

ADAPTIVE SURFACE MEASUREMENT

REAL-TIME, ADAPTIVE MEASUREMENT OF CORNEAL SHAPES

Conventional tools for measuring the shape of the cornea perform poorly when applied to abnormal eyes. The image processing regularly fails, and the shape reconstruction often produces inaccurate results. This article describes a single measurement instrument that could integrate real-time solutions to both problems.

The outermost layer of the human eye—the cornea, see Figure 1—is of tremendous importance to good vision. By the early 19th century, physicians recognized the cornea's role in the refraction process. In the present day, several types of refractive surgery—such as corneal transplants and laser adjustment of the cornea—have become well established as techniques for improving a patient's sight. To support these types of surgery, it is essential to have accurate techniques for measuring corneal shape. However, the systems available for this task have some serious shortcomings. This article describes how we can use adaptive surface measurement and parallel cluster computing to improve corneal measurement instruments.

The conventional approach and its problems

Most instruments that measure corneal topography do so by using the eye to *mirror* a pattern. Figure 2a shows the cross section of such an in-

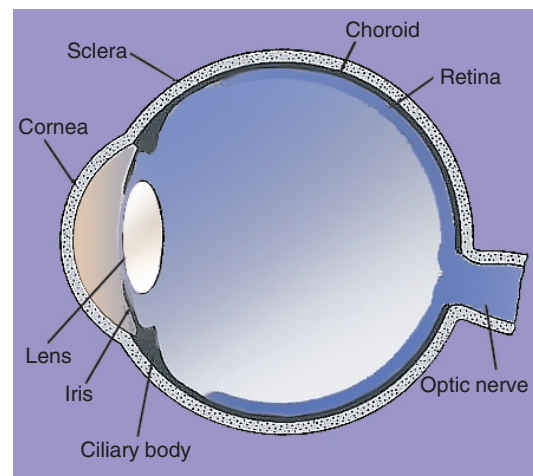


Figure 1. Cross section of the human eye. The cornea constitutes the most anterior part.

strument.^{1,2} It consists of a cylinder closed at one end containing a pattern (the *stimulus*) that is brightly lit from the back. The cornea to be measured is positioned in front of the cylinder's open end, while a camera behind the closed end registers the reflection (Figure 2b). The instrument reconstructs corneal shape from the distortions in the reflected image.

The flow diagram in Figure 3 depicts the consecutive stages in the data processing. The first step after data acquisition is *pattern recognition*, which involves uniquely localizing and identifying certain positions in the input image (the reflection of the stimulus). Subsequently, the tool

1521-9615/02/\$17.00 © 2002 IEEE

FRANS M. VOS

Delft University of Technology and the Academic Medical Center

HANS J.W. SPOELDER, DESMOND M. GERMANS,

RUTGER HOFMAN, AND HENRI BAL

Vrije University

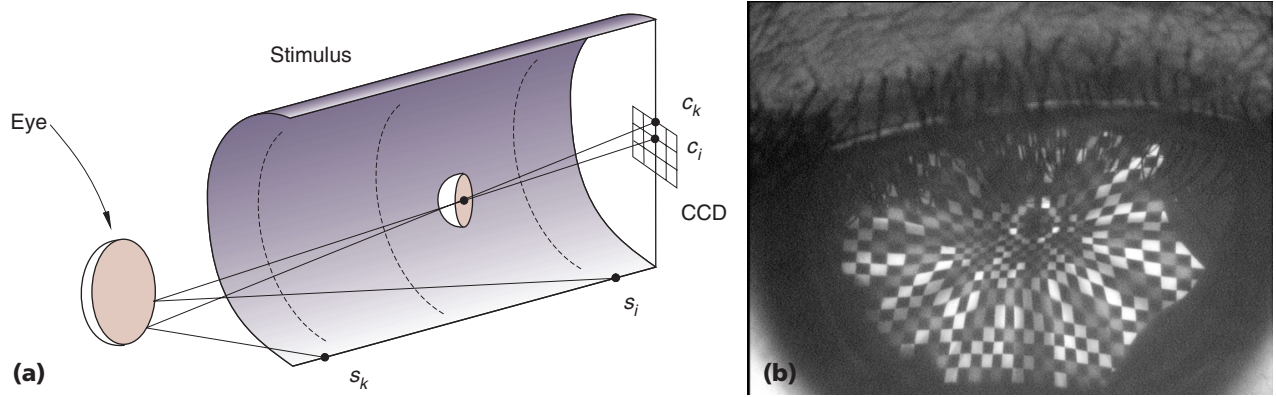


Figure 2. Conventional technology: (a) crosscut of a cornea topograph. The eye mirrors a pattern (stimulus), brightly lit from the back by neon tubes. (b) A charge-coupled device camera (CCD) registers the reflection.

models corneal shape from a correspondence of points on the stimulus and the sensor device; this step is called *shape reconstruction*.^{2,3} In the last step, *visualization*, the estimated shape is visualized in a graphical user interface.⁴

However, this conventional measuring technique does not work ideally. First, the stimulus is designed to yield a certain desired reflection specifically with the average, normal cornea. Pathological eyes, however, produce reflections with severe, unpredictable distortions that can foil the pattern recognition software and cause the measurement to fail. Second, to support use in clinical practice, these tools generally implement the shape reconstruction step with a lookup table approach,³ which can lead to severe errors.^{5,6} Nonlinear techniques have demonstrated greater accuracy but are not popular because of their poor execution time.

Improving the conventional

To address the problem the conventional approach has with abnormal corneas, we will look into the development of a measurement system that adapts the stimulus to the cornea's shape. The adaptation aims to nearly always register a regular pattern, even with deformed shapes. Figure 4 gives a functional sketch of a system implementing this approach.

Clearly, any approach must also incorporate accurate shape modeling—through nonlinear shape reconstruction, for example. Moreover, the entire process must permit real-time execution to allow practical applications such as surgery. This means that the total measurement time should be on the order of a few seconds. (Such a “soft” real-time constraint is acceptable because the eye is fixated during intervention.) Meeting this prerequisite while using accurate modeling techniques is a

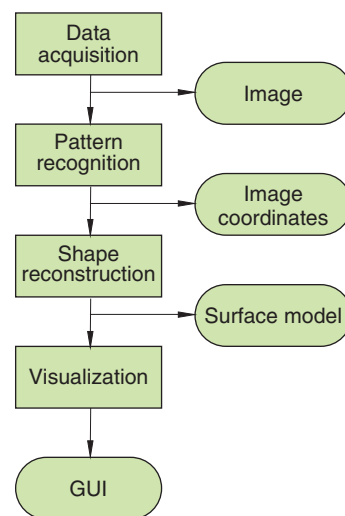


Figure 3. Measuring the shape of the cornea: a schematic rendering of the data-processing steps, from data acquisition to the representation of the results in a graphical user interface.

