A Spline Surface Algorithm for Reconstruction of Corneal Topography from a Videokeratographic Reflection Pattern

Mark A. Halstead, M.Sc.

Brian A. Barsky, Ph.D.

Stanley A. Klein, Ph.D.

Robert B. Mandell, O.D. Ph.D. F.A.A.O.

Abstract

A videokeratograph quantifies one or more aspects of corneal shape by the computer analysis of a digital image. The image is the reflection formed by illuminating the cornea with a known source pattern. The type of shape information recovered and the accuracy of this information depend upon properties of the computer algorithm used in the analysis. We present a new algorithm that recovers a polynomial spline description of corneal depth.

Keywords: videokeratograph, spline surface, cornea, topography, reconstruction.

Introduction

The introduction of *videokeratography* has transformed photokeratoscopy from a basic research method to a practical clinical technique[1–6]. The major advance has been the introduction of computer algorithms that enable the detection and analysis of several thousand data points in only a few seconds, a process that formerly took days or even weeks with photokeratoscopy[7]. The computer programs also provide a range of data displays which allow the operator to visualize corneal features that were previously undetectable.

Unfortunately, apart from the advances in electronic features, there have not been major improvements in other parts of the videokeratograph, particularly in the optical system. The configuration of the mires, which are the light sources that are shone onto the cornea (see Figure 1), are basically those taken from the optical system of previous photokeratoscopes. In addition, corneal alignment methods have not changed[8]. The algorithms that are used to calculate the radius of curvature values are generally the same as those used previously in photokeratoscopy[9]. In particular, the optical portion of photokeratoscopy has lagged in development and has limited the accuracy of current systems.

The major optical problem that has plagued the advancement of videokeratography is that although the system measures an array of points in three dimensions, it limits the interpretation to two dimensions, which severely curtails the power of the analysis. In other words, most current videokeratograph algorithms are designed to analyze one meridian at a time and they fail to link the information that might be available if all the meridians could be treated as a single entity[4,10,11]. This is adequate if the cornea has symmetry for all meridians, such as would occur for a sphere or a radially symmetrical asphere (e.g., a surface of revolution of a conic section), but fails to address the problem of corneal asymmetry, which in fact is the usual condition[12].

Consider the problem presented for a videokeratograph when a toric cornea is to be measured, representing the simplest case of radial asymmetry. Rays from the mires that strike the cornea in one of the principal meridians are in a plane that includes both the normal to the corneal surface and the reflected ray to the videokeratograph camera. Hence, a two-dimensional analysis is adequate in the algorithm for the radius calculation. Next, consider a ray from a point on a mire that strikes the cornea in a meridian that is not one of the principal meridians. In this case, the ray leaving the corneal surface to the image will be skewed somewhat so that it no longer remains in the same plane as the object ray. However, current algorithms for videokeratographs treat this ray path as though it continues to lie in the same plane and contributes error to the system. This error can be corrected by a new approach to the analysis of videokeratograph output in which the data for all corneal points is analyzed as a complex surface, with no shape restrictions.

Computer analysis of the image enables the videokeratograph to quantify the shape of the cornea in new ways. The analysis steps are: inputing an image from the videokeratograph, running an algorithm to recover some aspect of corneal shape, and outputing this information for further display and analysis. The algorithm used in the analysis determines which aspect of corneal shape is measured and to what degree of accuracy.

There are three aspects of corneal shape that are of primary interest to a clinician[13,14]. The first aspect is *position*, which is the zeroth derivative of shape. This is sometimes referred to as *height*, *depth*, or *sag*. The position of a point on the cornea is determined by its three-dimensional spatial coordinates. The collection of points forming a corneal surface is conveniently defined by giving depth z as a function of coordinates x and y in the corneal plane.

