orthogonal viewing angles, to take advantage of reflective symmetry in the input without overwhelming the above constraints.

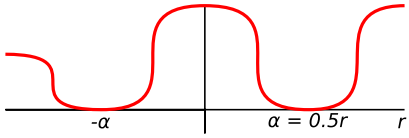


Figure 14: *The SDF used for viewpoint selection. The peak in the centre favours planes near the silhouette, while the plateau at $+x$ favours front-facing planes with orthogonal viewing angles. The shallower plateau at $-x$ slightly favours back-facing orthogonal planes, encouraging the use of symmetry without overwhelming the other factors.*

7.2. Domain selection and voting scheme

We select viewpoints within a spherical shell around the mesh, with inner radius $\sqrt{3}$ and outer radius 2. This is sufficient to fit the whole mesh within our view frustum while not placing the camera too far away. As silhouette size can vary with distance, we search over a thick shell rather than a set of points at constant radius like most other methods. Our final SDF is shown in Figure 14. We have set the α parameter to 1, half of the outer radius. This ensures angle-selecting behaviour as shown in Figure 6.

We have found experimentally that on most "object" meshes, points that maximize total silhouette size also have large visible silhouettes. However, we can easily account for any discrepancies introduced here by further weighting the SDF by the ratio of visible silhouette pixels to total silhouette pixels. At each point, we render the silhouette to a buffer, first with the depth test enabled to find the visible silhouette, then with the depth test disabled to find the total silhouette. We calculate the total votes cast for the point in question as usual, and scale the number by the ratio of visible to total silhouette pixels. This does not drastically change the positions of the selected viewpoints, but does bias them slightly to make more features visible. See Figures 15 and 16.

7.3. Experimental results

The object views shown in Figure 15 show several improvements over the methods summarized in [PPB*05]. For example, the results presented here show each model from a variety of well-separated viewing angles, rather than selecting a number of spatially similar viewpoints. This is similar to the result achieved by [YSY*06], but does not require graph partitioning as in that work. Our method also avoids viewpoints positioned opposite each other across the model's planes of symmetry, correcting a flaw noticeable in the purely silhouette-based algorithms detailed in [PPB*05].

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.

			Unoptimized; single-origin		Optimized; multi-origin			
Model	No. faces	Avg. Sil. Size	Bbox checks	Time (ms)	Origins	Bbox checks	Time (ms)	Time speed-up
Hand	12,378	639	5,691	1.45	2	1,818	0.44	3.29
Horse	39,698	2,511	16,049	4.85	3	6,884	1.96	2.47
Bone	65,001	4,724	30,825	10.70	2	11,816	3.78	2.83
Bunny	69,452	3,498	20,376	5.55	3	9,073	2.46	2.25
Dragon	200,000	13,666	72,472	17.20	3	27,767	7.04	2.44
Igea	268,686	7,191	36,757	16.60	4	20,268	7.36	2.25

Table 2: *Performance statistics for our Hough-space origin optimization algorithm ($\alpha = 1.72$). Both the total number of bounding box (Bbox) checks and average silhouette extraction times (in milliseconds) are given. We also list the optimal number of origins, found by the heuristic described in Section 6.5.*

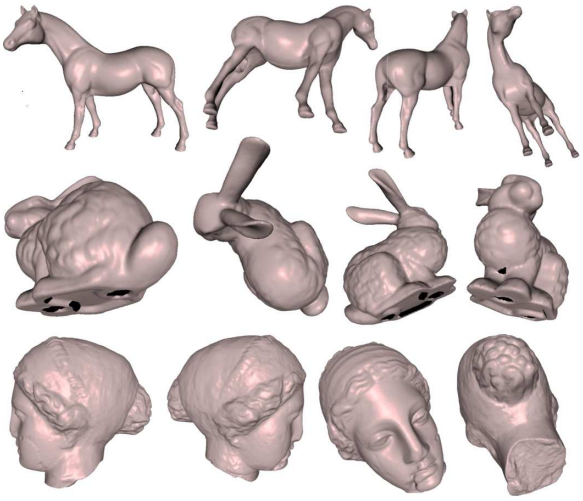


Figure 15: *Four best views selected for the horse, bunny, and igea. As the results for the bunny demonstrate, our method produces good views even for meshes with holes or borders.*

Figure 17 shows views generated for simplified versions of the horse shown in Figure 15. These views are nearly identical to those produced for the full-resolution horse; the only visually apparent difference is the fourth view chosen for the lowest-resolution mesh. This suggests that we can use the smoothing nature of the TDF to generate views for high-resolution models from a simplified input.