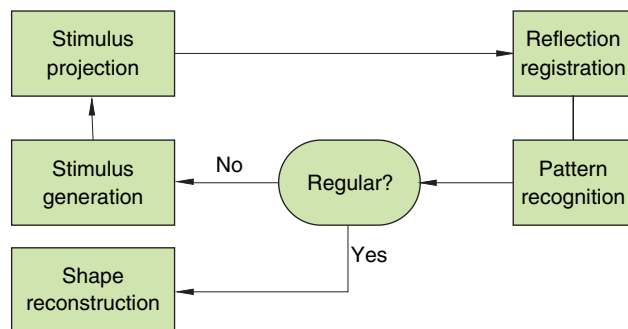


Figure 4. Adaptive system for corneal measurement, functional scheme.

challenging problem that we address through parallel cluster computing.

Stimulus adaptation

As we mentioned, existing cornea topographs

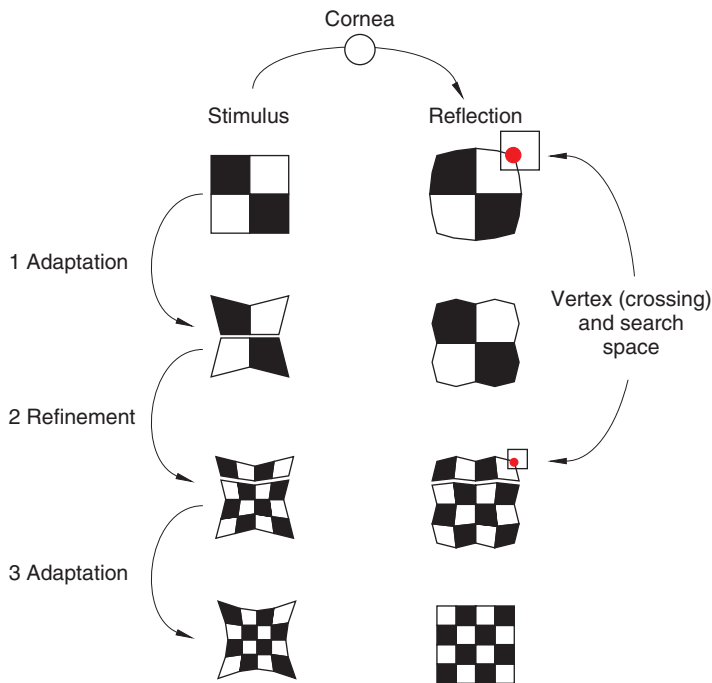


Figure 5. Graphical representation of the proposed algorithm. Successive steps deform the stimulus pattern to make the reflection regular and then refine the pattern. The search space for a vertex becomes smaller by a factor of two after refinement; the regularization allows better vertex position prediction.

use a fixed stimulus pattern that yields the desired reflection for the average, normal eye but distorted reflections for pathological corneas. When the distortion exceeds the dynamic range of the pattern recognition software, the measurement fails. To obviate this problem, we can adjust the stimulus to fit individual eye characteristics. We propose a scheme in which the topograph iteratively adapts the stimulus to arrive at a desired pattern (Figure 5). The obvious benefit is that this scheme results in extremely robust pattern recognition. For such a system to be useful in surgical practice, however, the adaptation must take place in real time.

In earlier studies, we demonstrated the usefulness of a *chessboard* stimulus (see Figure 2b)⁷ composed of quadrilaterals sharing lines between common vertices (that is, at the crossings). In this stimulus, adjustable parameters are the number of quadrilaterals and the positions of the vertices.

Here's how stimulus adaptation works: In its most simplified state, the prescribed pattern consists of four squares (see Figure 5), giving only nine vertices in the image. Once we recover the feature locations from the image, we adjust the stimulus pattern by moving each vertex (which we'll describe later). As soon as a regular chessboard is registered,

we refine the number of quadrilaterals, going from 2^2 to 4^2 to 8^2 , and so on. (For optimal tuning, we will consider implementing partial refinement in the future.) Stimulus shape and reflected image are inversely related in this process, as regular squares become deformed quadrilaterals and vice versa.

A priori assumptions on the object shape can limit the search space for each vertex. For instance, we can calculate the center of such a search area as the location upon reflection from the average human cornea (a sphere with a radius of 8 millimeters). The maximal variation in object shape determines the outer limits. To locate the vertices, we use a matched-filter approach.¹ Because we can expect a better location prediction as the adaptation proceeds, we can halve the search space each time we refine the pattern. Accordingly, we can also reduce the size of the matched filter.

Thus, we come to the following algorithm:

1. Generate a stimulus with four squares.
2. Register its reflection from the object (eye) to be measured and find the vertices in the image (using the matched-filter technique¹).
3. Adjust the positions of the vertices to minimize the deviation from the desired geometry (a regular chessboard).
4. Increase the number of quadrilaterals and repeat steps 2 and 3 until you cannot recover one of the quadrilaterals' vertices or until the number of quadrilaterals exceeds a preset threshold.

To regularize the pattern, we adopt a strategy in which the vertices' positions are iteratively adjusted as follows (Figure 6). Suppose that at first the k th vertex on the stimulus (\mathbf{s}_k^0) is reflected to a position (\mathbf{c}_k^0) on the CCD. Additionally, let (\mathbf{c}_k^t) be the target point in a regularized reflection. Initially, the vertex's position in the stimulus is changed with a fixed (small) step (δ_k^0). Consequently, $\mathbf{s}_k^1 = \mathbf{s}_k^0 + \delta_k^0$, a point that will correspond to position \mathbf{c}_k^1 in the CCD. Subsequently, we can obtain a better estimate for the step vector by linear extrapolation:

$$\delta_k^1 = \delta_k^0 \cdot \frac{\mathbf{c}_k^t - \mathbf{c}_k^1}{\mathbf{c}_k^1 - \mathbf{c}_k^0} \quad (1)$$

The division and the product in Equation 1 address vector elements individually to yield a new vector.

Thus, δ_k^1 can update \mathbf{s}_k^1 , after which the process repeats until deviation from the desired position falls below a preset threshold.

Shape reconstruction

Real-time, adaptive corneal measurement requires fast shape reconstruction. Practitioners rarely use nonlinear reconstruction techniques in practice because of their slow execution. A solution to this problem is to parallelize the code. This section will focus on the algorithm, and we'll later evaluate its implementation.