A second aspect of corneal shape which is of particular interest is *surface curvature*, which

is related to the second derivative of shape. Mathematically, the curvature of a surface is defined by how rapidly a plane which remains tangent to the surface changes orientation as its contact point moves over the surface. A sphere with a small radius has a higher curvature than one with a larger radius, because as the tangent plane moves over the small sphere its orientation changes more rapidly than when it moves over the larger sphere. Curvature is often converted to *power* expressed in dioptric units, although this has little meaning in an optical sense[13].

On a general surface (one without symmetry) the rate of change of the tangent plane's orientation depends on the *direction* in which the contact point moves. Therefore the curvature at a point on a surface depends upon the direction of movement through the point. There is a direction which gives *maximum curvature* and a direction which gives *minimum curvature*. These curvatures are referred to as the *principal curvatures*. The average of the two curvatures is the *mean sphere*, or more simply *sphere*, of the surface, and the difference is the *cylinder*. By examining the overall distribution of curvature across a patient's cornea, properties such as astigmatism may be quantified. Furthermore, local regions of high curvature on the cornea may indicate conditions such as *keratoconus*[15–17].

Many commercially available videokeratographs that claim to measure curvature (or power) in fact report a third aspect of corneal shape which is sometimes referred to as *slope*, related to the first derivative of shape[13]. Due to an incorrect assumption, however, the image analysis algorithms used by these videokeratographs can produce substantial errors when reporting the slope of distorted corneas such as in keratoconus.

In addition to which aspect of corneal shape the analysis algorithms choose to report, they are further distinguished by the *continuity* of their output. It is common practice to output information — be it position, slope, or curvature — as a set of discrete sample points[5,10,18].

Information at points lying between samples must be interpolated. For example, interpolation would be necessary to display a continuous map of curvature over the surface.

The analysis algorithm described in this paper takes an alternate approach. It uses the mathematical technique of *polynomial splines*[19,20] to represent its surface information as a continuous data set. The continuity is maintained and communicated to subsequent display algorithms. The algorithm directly recovers corneal position. It should be noted that from a continuous map of corneal position, it is trivial to recover corneal slope and curvature by computing the appropriate derivatives. The algorithm differs significantly from other current analysis algorithms. It operates by constructing a computer simulation of the videokeratograph and searching for a surface that matches the original cornea[21,22]. It uses many sample points of the videokeratograph image concurrently in its search. This is in contrast to other algorithms that tend to compute information independently at each sample or along meridians only.

Method

The analysis algorithm inputs a videokeratograph image recorded from a physical cornea. Its goal is to output the piecewise polynomial description of a *simulated corneal surface*. The simulated surface is shaped so that the image created, if we could manufacture the surface and place it in the videokeratograph, would match the one recorded from the physical cornea. To ensure that the simulated surface produces the same image, we use a computer simulation of the videokeratograph. Under certain assumptions and constraints, we claim that the simulated surface is a good fit to the physical cornea because the images generated by the two objects match.

The input image was formed by light from the mires reflected at the cornea and gathered

by the video camera. We simulate this process by *backward ray-tracing*, as illustrated in Figure 1. In the simulation, the lens of the video camera is replaced by its equivalent *nodal point*, which becomes the center of projection. The video CCD array that records the image is represented in the simulation by the image plane. We refer to the set of mires as the *source*. Circular mires form *rings* in the image.

(Insert Figure 1 here)

Our algorithm uses an iterative process in which an initial estimate of the surface shape is adjusted until we converge to a solution. Each iteration consists of an *error reduction* phase and a *refinement* phase. The error reduction phase adjusts the shape of the surface to reduce an *error measure* that estimates the difference between the current surface and the real cornea using the ray-traced simulation and the input image. The refinement phase adds degrees of freedom to the surface model as needed for a more accurate fit.

Surface Model

Our algorithm uses a continuous distance function to define the shape of the simulated surface (see Figure 2). The image plane is given a coordinate system defined by axes u and v. There is a ray that originates at each point on the image plane and passes through the nodal point. The distance from the nodal point to the surface (measured along the ray) is given by the distance function $D(u, v)$. The shape of the surface is completely determined by specifying the value of the distance function at every point in a rectangle on the image plane. The rectangle (or domain of the function) is chosen to be just large enough to contain the videokeratograph image.