Figure 18 shows a wolf model in two different poses. Note that the views generated differ significantly between the poses and from the structurally similar horse mesh, highlighting the differences between the meshes.

Finally, Figure 19 shows the output of our algorithm on a CAD-type mesh, in this case a marching-cubes reconstruction of the fan blade. The large flat bottom of the fan blade dominates the TDF for the first viewpoint, producing a view that by itself is insufficient to capture the whole shape. However, as the faces on the flat bottom are penalized in subsequent iterations, the silhouette length criterion takes on increasing importance. The chosen set of four views shows all sides of the mesh from well-separated viewing angles.

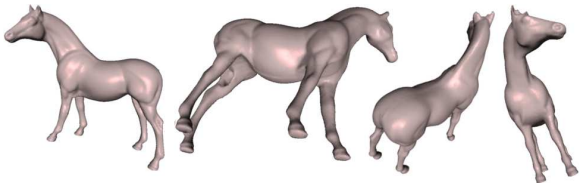


Figure 16: *Four views selected for the horse mesh without weighting by the visible silhouette ratio. Note that the results are similar to those shown in Figure 15; however, in the third and fourth views, one leg is occluded.*

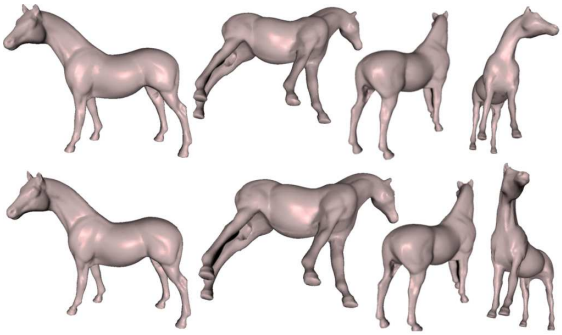


Figure 17: *Four viewpoints selected for simplified horse meshes with 10k (top) and 20k (bottom) faces. The results are nearly indistinguishable from the viewpoints found for the full resolution mesh with over 39k faces.*

While our method chooses well-separated and informative viewpoints, it does not compute a natural orientation for the camera. This is an interesting and difficult problem in its own right which is beginning to be addressed, for example by Fu *et al.* [FCODS08].

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.

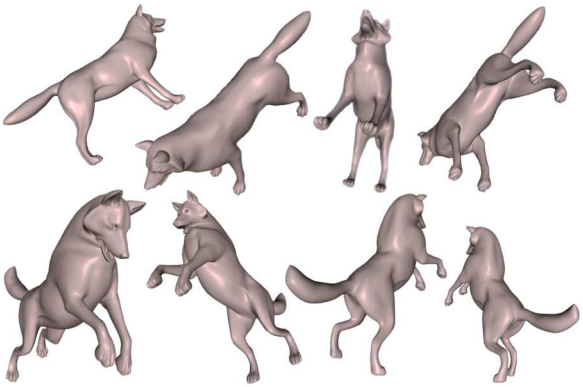


Figure 18: *Four views selected for two poses of the wolf mesh from ISDB. Views differ between the poses as the visually significant parts of the mesh change.*

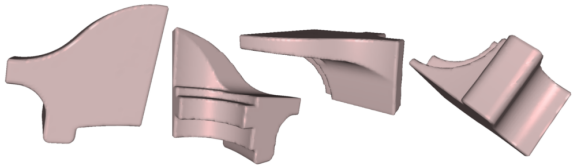


Figure 19: *Four views for the fan blade mesh, from first choice to fourth clockwise from the top left. The mesh is readily identifiable even though the first view is dominated by a large flat region.*