Basic theory

To represent corneal shape, we follow Marc Halstead and colleagues,⁴ using a biquintic tensor B-spline surface as a model. Such a B-spline surface (h) is defined as a piecewise description, which we can think of as a patchwork quilt. Each piece of surface—each patch—is a polynomial of preset degree; in our application, the degree is 5. The sum of scaled basis functions gives the full surface:

$$h(\mathbf{d}, \mathbf{c}_k) = \sum_{i=1}^{m_1+n} \sum_{j=1}^{m_2+n} \mathbf{d}_{ij} \mathbf{b}_{ij}(\mathbf{c}_k) \equiv \mathbf{d} \cdot \mathbf{b}, \quad (2)$$

where m_1 and m_2 are the numbers of patches; n the degree of the spline (for us, 5); \mathbf{c}_k the independent variables; \mathbf{b}_{ij} the spline basic function; and \mathbf{d}_{ij} their weights. In our approach, \mathbf{c}_k corresponds to the positional coordinates of a vertex in the CCD. After the double sum's linearization, surface function evaluation merely involves the vector product of parameters \mathbf{d} and basic functions \mathbf{b} .

To estimate the parameter vector \mathbf{d} , we need a function that expresses how well a given representation models the data. The "Definition of the Residual Function" sidebar contains a proper definition of the one we use; for now, let's consider it an abstract function of the parameters that sums a residual function r over all vertices,

$$SSE(\mathbf{d}) = \sum_k r(\mathbf{d}, \mathbf{c}_k). \quad (3)$$

We estimate the model's parameters by minimizing Equation 3. To this end, standard nonlinear regression techniques are available.⁸ All these approaches iteratively update the parameters from a given starting point to let the sum of the squared errors converge to a minimum. In the Levenberg-Marquardt method that we use, we find the direction vector $\boldsymbol{\tau}$ that updates the current parameter by solving the equation,

$$(J^T J + \lambda I) \boldsymbol{\tau} = J^T (\mathbf{r}(\mathbf{d}^j, \mathbf{c}_k)). \quad (4)$$

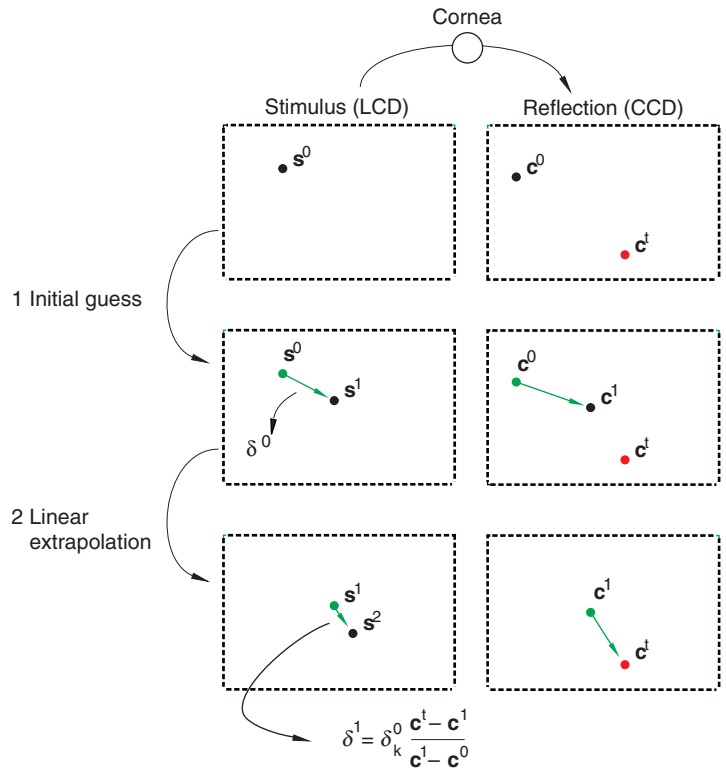


Figure 6. Iterative adjustment of a vertex in the stimulus to obtain recovery in the target position in the CCD. Initially, a fixed step modifies the position. Subsequently, we apply linear extrapolation. (For simplicity, this drawing omits index k , used in the text.)

Here, $\mathbf{r}(\mathbf{d}^j, \mathbf{c}_k)$ represents the residual vector (considering each vertex \mathbf{c}_k in the CCD) for the estimate of the parameters \mathbf{d}^j after j iterations. J is the Jacobean matrix of the residual function that we can calculate using a finite-differencing method, and λ is a weighting parameter that is conventionally taken small. To calculate the inverse of $J^T J$, we first determine a QR decomposition of J ($J = QR$), after which

$$(J^T J)^{-1} J^T = R^{-1} Q^T. \quad (5)$$

Time complexity

Later, we'll demonstrate how to implement the minimization of the residual function in parallel. But first, to set an objective for the speedup to be gained, we'll explore the performance of a sequential implementation. We can easily partition the algorithm into the four main segments that consume most of the reconstruction time (t_r):

1. Initialization—precomputation of the spline basic functions. (See Equation 2: because each position \mathbf{c}_k is fixed, we can precompute \mathbf{b}_{ij} .)

Definition of the Residual Function

The starting point for modeling corneal shape is an unambiguous point-to-point relation of positions at the stimulus and at the sensor device. At the sensor site, these positions are defined as the crossings in the chessboard, which should be recovered through image processing. To establish the aforementioned relation, such crossings must be uniquely identified. In static studies—that is, when only a single registration of the pattern is available—redundant information (such as PRBA color encoding) is necessary to enable this.¹ With an adaptive approach, proper tracking of the sequence of patterns should allow the identification, ruling out the necessity of color coding. At the stimulus, the calibration must determine the exact position of vertices.

Suppose a position s_k on the stimulus is reflected on the cornea at a point c_k on the charge-coupled device camera (CCD, see Figure A).

Given a surface model with parameters d , we can calculate a point s'_k on the stimulus that would also be registered at c_k when mirrored to the model. We do this by simple backward ray-tracing. First, we calculate the position where a primary ray through c_k and the lens center crosses the surface model. Subsequently, we determine the surface normal in this

position. We generate a secondary, reflected ray by mirroring the primary ray around the surface normal, after which we find s'_k as the intersection of the secondary ray with the stimulus.

We define the residual function as the distance between s_k and s'_k summed over all unique points:

$$SSE(d) = \sum_k \|s_k - s'_k\|^2.$$

Reference

1. F.M. Vos et al., "A New PRBA-Based Instrument to Measure the Shape of the Cornea," *IEEE Trans. Instrumentation and Measurement*, vol. 46, no. 4, Aug. 1992, pp. 794–797.