(Insert Figure 2 here)

We define the distance function with a piecewise polynomial surface. A piecewise surface

is divided into *patches* which join smoothly along their shared boundaries. In our case, the original domain is divided uniformly into a grid of rectangular subregions. Each subregion forms the domain of one surface patch. A patch is defined by a polynomial of maximum degree five in domain coordinates u and v. Across the boundary between two patches the polynomials are continuous up to the fourth derivative. The coefficients of the polynomials are determined from degrees of freedom called *control points*. Figure 3 shows a sample distance function plotted as a height field and its control points. Note that the surface approximates, rather than interpolates, the points. The control points always form a rectangular grid. A surface with $m \times n$ patches has an $(m+5) \times (n+5)$ grid of control points. In the terminology of computer-aided geometric design, this surface is a biquintic tensor product scalar-valued B-spline surface, with uniform knot spacing and no boundary conditions[19]. Further mathematical details are given in the appendix.

(Insert Figure 3 here)

Initialization

The simulated cornea is initialized to a single patch with 36 control points. The control points are set to lie on a paraboloid with dimensions chosen to approximate the front surface of a sphere of radius 9mm. The points are then moved along the z-axis so that the surface interpolates a known point on the measured cornea. The coordinates of this point are either estimated from the inner ring in the image and the focal length of the camera lens, or measured directly by the videokeratograph. Although this initial estimate of corneal shape is in many cases not very accurate, it is a satisfactory starting point for the algorithm.

The Error Reduction Phase

Videokeratographs use image processing techniques to locate features in the image. These

features may be the boundaries or the center lines of circular mires, or the intersections between black and white patches in a dartboard source pattern. We do not perform the image processing ourselves. Instead, we use sample positions output by the videokeratograph.

The important point to note is that features in the image correspond to features in the source. For example, the edge between a black and a white ring in the image is the reflection of the edge between corresponding black and white circular mires in the source. Our aim is to adjust the simulated corneal surface so that in the ray-traced simulation of the videokeratograph, each sampled feature is the image under reflection of a corresponding feature in the source.

The algorithm uses a subset of the available samples. Approximately 25 times as many samples as surface control points are selected. They are uniformly distributed over the image. Re-selection takes place only when the number of control points is changed by the refinement phase.

For each sample in the subset, we locate all points in the source pattern that could be reflected to the image sample. Each point on a ring could have originated from any point on a mire, depending on the shape of the reflecting surface. Therefore, for any one sample point, the set of possible source points includes the entire mire. For a source pattern made up of patches, as in the dartboard pattern, sample points lying at the patch corners are matched to a single point in the source. We refer to the mire or point generically as the *source curve*. Having matched an image sample to its source curve, we trace a primary ray from the image sample, through the nodal point, to its point of intersection with the simulated surface. This operation is made particularly simple by our choice of surface representation. The origin of the ray is specified by its coordinates on the image plane. Evaluating the distance function $D(u, v)$ at those coordinates gives the distance along the ray from the nodal point to the surface. The geometry of the ray and this distance give the spatial coordinates of the point

on the surface (see Figure 2). The functions that map image plane coordinates to surface position are:

$$\begin{aligned} x(u,v) &= \frac{-u}{\sqrt{u^2+v^2+L^2}} D(u,v) \\ y(u,v) &= \frac{-v}{\sqrt{u^2+v^2+L^2}} D(u,v) \\ z(u,v) &= \frac{L}{\sqrt{u^2+v^2+L^2}} D(u,v). \end{aligned}$$

The polynomial coefficients in $D(u,v)$ are related linearly to the surface control points (see Appendix). For an image sample at a fixed (u,v), the three surface position functions are also linear in the control points. Therefore the surface intersections of rays from many fixed sample points are evaluated simultaneously by multiplying three matrices of coefficients (one for each of x, y and z) with a vector of control points.