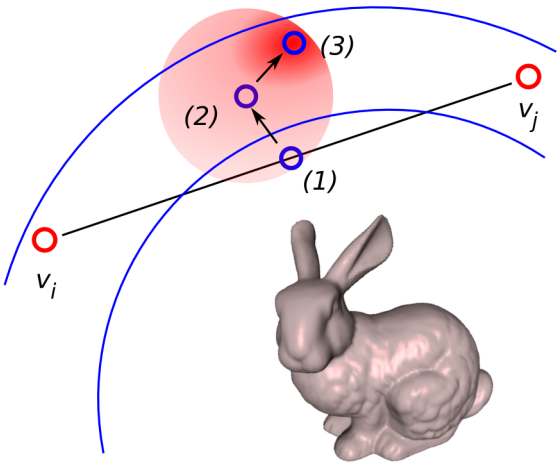


Figure 20: *Generating an intermediate path node between two viewpoints v_i and v_j. We start with (1) the point between v_i and v_j, then (2) project it into the TDF's restricted domain, and finally (3) move it to a nearby peak.*

7.4. Camera path planning

Certain applications will benefit from an automatically-generated camera path, providing a view of the whole model while focusing on the most visually significant parts. Generating such a path from our viewpoint selection TDF is efficient and straightforward. We construct a path that interpolates all of the selected viewpoints in a way that seeks to maximize the significance of intermediate viewpoints.

Let us first define *intermediate path nodes*. These nodes serve two purposes: they keep the camera path between viewpoints from intersecting the bounding sphere of the viewed model, and they guide the path to local maxima or ridges in the TDF, representing points with interesting views. To find the intermediate path node p_{ij} between two viewpoints v_i and v_j, we simply take the midpoint of the line segment $\overline{v_i v_j}$, extend it to the middle circumference of the TDF domain, and move it to the peak TDF value within a small neighbourhood, as illustrated in Figure 20.

Note that we use the initial TDF to compute intermediate point positions, rather than the modified TDF used to compute the second and subsequent viewpoints. The initial, unbiased TDF provides an overall metric for the significance of every viewpoint around the mesh, while the later, modified TDFs reduce the contributions of planes that are satisfied by previous viewpoints.

We can now compute a camera path as a sequence of alternating viewpoints and intermediate path nodes. Starting at the first viewpoint v_1, we select the destination viewpoint v_d that maximizes the value of the TDF at the intermediate node p_{1d}. We add p_{1d} and v_d to the camera path, remove v_1 from consideration, and repeat the process from v_d. When we have visited all viewpoints, we return to v_1, closing the path. When following the path, we interpolate between the path nodes using cubic Bézier curves. By applying the methods from Chapter 4 of [SL89] – namely, by ensuring that control points across an endpoint are collinear and symmetric – we can ensure a smooth path between viewpoints with GC^2 continuity at its joints.

8. Placement of single light source

Like silhouette arcs, lighting information is a powerful perceptual cue that can add significant visual information to a scene. The cues provided by a light source depend upon the location of the viewpoint and the properties of the mesh. Given a viewpoint (perhaps chosen by our algorithm from Section 7), we would like to place a directional light source to provide as much visual information as possible.

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.

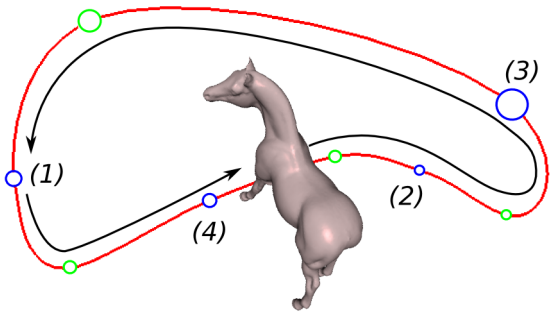


Figure 21: *Camera path generated for the horse mesh based on the four viewpoints chosen by our viewpoint selection algorithm, as shown in Figure 15. Viewpoints are shown as blue circles, intermediate path nodes as green circles.*

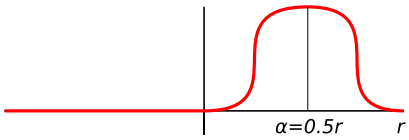


Figure 22: *The SDF for light source placement.*

8.1. Placement criteria and SDF

We wish to light the surface facing the viewer with as much variation in intensity as possible to highlight even small changes in angle. Within the Phong lighting model, this involves minimizing the angle from the light source to the set of planes facing the viewpoint. We wish to avoid large lighting angles, which will wash out the small features we wish to highlight. However, we must also minimize the size of the silhouette from the light source: silhouette loops enclose back-facing regions, which will not be lit (except by the light's ambient component) and thus give no information by lighting. This is almost exactly the opposite of our viewpoint selection criteria. We thus use the SDF in Figure 22 on the same restricted domain used for viewpoint selection.