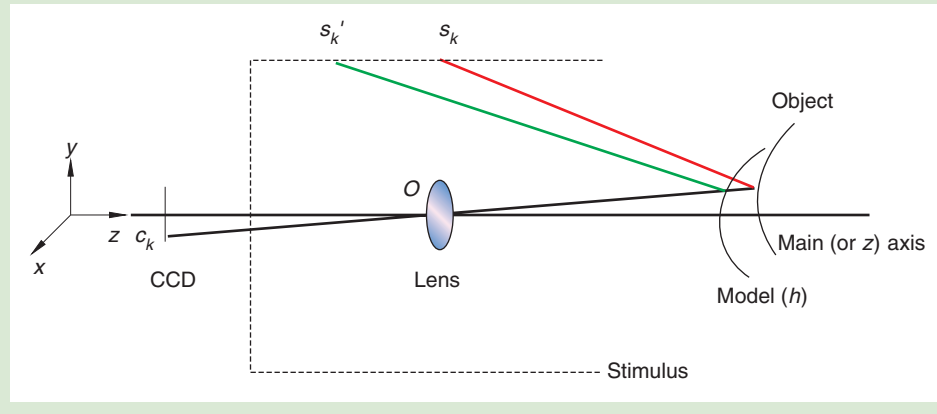
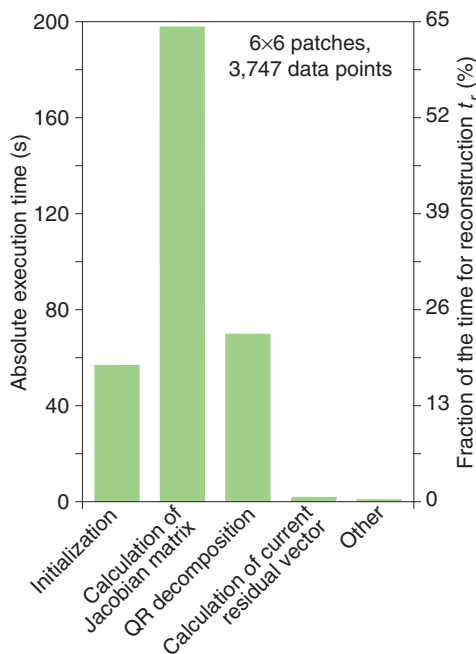


Figure A. Outline of the residual model function: s_k is a point (observation) on the stimulus that is registered at c on the CCD (independent variable); s'_k also results in registration at c when mirrored to an instance of the model function h . The lens center is the coordinate system's origin (O).

Figure 7. Distribution of the execution time over the main parts of the shape reconstruction program for a representative case.



2. Calculation of the Jacobian matrix \mathcal{J} (in Equation 4).
3. Calculation of the QR decomposition of the Jacobian matrix (Equation 5).
4. Calculation of the current residual vector $\mathbf{r}(d^j, c_k)$ (in Equation 4).

We execute segments 2 through 4 of this process once every iteration to update the parameter vector (by τ , through Equation 4). Consequently, t_r is given by

$$t_r = t_{init} + n \cdot (\bar{t}_{Jacobian} + \bar{t}_{QR} + \bar{t}_{res}) + t_{rest} = t_{init} + n \cdot \bar{t}_{iter} + t_{rest}, \quad (6)$$

in which t_x is the execution time for program segment x , and the overbar denotes normalization by the number of iterations n (to update the parameter vector d^j). The normalized time per

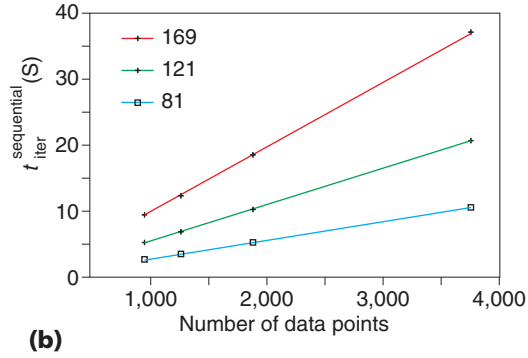
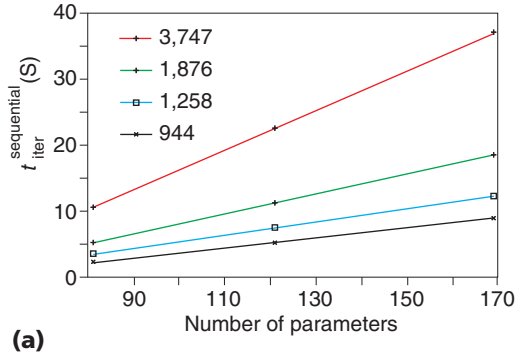


Figure 8. The time per iteration ($t_{iter}^{sequential}$) as a function of (a) the number of parameters and (b) the number of data points.

iteration is given by t_{iter} .

Figure 7 shows the performance of these parts of the algorithm (measured on a 200-MHz Pentium Pro configuration) in absolute time and as a percentage of the total. We obtained this data by modeling a test object using 121 parameters (that is, 6×6 patches). The stimulus contained 64×64 squares. The algorithm missed vertices only in the periphery of the reflected pattern, yielding 3,747 data points. (The maximum number that can be detected is $(64 + 1)^2 = 4,225$.) In this representative case, total execution time was 322 seconds, comprising 12 iterations.

Now let's turn to parallelization. Clearly, the time spent in the iterations—to calculate the Jacobian matrix and its QR decomposition—consumes the largest part of the total execution time. The initialization segment is trivial to parallelize, yielding a linear speedup, so we'll focus on calculating \mathcal{J} and the QR decomposition.

The algorithm's execution time is a function of the numbers of input data points and parameters (which determine the Jacobian matrix's height and width). Previous practical experiments using a fixed stimulus resulted in 1,000 to 4,000 data points.¹ The shadow of the patient's nose and forehead can partially occlude the reflection (see Figure 2), so the algorithm detects only a fraction of all vertices. The shape to be measured determines the number of parameters. A B-spline surface consisting of 4×4 patches can well approximate a regular surface such as a sphere. Pathological surfaces may require as many as 8×8 patches.

We varied the data size experimentally by sampling every second, third, and fourth point from the 3,747-element data set described earlier. Consequently, the original set was down-sampled to 1,876, 1,258, and 944 elements. Figure 8 shows that the relation between the time per iteration (t_{iter}) versus the numbers of data points and parameters is approximately linear.