A secondary ray is generated for each primary ray by reflecting it at the surface intersection point. The secondary ray lies in a plane containing the primary ray and the surface normal vector. The normal vector is computed as the cross product of two surface tangent vectors:

$$\left[\frac{\partial x(u,v)}{\partial u}, \frac{\partial y(u,v)}{\partial u}, \frac{\partial z(u,v)}{\partial u}\right] \quad \text{and} \quad \left[\frac{\partial x(u,v)}{\partial v}, \frac{\partial y(u,v)}{\partial v}, \frac{\partial z(u,v)}{\partial v}\right].$$

Since $D(u,v)$ is a continuous piecewise polynomial function, its partial derivatives are also piecewise polynomial with coefficients depending linearly on the surface control points. Therefore multiplying a series of constant matrices with a vector of control points efficiently yields the tangent vectors for a fixed set of sample points.

Each secondary ray is traced to its intersection point on the geometry containing the source pattern. If the point does *not* lie on the source curve matching the origin of the primary ray, then the simulated surface has an error in its shape. We have identified two alternative formulae for quantifying the error. First we compute the point on the source curve that is closest to the actual intersection. This is the *desired intersection*. One error measure is the

squared distance in z between the actual and the desired intersection. The other measure defines what we call the *desired normal vector*. This is the vector that, if used in place of the current surface normal, would reflect the secondary ray to the desired, rather than the actual, intersection. The error is measured as the dot product of the current surface tangent vectors (the partial derivatives given above) with the desired normal. These quantities go to zero when the actual surface normal is equal to the desired normal.

These two formulae generate different methods for reducing the surface error. Both methods change the shape of the surface by manipulating the control points so that the secondary rays are pushed towards their desired intersections. Keep in mind, however, that the desired intersections are recomputed at each iteration and may migrate around the source curves.

The method which measures error based on squared distance is discussed in detail in an earlier paper[21]. Summing the errors over all image samples produces a scalar value which is greater than zero for all surfaces with an incorrect shape. We apply standard non-linear minimization techniques to the problem[23]. The variables in the minimization, which are the control points of the surface, are manipulated to follow the gradient of the error function towards zero. As reported in the paper, this algorithm was accurate but slow.

We now describe a new algorithm that computes errors using the vector dot product. We present the overall concept — mathematical details are reported elsewhere[a]. As mentioned above, the method computes a desired normal at the intersection point of each primary ray with the surface. The dot product between the tangent vectors and this desired normal measures the surface error. Once the desired normal for a sample has been determined, the dot product with a tangent vector is expressed as a linear combination of the surface control points. This follows from a previous claim that the surface tangents themselves can be expressed as linear combinations. Therefore the error at many image samples is computed

by the matrix multiplication:

$$\mathbf{e} = \mathbf{M}.\mathbf{p},$$

where $\mathbf{e}$ is a vector of error values, $\mathbf{M}$ is a matrix of constant coefficients derived from the fixed sample positions, and $\mathbf{p}$ is a vector of control point values. We use many more sample points than control points, so $\mathbf{M}$ has more rows than columns. We wish to reduce the error to zero, so it makes sense to seek a solution to the equation:

$$\mathbf{M}.\mathbf{p} = \mathbf{0}.$$

Unfortunately, one solution to this equation is to set all the control points in $\mathbf{p}$ to zero. To avoid this trivial solution we force the surface to interpolate a known point. This is the point that was mentioned in the section on initialization and is measured by the videokeratograph or estimated from the image. The interpolation constraint is linear in the control points. Therefore the previous equation is rewritten in the form[a]:

$$\bar{\mathbf{M}}.\bar{\mathbf{p}} = \mathbf{d}.$$

where $\bar{\mathbf{p}}$ has one fewer entries than $\mathbf{p}$. We find the least squares solution to this overconstrained problem. The new control points define a surface that approximates the desired normals and interpolates the known point. The overall surface error is reduced.