Note that the SDF chosen for light source placement attempts to model the information provided by the light shining on a surface, and is thus tied to the parameters of the light. In this case we have chosen a matte material and a light with strong diffuse and weak specular components, in order to maximize the information conveyed by shading and minimize the saturating effects of specular highlights. See Figure 23 and compare to the other renderings in this paper. It may be possible to generate a light-placement SDF automatically from a set of material and light parameters; this is a direction for future research.

There are two major differences from the basic algorithm outlined in Section 4 in our approach to light source placement. First, rather than considering every supporting plane in the mesh, we operate only upon planes that face towards the viewpoint. Second, we are forced to discard the greedy next-best-point selection algorithm. In the other applications, once a plane has selected a point in which it has a high interest, it is unaffected by the selection of subsequent points. In this case, however, we may inadvertently select a new light position which washes out the contrasts which we have carefully selected with our first. Hence, we are only able to place a *single* light source. Adapting our method for robust multiple light source placement, including control over intensity, is a topic for future research.

8.2. Experimental results

We show light placement results for the horse mesh in Figure 23, compared to an arbitrary light source placement from Section 7. We placed light sources for each of the four best viewpoints chosen by our algorithm.

Let us focus on regions highlighted by circles in Figure 23, where the colors of the circles indicate correspondence between the regions. First note the near-total lack of shadowed (ambiently lit) regions in the bottom row of images. By comparison, the top row of horse images show a great deal of shadowing. This obscures detail of the horse's left hind leg in the second view, and much of the horse's sides in the third and fourth views. In all views, the musculature of the horse is more apparent, particularly on the neck, forelegs, and hindquarters, in the bottom row of images. Finally, in the fourth view, the unoptimized light source washes out detail on the side of the horse's face and neck, whereas the optimized light position shows this detail clearly.

9. Conclusion and future work

We have presented a unified framework for solving global point-selection problems based on planarity- and silhouette-related characteristics of a mesh or set of planes. This method is sufficiently flexible to address a number of separate problems: we have used it to partition and optimize Hough points for silhouette extraction, find several mutually supporting viewpoints for a target mesh, plan a smooth camera path which connects the selected viewpoints, and place light sources for maximal intensity variation. We have obtained excellent results, comparable to or exceeding those produced by specialized, domain-specific methods.

Our framework has a great deal of potential for addressing even more applications. Billboard cloud construction [DDSD03] can be seen as a viewpoint-selection problem with a strong dependence upon planarity constraints, and is a clear candidate for our method. The occlusion potential characteristic in Sander *et al.*'s triangle reordering paper [SNB07] is also strongly dependent upon the ordering, position, and orientation of planar clusters. We may be

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.