The whole system's performance should be on the order of a few seconds to allow practical measurement. To meet this requirement, each iteration should execute on a subsecond scale. For example, to take 0.5 seconds per iteration, we would need a speedup of approximately 80 for the case we described (169 parameters and 3,747 data points). Henceforth, we'll use SU_x to denote the quotient of sequential and parallelized execution time of program part x —that is, the speedup achieved through parallelization.

Feasibility

The feasibility of our approach depends on the execution time for the entire process. Total measurement time (t_{tot}) consists of the sum of the times for a full stimulus adaptation (t_a) and then shape reconstruction (t_r):

$$t_{tot} = t_a + t_r \quad (7)$$

Additionally, the measurement process involves two types of refinement: we refine the stimulus pattern via more quadrilaterals and the B-spline model by incorporating more patches.

Stimulus adaptation

To evaluate our stimulus adaptation approach, we built the system that Figure 9 shows schematically.

Instrumentation. For stimulus generation, we use the LCD screen of a Sony XV-M30E portable video camera. This device's resolution is 382×240 pixels at dimensions of 62×46 mm. A 350-MHz Pentium II system generates the input stimulus, which the LCD screen displays via the monitor signal. To accommodate the difference in resolution of the screen and the LCD we use a VHX 470 Scan Vision scan line converter that converts computer signals into video. A Panasonic GP US502 camera registers the reflection,

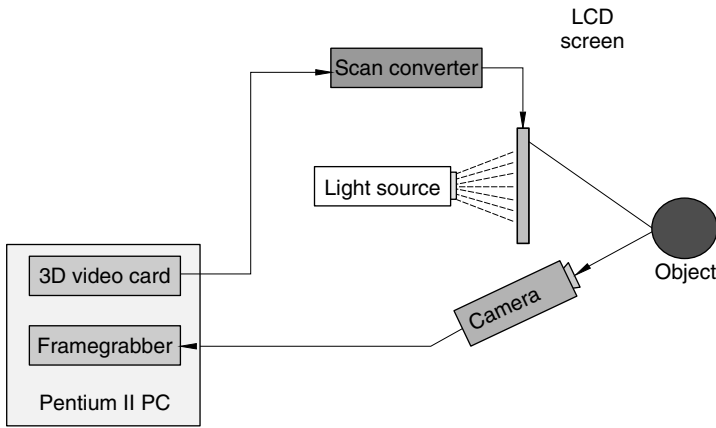


Figure 9. Experimental system configuration for stimulus adaptation, schematic drawing.

and a Matrox Meteor frame-grabber digitizes the recorded image. All image processing takes place on the Pentium II configuration.

Time complexity. Execution time for stimulus adaptation (t_a , see Equation 7) depends on the LCD screen's update frequency (t_{LCD}), the CCD's integration time CCD (t_{CCD}), and the image-processing time. Because both the LCD's update frequency and the CCD's integration time are fixed at 40 ms per image, the implementation's effectiveness is visible only in the image-processing execution time. Particularly, the implementation's performance depends on the level of refinement (which determines the total number of vertices), the time to locate a vertex, and the number of iterations to arrive at a regular pattern (see Figure 5).

Table 1. Time to localize crossings at various refinement levels.

| Refinement level (n) | $t_c(n)$ (milliseconds) |
|--------------------------|-------------------------|
| 1 | 162 |
| 2 | 111 |
| 3 | 78 |
| 4 | 2 |

Table 2. Number of iterations for adaptation of the stimulus to a test object.

| Object | $R(1)$ | $R(2)$ | $R(3)$ | $R(4)$ |
|-----------|--------|--------|--------|--------|
| Flat | 3 | 1 | 1 | 1 |
| Ellipsoid | 6 | 3 | 2 | 1 |
| Sinusoid | 1 | 3 | 2 | 1 |

A good approximation of the time to do the full stimulus adaptation (t_a) is

$$t_a = \sum_{n=1}^l (t_c(n) + t_{LCD} + t_{CCD})R(n), \quad (8)$$

where l is the total number of refinement levels, $t_c(n)$ is the time to locate vertices at refinement level n , and $R(n)$ expresses the number of iterations it takes to regularize the pattern at level n .

Here, we will determine $t_c(n)$ and explore the behavior of $R(n)$ to discover the characteristics of t_a . We monitored these numbers while measuring three test objects: a flat surface, an ellipsoid, and a sinusoidally shaped surface (an object with a small protrusion simulating a deformity). We adopted four levels of refinement to arrive at 256 quadrilaterals.

Table 1 shows time $t_c(n)$ for locating vertices at each refinement level. Although the number of crossings increases exponentially with each level, this effect is more than balanced by the shrinking search spaces and smaller matched filters. Consequently, the execution time falls back dramatically at higher levels of refinement.

Table 2 shows the number of iterations $R(n)$ for each level of refinement before arriving at a regularized reflection. At the first level (four quadrilaterals), the shape adaptation is the most cumbersome. Subsequently, position prediction is already so reliable that only a few iterations suffice. After level 3, it takes just one iteration to achieve regularization.

Clearly, regularization requires more steps as the object shape exceedingly deviates from linearity (going from flat to sinusoidal surface shape). This is to be expected, because the algorithm assumes a linear transformation from stimulus to reflection upon shape adaptation (Figure 6).

From Equation 5, using $t_{LCD} = t_{CCD} = 0.04$ seconds and the data from Table 1 and Table 2, we find total execution times of 1.2 seconds for the flat surface, 2.4 seconds for the ellipsoidal surface, and 3.4 seconds for the sinusoidal surface.

Shape reconstruction

Nonlinear shape reconstruction has proved too slow for surgical applications, so the effectiveness of parallelizing this step crucially affects our instrument's overall feasibility.

Implementation. In a sequential implementation, calculating the Jacobian matrix consumes the largest part of the execution time (about 65 per-

cent, see Figure 7). It requires $(n \times m)$ independent evaluations of the residual function (with n representing the number of data points and m the number of spline parameters). Therefore, parallelization is an obvious way to speed up this part of the algorithm. Clearly, to get a speedup exceeding a factor of 3 (about $100/(100 - 65)$, by Amdahl's law) for the full program, we must parallelize the other parts as well.

Parallel calculation of the Jacobian matrix merely involves parceling out the individual calculations to the available processors. We can apply the same strategy to the assessment of the current residual vector, which involves n independent calculations, as described earlier. For a technique to parallelize the QR decomposition of the Jacobian matrix we refer to work by Gene H. Golub and James M. Ortega.⁹

We implemented the parallel algorithm in Orca,¹⁰ a language for parallel programming on distributed-memory systems, and we tested the program on a cluster computer running the Linux operating system. Each node contains a 200-MHz Pentium Pro processor with 128 Mbytes of internal memory. The processors are connected by a 1.2-gigabits/second Myrinet network.