This section has described the error reduction phase of a single iteration of the algorithm. The phase is repeated many times before an accurate solution is found (see the Results section for examples).

Refinement

The algorithm takes on the order of minutes to complete (details follow in the Results section). However, our use of a *refinement strategy* means that we can display a fairly good

map of corneal topography in seconds. While the map is viewed and the algorithm runs to completion in the background, the display updates with more accurate data. The refinement strategy is formulated so that the initial search steps execute quickly at the expense of accuracy. The surface at this early stage is defined by a single patch. This minimizes the number of control points and the number of image samples required during error reduction. Accuracy is improved at the expense of iteration time by refinement: increasing the number of patches and selecting a larger set of samples. The algorithm decides when to refine by comparing the sum of the squared error values (the dot products) after each iteration to the sum from the previous iteration. Refinement occurs when the difference falls below 1% of the sum. Refinement divides each patch into four subpatches. Dividing the domain rectangle of each old patch horizontally and vertically through the center provides domains for four new patches. Control points are computed so that the union of new patches equals the original surface[20]. Now the shape of the surface is unchanged but it has more degrees of freedom. Using refinement, the surface moves through representations using $(2 \times 2, 4 \times 4, 8 \times 8, \ldots)$ patches until the error difference falls below a predetermined threshold at a predetermined level of refinement.

Testing

For the results reported in this paper, we terminated the search at 8×8 patches when the error difference was less than 1%.

The algorithm was tested on four images. Three of the images were collected by a videokeratograph from black plastic hemispheres of radius 7mm, 8mm, and 9mm. The hemispheres were clamped into position and imaged using the standard procedure. The fourth image was generated synthetically by a software simulation of the videokeratograph. The object used

to generate the image was an ellipsoid defined by the equation

$$\frac{x^2}{8^2} + \frac{y^2}{9^2} + \frac{z^2}{10^2} = 1,$$

where x, y, and z are measured in millimeters and the axes are oriented as in Figure 2. The ellipsoid is translated along the z-axis so that it is illuminated by the source pattern and the simulated image is in focus. The advantage of using simulated images is that irregular objects that would be difficult to manufacture can still be tested.

The algorithm was implemented in Matlab[b] and the tests reported here were executed on a midrange Unix workstation. We estimate that similar results could be accomplished by a fast Apple PowerPC or Pentium based personal computer. The time spent by the algorithm at each level of refinement was recorded. Before each refinement and at the conclusion of the algorithm, the shape of the simulated surface was compared to the true surface used to create the input image. First the domain rectangle of the surface was covered by a 50×50 grid of sample points. Those sample points that fell outside the peripheral image ring were discarded due to the lack of sensible data in that region. The surface position functions $x(u, v)$, $y(u, v)$, and $z(u, v)$ were evaluated at the remaining sample points. The depth (in z) was then subtracted from the depth of the true surface at the corresponding position (in x and y) to give an error at each sample. To eliminate any global offset between the simulated and true surfaces, the mean error was subtracted from the individual errors. Finally the root mean square (RMS) error value was recorded.

Results

Table 1 lists the RMS error in depth for the synthetic ellipsoid recorded at each level of refinement. The number of patches in the surface is $n \times n$, where $n = 1, 2, 4, 8$. Table 2 lists the RMS error for the three physical spheres at the conclusion of the algorithm (8×8

patches). Timing results for the spheres appear in Tables 3–5. We list the number of iterations per refinement level, total time for each level, and the time per iteration.

(Insert all tables here)

Discussion

The algorithm we have presented uses a new technique to recover the shape of a cornea from a videokeratograph image. The correspondence between a simulated surface and the real cornea is improved iteratively by using a difference estimate computed using backward ray-tracing. As stated in the introduction, algorithms that treat each meridian of image data independently are prone to error when the surface is asymmetric. Since our algorithm uses data from all meridians to reconstruct a continuous map of surface position, asymmetric surfaces should be handled correctly.