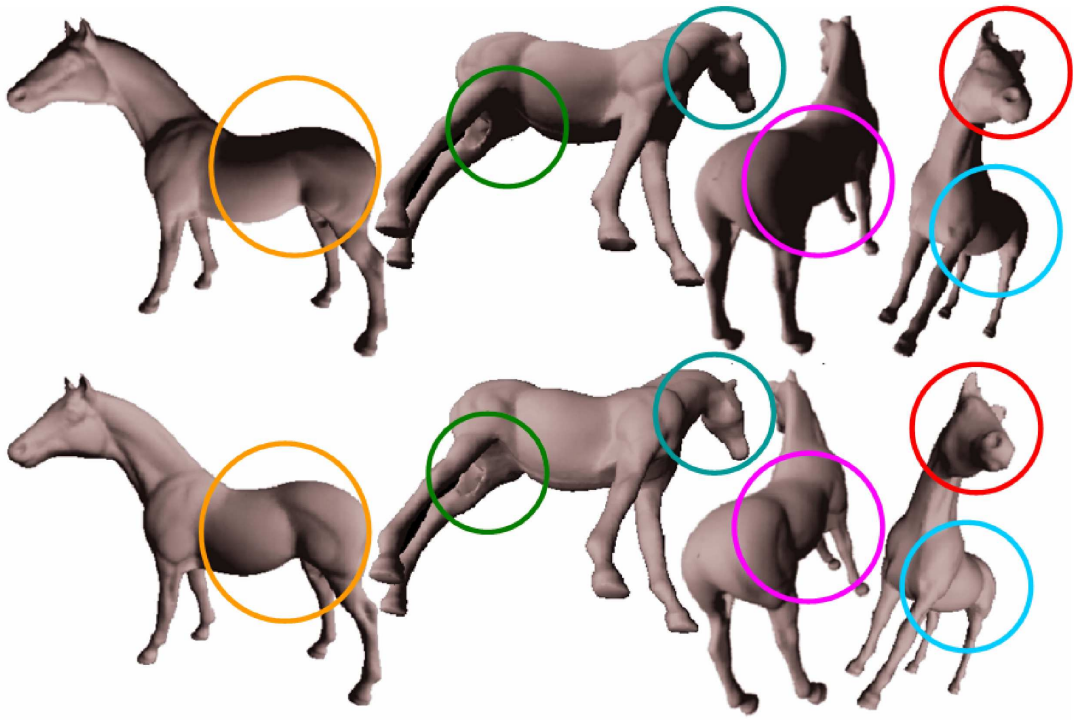


Figure 23: *Light source placement results for the horse (bottom) compared to an arbitrarily-placed light source (top). Colored circles indicate corresponding highlighted regions where we can observe differences made by our algorithm.*

able to use this characteristic to expand our method into occlusion-related problems. Since we generate a small number of features from aggregate mesh geometry using what can be seen as a low-pass filter, we expect our method to be robust to changes in connectivity and, to a certain extent, even geometry. We may therefore be able to use our framework to produce a compact mesh descriptor for rigid models.

The results from Section 6 are interesting by themselves. We have shown that by judiciously choosing multiple origins and grouping mesh faces accordingly, we can achieve significant performance gains. It is somewhat surprising that we can make so significant an improvement in performance by partitioning the transform space to suit the application without changing the geometry of the problem. We would like to apply this insight to other Hough-space methods. Given the inversion relationship between the 3D Hough transform and the dual transform, we may also be able to apply the same principles to dual-space methods.

References

[AVF04] ANDUJAR C., VAZQUEZ P., FAIREN M.: Wayfinder: Guided tours through complex walkthrough models. *Computer Graphics Forum 23*, 3 (2004), 499–508.

[CM93] COWAN C., MODAYUR B.: Edge-based placement of camera and light source for object recognition and location. In *IEEE International Conference on Robotics and Automation* (1993), pp. 586–591.

[CO06] CHRISTIE M., OLIVIER P.: Camera control in computer graphics. In *Eurographics 2006 STARs* (2006), pp. 89–113.

[DDSD03] DÉCORET X., DURAND F., SILLION F. X., DORSEY J.: Billboard clouds for extreme model simplification. *ACM Transactions on Graphics 22*, 3 (2003), 689–696.

[ELL01] EVERITT B. S., LANDAU S., LEESE M.: *Cluster Analysis*. Hodder Arnold Publishing and Oxford University Press, 2001.

[FCODS08] FU H., COHEN-OR D., DROR G., SHEFFER A.: Upright orientation of man-made objects. *ACM SIGGRAPH 27*, 3 (2008), to appear.

[Gum02] GUMHOLD S.: Maximum entropy light source placement. In *IEEE Visualization* (2002), pp. 275–282.

[Har92] HARRIS J.: *Algebraic Geometry: A First Course*. New York: Springer-Verlag, 1992.

[HDD*92] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Surface reconstruction from

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.

unorganized points. In *ACM SIGGRAPH* (1992), pp. 71–78.

[Hou59] HOUGH P. V. C.: Machine analysis of bubble chamber pictures. In *Int. Conf. on High Energy Accelerators and Instrumentation* (1959).

[HZ00] HERTZMANN A., ZORIN D.: Illustrating smooth surfaces. *ACM SIGGRAPH 18*, 3 (2000), 517–526.

[JBS06] JONES M. W., BAERENTZEN J. A., SRAMEK M.: 3d distance fields: A survey of techniques and applications. *IEEE Transactions on Visualization and Computer Graphics 12*, 4 (2006), 581–599.

[Joh04] JOHNSTONE J. K.: The bézier tangential surface system: a robust dual representation of tangent space. *Computing 72*, 1-2 (2004), 105–115.