Time complexity. To explore the parallelized reconstruction algorithm's time complexity let's first look into our implementation's efficiency to identify the limiting factors.

Figure 10 shows SU gained for each program part when we use increasingly more processors. We obtained these results by modeling the sinusoidal test object through a B-spline with 6×6 patches using 3,747 data points (see Figure 7 for the sequential result).

From the linear relation between number of processors and $SU_{Jacobian}$, it's clear that the Jaco-

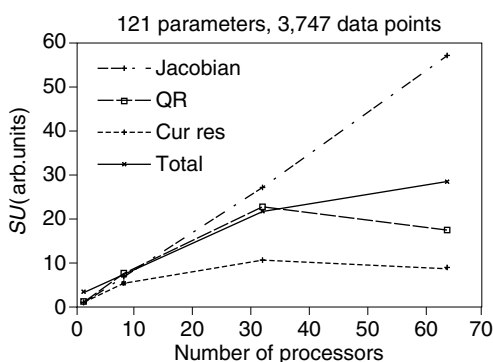


Figure 10. Speedup (SU) of the entire reconstruction's main segments as a function of the number of processors.

bian matrix calculation is implemented efficiently ($\text{Max}(SU_{Jacobian}) = 60$). However, SU_{QR} reaches a maximum of about 20, and SU_{cur_res} reaches a maximum of about 10. These program segments restrict the total speedup per iteration (SU_{iter}).

Figure 11 depicts the net result per iteration (SU_{iter}) for various numbers of data points and parameters. Obviously, the speedup improves as the number of data points increases (Figure 11a). In contrast, the difference when using more parameters for the B-spline seems insignificant (Figure 11b). From these curves' profiles, we concluded that speedup is at its maximum with 64 processors. The limiting factors are the QR-decomposition and residual vector calculations.

Figure 12 gives the time per iteration (t_{iter}) with the program running on 64 processors to obtain maximal speedup.

The complete picture

Now, let's explore the consequences of the results we've presented for the total measurement time (t_{tot}). Consider the two cases of the ellipsoidal and the sinusoidal surfaces, representing a normal and a pathological eye. The full measurement time for each object is given by the

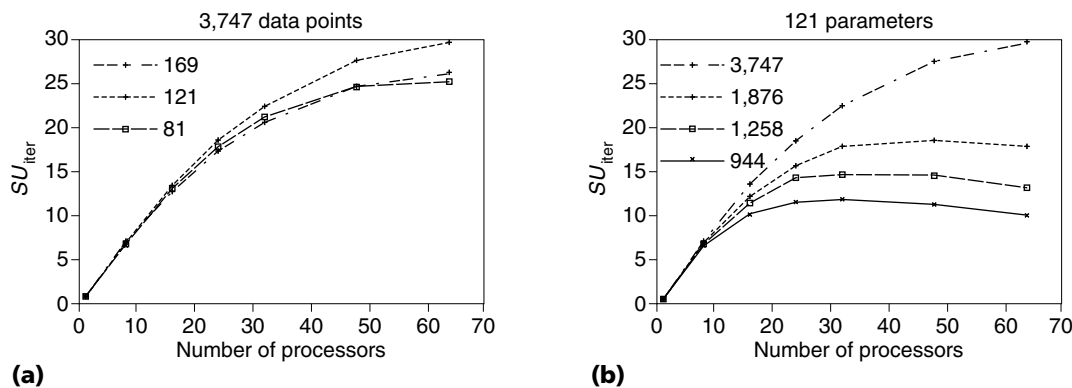


Figure 11. Speedup per iteration (Su_{iter}) as a function of the number of processors for (a) the surface model's complexity and (b) the input data set's various sizes .

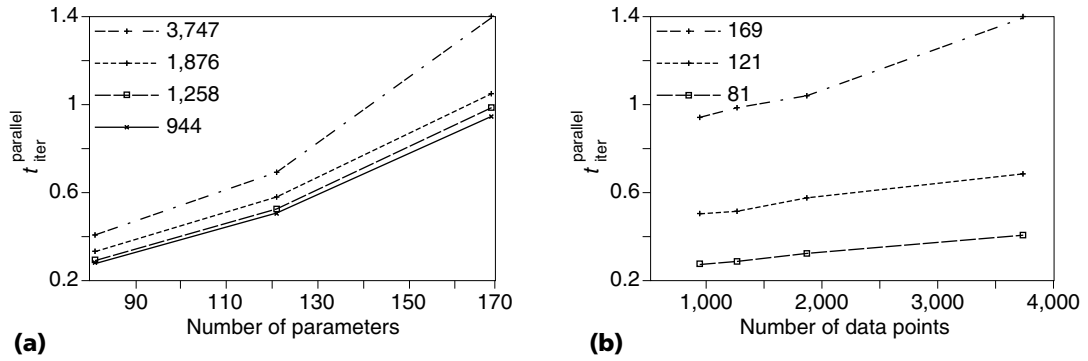


Figure 12. Time per iteration on a parallel computer (a) as a function of the number of parameters for variably sized input data sets and (b) as a function of the number of data points for varying numbers of parameters.

sum of the times to do stimulus adaptation and shape reconstruction (see Equation 7).

The stimulus adaptation took 2.4 seconds for the ellipsoidal object and 3.4 seconds for the sinusoidal object (measured on a 350-MHz Pentium II system). The time per iteration for shape reconstruction strongly depends on the numbers of data points and parameters (see Figure 12, tested on the 200-MHz Pentium Pro cluster configuration). We can obtain a good approximation of the ellipsoid with a B-spline surface of 4×4 patches (81 parameters). The sinusoid requires 8×8 patches. Multiplying the time per iteration by their total number gives the approximate shape modeling execution time (for completeness, we must also add initialization time).

Table 3 summarizes total measurement time t_{tot} for the test objects with a varied number of data points.

It might come as a surprise that the total execution time does not grow with larger numbers of data points. This is because of the more efficient implementation (greater speedup) with increasing data size, as Figure 12 illustrates. Other small variations occur because of fluctuations in the number of iterations.

For the ellipsoid, our system well approximates our goal, achieving response times on the order of a few seconds. In contrast, the mea-

surement time of the sinusoidal object deviates by a factor of 3 (see Table 3). As the times for stimulus adaptation are comparable—2.4 versus 3.4 seconds—it is mainly the nonlinear reconstruction that needs further improvement.