Our goals were to produce an algorithm which gave a good fit to empirical data, dealt correctly with asymmetric surfaces, and was fast enough to be practical in clinical use. To see how well these goals were met, we performed a series of tests.

The algorithm has shown an error of less than 1 micron for real images, which we find encouraging. There is a difference in the accuracy of the algorithm for simulated versus physical test objects. This suggests errors in the various measurement processes. Possibilities are the extraction of features in the image and the measurements we have made of the videokeratograph setup (such as the geometry of the mires).

The high accuracy of the results reported for the simulated ellipsoid indicate that the algorithm handles asymmetric objects and overcomes the problems discussed in the introduction.

The algorithm takes on the order of minutes to complete. We believe this is appropriate for

clinical use. Furthermore, through the use of refinement, the algorithm produces adequate results in less than a minute. The number of refinement levels can be tailored to the situation. For the results reported in this paper, we terminated the algorithm at the 8×8 refinement level. This gives adequate accuracy, although there is no reason why we cannot go to the next level of 16×16 patches. Beyond that we are reaching the limits of the feature extraction process. Usually the features are not uniformly spread across the image but are concentrated along rings. This limits how small the patches can be, because if a patch falls between feature clusters it will be unconstrained (except by continuity between adjacent patches). Also, by fitting a splined surface to the data, we are effectively smoothing any errors in the measurement of the features. As the number of patches is increased, the fit to the data becomes tighter but the smoothing effect is reduced. The new algorithm is more sophisticated than currently available algorithms because it constructs a continuous map of corneal position. From this map, it is simple to compute other common slope- or curvature-based representations of shape. One danger is that these computations require taking derivatives of the surface position function, an operation that magnifies irregularities in the surface. Therefore, there is also a tradeoff between how well the data can be fit and the utility of these other representations.

We are currently performing experiments to confirm the accuracy of the algorithm with more complex objects. We are also working to produce data comparing the algorithm to existing algorithms for objects with asymmetry or localized areas of high curvature.

Acknowledgments

The authors would like to thank Lillian Chu for her help in implementing the algorithm and generating results.

Footnotes

[a]Internal report available from the authors.

[b]Matlab is an interpreted mathematical language available from The Mathworks, Inc., 24 Prime Park Way, Natick, MA 01760-1500.

References

[1] Klyce SD. Computer-assisted corneal topography, high-resolution graphic presentation and analysis of keratoscopy. Invest Ophthalmol Vis Sci 1984; 25:1426–35.

[2] Koch DD, Foulks GN, Moran T. The corneal EyeSys system: accuracy analysis and reproducibility of first-generation prototype. Refract Corneal Surg 1989; 5:424–9.

[3] Mammone RJ, Gersten M, Gormley DJ, Koplin RS, Lubkin VL. 3-D corneal modeling system. IEEE Trans Biomedical Eng 1990; 37:66–73.

[4] Wang J, Rice DA, Klyce SD. A new reconstruction algorithm for improvement of corneal topographical analysis. Refract Corneal Surg 1989; 5:379–87.

[5] Warnicki JW, Rehkopf PG, Curtin SA. Corneal topography using computer analyzed rasterographic images. Am J Opt 1988; 27:1125–40.

[6] Wilson SE, Klyce SD. Advances in the analysis of corneal topography. Surv Ophthalmol 1991; 35:269–77.

[7] Klyce SD, Dingeldein SA. Corneal topography. In: Masters BR. Noninvasive Diagnostic Techniques in Ophthalmology. New York: Springer-Verlag, 1990:61–77.

[8] Mandell RB. The enigma of the corneal contour CLAO J 1992; 18:267–73.

[9] Bibby MM, Townsley MG. Analysis and description of corneal shape. CL Forum 1976; 1(8):27–35.

[10] Doss JD, Hutson RL, Rowsey JJ, Brown DR. Method for calculation of corneal profile and power distribution. Arch Ophthalmol 1981; 99:1261–5.

[11] van Saarloos PP, Constable I. Improved method for calculation of corneal topography for any photokeratoscope geometry. Optom Vis Sci 1991; 68:960–6.