[KMGL96] KUMAR S., MANOCHA D., GARRETT W., LIN M.: Hierarchical back-face computation. In *Proc. of the Eurographics Workshop on Rendering Techniques* (1996), pp. 235–253.

[Koe84] KOENDERINK J. J.: What does the occluding contour tell us about solid shape. *Perception 13* (1984), 321–330.

[Mor02] MORRIS R. J.: Visualising duals of surfaces, 2002. Unpulished manuscript.

[OZ06] OLSON M., ZHANG H.: Silhouette extraction in hough space. *Computer Graphics Forum 25*, 3 (2006), 273–282.

[PDB*01] POP M., DUNCAN C., BAREQUET G., GOODRICH M., HUANG W., KUMAR S.: Efficient perspective-accurate silhouette computation and applications. In *Proc. of Annual Symp. on Computational Geometry* (2001), pp. 60–68.

[PPB*05] POLONSKY O., PATANE G., BIASOTTI S., GOTSMAN C., SPAGNUOLO M.: What's in an image? towards the computation of the "best" view of an object. *The Visual Computer 21*, 8 (2005), 840–847.

[PPS03] PETERNELL M., POTTMANN H., STEINER T.: *Hough Transform and Laguerre Geometry for the Recognition and Reconstruction of Special 3D Shapes*. Tech. Rep. 100, TU Vienna, 2003.

[PSG*06] PODOLAK J., SHILANE P., GOLOVINSKIY A., RUSINKIEWICZ S., FUNKHOUSER T.: A planar-reflective symmetry transform for 3d shapes. In *ACM SIGGRAPH* (2006), pp. 549–559.

[SGG*00] SANDER P. V., GU X., GORTLER S. J., HOPPE H., SNYDER J.: Silhouette clipping. In *ACM SIGGRAPH* (2000), pp. 327–334.

[SJGar] SIR Z., JUTTLER B., GRAVESEN J.: Curves and surfaces represented by polynomial support functions. *Theoretical Computer Sciences* (to appear).

[SL89] SU B., LIU D.: *Computational Geometry: Curve and Surface Modeling*. Academic Press, 1989.

[SNB07] SANDER P. V., NEHAB D., BARCZAK J.: Fast triangle order optimization for vertex locality and reduced overdraw. *ACM SIGGRAPH 27*, 3 (2007), 89–98.

[SRKP04] SETHI A., RENAUDIE D., KRIEGMAN D., PONCE J.: Curve and surface duals and the recognition of curved 3d objects from their silhouettes. *Int. J. Comput. Vision 58*, 1 (2004), 73–86.

[TCO*05] TSAI Y., CHENG L., OSHER S., BURCHARD P., SAPIRO G.: Visibility and its dynamics in a pde-based implicit framework. *Journal of Computational Physics 199*, 1 (2005), 260–290.

[TL95] TRIGGS B., LAUGIER C.: Automatic camera placement for robot vision tasks. In *IEEE International Conference on Robotics and Automation* (1995), pp. 1732–1738.

[Vaz06] VAZQUEZ P.: Automatic light source placement for maximum visual information recovery. *Computer Graphics Forum 26*, 2 (2006), 143–156.

[VFSH01] VAZQUEZ P., FEIXAS M., SBERT M., HEIDRICH W.: Viewpoint selection using viewpoint entropy. In *Proc. of Vision, Modeling and Visualization* (2001), pp. 273–280.

[WK03] WU J., KOBBELT L.: Piecewise linear approximation of signed distance fields. In *Proc. of Vision, Modeling and Visualization* (2003), pp. 513–520.

[YSY*06] YAMAUCHI H., SALEEM W., YOSHIZAWA S., KARNI Z., BELAYEV A., SEIDEL H.-P.: Towards stable and salient multi-view representation of 3d shapes. In *Proc. of Shape Modeling and Applications (SMI)* (2006), pp. 40–46.

[ZP01] ZAHARIA T., PRETEUX F.: Hough transform-based 3d mesh retrieval. In *Proc. of the SPIE Conf. 4476 on Vision Geometry X* (2001), pp. 175–185.

© 2008 The Author(s)
Journal compilation © 2008 The Eurographics Association and Blackwell Publishing Ltd.