Alternatively, we might distinguish an initial starting period from the steady-state situation. The numbers we've reported so far apply to situations involving a significant learning period. As soon as an accurate model of the cornea can be established, we might obtain a much quicker response with small eye movements. In that case, the stimulus adaptation should require fewer iterations to regularize. From Equation 8 and Table 1, we can deduce that the surplus execution time is on the order of 0.1 seconds. Also, we could facilitate the shape reconstruction by starting with a good approximation of the measured shape. From Figure 12, we conclude that the extra modeling time incurred by *not* starting with an approximation is on the order of a few seconds.

Although we determined that speedup for our shape reconstruction process is at its maximum with 64 processors, a low-cost, 64-node configuration is not yet commercially available. However, computers containing four to eight processors are beginning to appear on the market. To give an impression of the speedup that we can expect in the near future, we ran our program

Table 3. Total test object measurement time using various numbers of data points.


| Number of data points | Ellipsoid | | Sinusoid | |
|-----------------------|---------------------|----------------------|---------------------|----------------------|
| | t_{tot} (seconds) | Number of iterations | t_{tot} (seconds) | Number of iterations |
| 944 | 5.8 | 12 | 17.1 | 12 |
| 1,258 | 6.1 | 12 | 17.2 | 12 |
| 1,876 | 6.0 | 10 | 15.8 | 9 |
| 3,747 | 7.2 | 10 | 20.3 | 10 |

on a 600-MHz Athlon system. As a preliminary test, we had the system model a sinusoidal object using 3,747 data points and 121 parameters (as in Figure 7). Whereas the reconstruction on the Pentium processor took 322 seconds in total, the Athlon configuration executed it in 116.7 seconds. Apparently, the speedup is clock-bound: $600 \text{ MHz}/200 \text{ MHz} \approx 322/116.7$. Thus—disregarding communication time for simplicity—just using a more up-to-date processor could bring the response times into a reasonable realm for practical use.

With normal corneas, the full program for the instrument we propose will take between 5.2 and 7.8 seconds to execute—a reasonable time for surgical application. However, in the most complicated situations the system cannot meet the goal of executing within a few seconds. Nevertheless, the difference amounts to no more than a factor of 3. We expect to be able to bridge that last, small gap in very near future.

Our discussion so far has disregarded our instrument's accuracy. One of us (Frans M. Vos) described an error analysis of a very similar instrument elsewhere.² In essence, all our errors but one are identical to this earlier description. The previous experiments using test objects showed an accuracy on the order of $1 \mu\text{m}$.² A new source of error that we have not investigated, however, is the discretized stimulus pattern (via the LCD screen). In practice, only at a high resolution (64×64 quadrilaterals, a quarter of the LCD's resolution), the reflection sometimes shows irregularities when measuring deformed objects. We expect that we can solve this problem by using state-of-the-art LCD screens that currently come with a much higher resolution.

Our experimental use of test objects was somewhat simpler than real cornea measurements because we had no occluding effect of eye or nose shadows. Consequently, we could adopt a straightforward strategy in which the program would cease shape adaptation as soon as it could not detect a vertex in the reflection.

Our current work to miniaturize the instrument to enable practical measurements also involves implementing more sophisticated stimulus adaptation by local refinement. Although this article serves as a proof of principle, we hope that our miniaturization work will prove that our instrument can measure real corneas as well. 

Acknowledgments

Several undergraduate students made significant contributions to this work. In particular, we thank Hamid Taikandi for the implementation in Orca. We also gratefully acknowledge Rob van der Heijde and Frans Groen for their contributions to discussions on the subject.

References

1. F.M. Vos et al., "A New PRBA-Based Instrument to Measure the Shape of the Cornea," *IEEE Trans. Instrumentation and Measurement*, vol. 46, no. 4, Aug. 1992, pp. 794–797.
2. F.M. Vos, *A System for Measuring, Modeling, and Reconstructing Corneal Shapes based on Pseudo Random Encoding*, doctoral dissertation, Physics Department, Vrije Universiteit, Amsterdam, 1998.
3. S.A. Klein, "A Corneal Topography Algorithm that Produces Continuous Curvature," *Optometry and Vision Science*, vol. 69, no. 11, Nov. 1992, pp. 829–834.
4. M.A. Halstead et al., "A Spline Surface Algorithm for Reconstruction of Corneal Topography from a Videokeratographic Reflection Pattern," *Optometry and Vision Science*, vol. 72, no. 11, Nov. 1995, pp. 821–827.
5. S.E. Wilson, S.D. Klyce, and Z.M. Hussein, "Standardized Color-Coded Maps for Corneal Topography," *Ophthalmology*, vol. 100, no. 11, Nov. 1993, pp. 1723–1727.
6. J.J. Antalis, R.G. Lembach, and L.G. Carney, "A Comparison of the TMS-1 and the Corneal Analysis System for the Evaluation of Abnormal Corneas," *Contact Lens Assoc. Ophthalmologists J.*, vol. 19, no. 1, Jan. 1993, pp. 58–63.
7. M.W. Belin and C.D. Ratliff, "Evaluation Data Acquisition and Smoothing Functions of Currently Available Videokeratoscopes," *J. Cataract and Refractive Surgery*, vol. 22, no. 4, May 1996, pp. 421–426.
8. H.J.W. Spoelder et al., "A Study of the Robustness of Pseudo Random Binary Array Based Surface Characterization," *IEEE Trans. Instrumentation and Measurement*, vol. 47, no. 4, Aug. 1998, pp. 833–838.
9. G.H. Golub and J.M. Ortega, *Scientific Computing: An Introduction with Parallel Computing*, Academic Press, Boston, pp. 259–264.
10. H.E. Bal, M.F. Kaashoek, and A.S. Tanenbaum, "Orca: A Language for Parallel Programming of Distributed Systems," *IEEE Trans. Software Eng.*, vol. 18, no. 3, Mar. 1992, pp. 190–205.

Frans M. Vos is a postdoctoral research fellow with the Pattern Recognition Group at Delft University of Technology. He is also a staff member in the Department of Radiology at the Academic Medical Center, Amsterdam. His research is chiefly in medical image processing and visualization. He received his master's degree in medical informatics and computer science at the University of Amsterdam and his PhD at the Vrije University, Amsterdam. Contact him at Pattern Recognition Group, Faculty of Applied Physics, Delft Univ. of Technology, Lorentzweg 1, 2628 CJ Delft, Netherlands; frans@ph.tn.tudelft.nl.

Hans J.W. Spoelder is an associate professor in the Physics Applied Computer Science Group, Division of Physics and Astronomy, Vrije Universiteit, Amsterdam.