[12] Roberts C. Characterization of the inherent error in a spherically-biased corneal topography system in mapping a radially aspheric surface. Refract Corneal Surg 1994; 10:103–11.

[13] Klein SA, Mandell RB, Barsky BA. Representing corneal shape. Vision Science and its Applications. Washington, DC: Optical Society of Am,1995.

[14] Mandell RB, Klein SA, Shie CH, Barsky BA, Yang Z. Axial and instantaneous radii in videokeratography. Invest Opthalmol Vis Sci 1994; 35 Suppl.:2079.

[15] Barsky BA, Mandell RB, Klein SA. Corneal shape illusion in keratoconus. Invest Opthalmol Vis Sci 1995; 36 Suppl.:5308.

[16] Maguire LJ, Bourne WD. Corneal topography of early keratoconus. Am J Ophthalmol 1989; 108:107–12.

[17] Mandell RB, Barsky BA, Klein SA. Taking a new angle on keratoconus. Contact Lens Spect 1994; 9(4):44–7.

[18] Belin MW, Litoff D, Strods SJ. The PAR technology corneal topography system. Refract Corneal Surg 1992; 8:88–96.

[19] Bartels RH, Beatty JC, Barsky BA. An Introduction to Splines for Use in Computer Graphics and Geometric Modeling. San Francisco: Morgan Kaufmann Publishers Inc., 1987.

[20] Cohen E, Lyche T, Riesenfeld RF. Discrete B-splines and subdivision techniques in computer-aided geometric design and computer graphics. Comp Graph Image Proc 1980; 14(2):87–111.

[21] Halstead MA, Barsky BA, Klein SA, Mandell RB. Geometric modeling of the cornea using videokeratography. In: Daelhen M, Lyche T, Schumaker LL, eds. Mathematical Methods for Curves and Surfaces. Nashville: Vanderbilt University Press, 1995.

[22] Klein SA, Halstead MA, Barsky BA, Mandell RB. Corneal topography for general surfaces. Invest Opthalmol Vis Sci 1994; 35 Suppl.:2079.

[23] Gill PE, Murray W, Wright MH. Practical Optimization. San Diego: Academic Press, 1993.

Appendix A. B-spline Surfaces

We represent our distance function $D(u, v)$ using B-splines — a common representation in the field of computer-aided geometric design (CAGD). The B in B-spline stands for basis. This alludes to the fact that $D(u, v)$ is defined by a sum of scaled *basis functions*:

$$D(u, v) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} c_{ij} b_i(u) b_j(v).$$

The coefficients c_{ij} are called *control points*, and are arranged on an $m \times n$ grid. The $b_i(u)$ and $b_j(v)$ are the basis functions and are defined as piecewise polynomials. In the general case, each basis function has a different shape. The shape is controlled by *knot sequences* which segment the set of reals. However, we simplify the situation somewhat by using uniform knot sequences. This means that each basis function is simply a translated copy of a standard basis function $b(u)$. For example, $b_i(u) = b(u - i)$. The domain of the function is defined by $u \in [0, m)$ and $v \in [0, n)$. If we require a different domain, then we must scale and translate u and v before evaluating the B-spline.

Figure 4 shows the standard basis function. Over each unit interval, it is defined by a polynomial of degree 5 in u. The polynomials are chosen so that they are continuous to the fourth derivative at their boundaries. The function $b(u)$ is non-zero over 6 unit intervals. Therefore, when $D(u, v)$ is evaluated for a given (u, v) at most 36 of the basis functions $b_i(u)$ and $b_j(v)$ will be non-zero. We discard from the domain those boundary regions for which there are fewer than 36 basis functions. In CAGD terms this means we have no boundary conditions. For the remaining domain, each unit square over which the same 36 basis functions are non-zero defines a *patch*. There is a unique 6×6 subgrid of control points c_{ij} which affect this patch. The subgrids of adjacent patches overlap. For example, a subgrid shares all but its rightmost column of points with the subgrid of the patch to the left.

(Insert Figure 4 here)

Over each patch, the 36 non-zero basis functions that are scaled by the appropriate c_{ij} can be summed according to the equation given above. This results in a single biquintic polynomial with coefficients which are linear combinations of the 36 control points. Further details on B-splines are available in the literature[19].

Contact Address

Mark Halstead or Brian Barsky

Computer Science Division

Department of Electrical Engineering and Computer Sciences

University of California

Berkeley, California 94720-1776

n	*RMS error*
1	4.1×10^{-2}mm
2	5.5×10^{-3}mm
4	1.7×10^{-4}mm
8	8.5×10^{-6}mm

Table 1: RMS error for a simulated ellipsoid according to the number of patches ($n \times n$).

Sphere Radius	*RMS error*
7mm	7.0×10^{-4}mm
8mm	8.0×10^{-4}mm
9mm	1.4×10^{-3}mm

Table 2: RMS error for physical spheres when 64 (8×8) patches were used.

n	*Iterations*	*Total time (secs)*	*Time per iteration (secs)*
1	24	10.9	0.5
2	9	9.1	1.0
4	10	32.0	3.2
8	7	191.6	27.4

Table 3: Timing results for 7mm radius sphere.

n	*Iterations*	*Total time (secs)*	*Time per iteration (secs)*
1	20	9.0	0.5
2	9	9.1	1.0
4	10	32.0	3.2
8	6	167.6	27.9

Table 4: Timing results for 8mm radius sphere.

n	*Iterations*	*Total time (secs)*	*Time per iteration (secs)*
1	6	2.6	0.4
2	14	13.7	1.0
4	4	12.3	3.1
8	4	112.6	28.2

Table 5: Timing results for 9mm radius sphere.

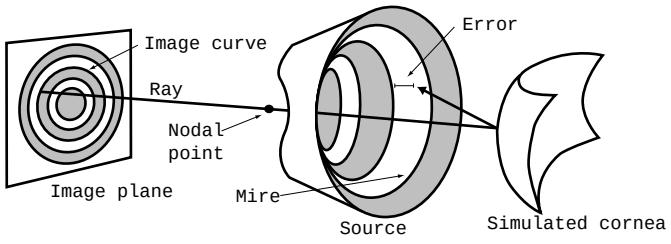

Figure 1: Backward ray-tracing determines the error in surface shape for a given image sample point.

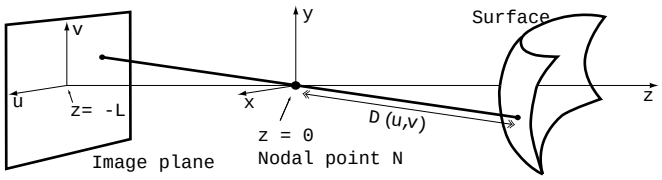

Figure 2: Surface representation scheme.

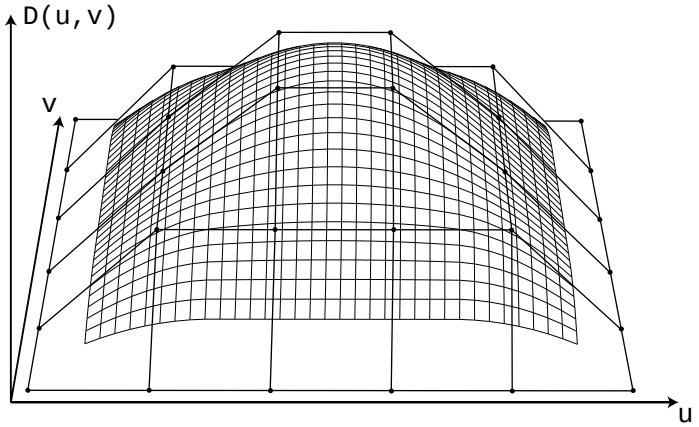

Figure 3: Distance function represented as a B-spline surface shown with control points.