He has also worked as a visiting scientist at IBM's T.J. Watson research Center, Hawthorne, N.Y. His research interests include modeling, virtual environments, virtual instrumentation, and man-machine interaction. He has a degree in experimental physics and a PhD in biophysics from the Free University of Amsterdam, Netherlands. He is a senior member of the IEEE. Contact him at Division of Physics and Astronomy, Faculty of Sciences, Vrije Universiteit, De Boelelaan 1081, 1081 HV Amsterdam, Netherlands.

Desmond M. Germans is a PhD student in the Physics Applied Computer Science Group at the Vrije University of Amsterdam, where he also earned an MSc in physics. His research interests include high-performance graphics and interactive applications in virtual reality. Contact him at Division of Physics and Astronomy, Faculty of Sciences, Vrije Universiteit, De Boelelaan 1081, 1081 HV Amsterdam, Netherlands.

Rutger Hofman is a research programmer in Henri Bal's Computer Systems Group. He holds a degree in computer science from the University of Amsterdam. His work focuses on parallel performance, especially in

network layers and parallel applications. Contact him at Division of Mathematics and Computer Science, Faculty of Sciences, Vrije Universiteit, De Boelelaan 1081A, 1081 HV Amsterdam, Netherlands.

Henri Bal is a professor in both the Department of Computer Systems and the Department of Physics-Applied Computer Science at the Vrije University. His research concerns parallel programming environments and their applications and cluster and grid computing. He designed the Orca language and leads the Manta, Albatross, and other projects. He is also program chairman of the 2002 IEEE International Symposium on Cluster Computing and the Grid (ccGrid'02). Bal holds an MSc in mathematics from the Delft University of Technology and a PhD in computer science from the Vrije Universiteit, Amsterdam. Contact him at Division of Mathematics and Computer Science, Faculty of Sciences, Vrije Universiteit, De Boelelaan 1081A, 1081 HV Amsterdam, Netherlands.

For more information on this or any other computing topic, please visit our Digital Library at <http://computer.org/publications/dlib>.

PURPOSE The IEEE Computer Society is the world's largest association of computing professionals, and is the leading provider of technical information in the field.

MEMBERSHIP Members receive the monthly magazine **COMPUTER**, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others interested in the computer field.

BOARD OF GOVERNORS

Term Expiring 2002: Mark Grant, Gene F. Hoffnagle, Karl Reed, Kathleen M. Swigger, Ronald Waxman, Michael R. Williams, Akihiko Yamada

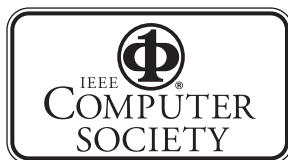
Term Expiring 2003: Fiorenza C. Albert-Howard, Manfred Brody, Alan Clements, Richard A. Kemmerer, Susan A. Mengel, James W. Moore, Christina M. Schober

Term Expiring 2004: Jean M. Bacon, Ricardo Baeza-Yates, Deborah M. Cooper, George V. Cybenko, Wolfgang K. Giloi, Haruhisa Ichikawa, Thomas W. Williams

Next Board Meeting: 10 May 02, Portland, OR

IEEE OFFICERS

President: **RAYMOND D. FINDLAY**
 President-Elect: **MICHAEL S. ADLER**
 Past President: **JOEL B. SYNDER**
 Executive Director: **DANIEL J. SENESE**
 Secretary: **HUGO M. FERNANDEZ VERSTAGEN**
 Treasurer: **DALE C. CASTON**
 VP, Educational Activities: **LYLE D. FEISEL**
 VP, Publications Activities: **JAMES M. TIEN**
 VP, Regional Activities: **W. CLEON ANDERSON**
 VP, Standards Association: **BEN C. JOHNSON**
 VP, Technical Activities: **MICHAEL R. LIGHTNER**
 President, IEEE-USA: **LeEARL A. BRYANT**



EXECUTIVE COMMITTEE

President: WILLIS K. KING*
University of Houston
Dept. of Comp. Science
 501 PGH
 Houston, TX 77204-3010
 Phone: +1 713 743 3349 Fax: +1 713 743 3335
 w.king@computer.org

President-Elect: STEPHEN L. DIAMOND*
Past President: BENJAMIN W. WAH*
VP, Educational Activities: CARL K. CHANG*
VP, Conferences and Tutorials: GERALD L. ENGEL*
VP, Chapters Activities: JAMES H. CROSS*
VP, Publications: RANGACHAR KASTURI*
VP, Standards Activities: LOWELL G. JOHNSON (2ND VP)*
VP, Technical Activities: DEBORAH K. SCHERRER(1ST VP)*
Secretary: DEBORAH M. COOPER*
Treasurer: WOLFGANG K. GILOI*
2001-2002 IEEE Division VIII Director: THOMAS W. WILLIAMS
2002-2003 IEEE Division V Director: GUYLAINE M. POLLOCK*
Executive Director: DAVID W. HENNAGE*

*voting member of the Board of Governors



COMPUTER SOCIETY WEB SITE

The IEEE Computer Society's Web site, at <http://computer.org>, offers information and samples from the society's publications and conferences, as well as a broad range of information about technical committees, standards, student activities, and more.

COMPUTER SOCIETY OFFICES

Headquarters Office

1730 Massachusetts Ave. NW
 Washington, DC 20036-1992
 Phone: +1 202 371 0101 • Fax: +1 202 728 9614
 E-mail: hq.ofc@computer.org

Publications Office

10662 Los Vaqueros Cir., PO Box 3014
 Los Alamitos, CA 90720-1314
 Phone: +1 714 821 8380
 E-mail: help@computer.org
 Membership and Publication Orders:
 Phone: +1 800 272 6657 Fax: +1 714 821 4641
 E-mail: help@computer.org

European Office

13, Ave. de L'Aquilon
 B-1200 Brussels, Belgium
 Phone: +32 2 770 21 98 • Fax: +32 2 770 85 05
 E-mail: euro.ofc@computer.org

Asia/Pacific Office

Watanabe Building
 1-4-2 Minami-Aoyama, Minato-ku,
 Tokyo 107-0062, Japan
 Phone: +81 3 3408 3118 • Fax: +81 3 3408 3553
 E-mail: tokyo.ofc@computer.org

EXECUTIVE STAFF

Executive Director: **DAVID W. HENNAGE**
 Publisher: **ANGELA BURGESS**
 Assistant Publisher: **DICK PRICE**
 Director, Volunteer Services: **ANNE MARIE KELLY**
 Chief Financial Officer: **VIOLET S. DOAN**
 Director, Information Technology & Services: **ROBERT CARE**
 Manager, Research & Planning: **JOHN C. KEATON**