Image Fusion and Subpixel Parameter Estimation for Automated Optical Inspection of Electronic Components

James M. Reed, *Student Member, IEEE,* and Seth Hutchinson, *Member, IEEE*

***Abstract*— We present a new approach to automated optical inspection (AOI) of circular features that combines image fusion with subpixel edge detection and parameter estimation. In our method, several digital images are taken of each part as it moves past a camera, creating an image sequence. These images are fused to produce a high-resolution image of the features to be inspected. Subpixel edge detection is performed on the high-resolution image, producing a set of data points that is used for ellipse parameter estimation. The fitted ellipses are then back-projected into 3-space in order to obtain the sizes of the circular features being inspected, assuming that the depth is known. The method is is accurate, efficient, and easily implemented. We present experimental results for real intensity images of circular features of varying sizes. Our results demonstrate that our algorithm shows greatest improvement over traditional methods in cases where the feature size is small relative to the resolution of the imaging device.**

I. Introduction

PARTS WITH circular features, such as holes, are common in the microelectronics industry. For example, holes used for mounting integrated circuits and other electronic components are found on printed circuit boards. If the holes in these circuit boards are not located or shaped correctly, electronic components may not fit into them correctly. Due to the small size of many electronic components, holes must be manufactured precisely; therefore, inspecting the circular shape of these holes requires a high degree of accuracy.

The research that we report in this paper is intended for use in automated optical inspection of via holes in printed circuit boards. Via holes are used to provide electrical connections between different sides or layers of a printed circuit board. In such an inspection system, circuit boards move along a conveyer at a fixed velocity. Several images are taken of each part as it passes beneath the camera, creating an image sequence $\mathcal{Q} = \{q_{(1)} \dots q_{(n)}\}$. The vias are inspected one-at-a-time, and the entire shape of each hole being inspected is visible in every image in $\mathcal{Q}$. The images in the image sequence are perspective projections of the scene; therefore, the via holes appear as ellipses in these images [1]. Given the image sequence, $\mathcal{Q}$, our task is to estimate the parameters of the elliptical shape of the via holes with subpixel accuracy. From these estimates, we can infer the properties of the shape of the actual via holes, and use this information to decide whether a via hole is properly shaped.

Our method combines image fusion (inlcuding image registration and image enhancement) with subpixel edge detection, and subpixel parameter estimation of ellipses to perform the inspection task described above. In Sections II and III we describe how, given the input image sequence $\mathcal{Q}$, we perform image enhancement using Peleg and Irani's superresolution algorithm [2]. The superresolution algorithm creates a high-resolution image $\mathcal{H}$ that has twice the resolution of the individual images in $\mathcal{Q}$. In Section IV, we describe how subpixel arc-edge detection is performed on the high-resolution estimate $\mathcal{H}$, yielding a list of data points. The arc-edge detector is a sample-moment-based edge detector that locates data points that lie on a circular arc with subpixel accuracy [3]. Then, in Section V, we describe how these data points are used by an ellipse parameter estimation algorithm [4]. Among the benefits of our system are increased noise tolerance and reduced hardware requirements. Because image sequence analysis is used for image enhancement, high-resolution cameras and high-precision positioning equipment are not needed. Our research was conducted using off-the-shelf hardware and tested on real images.

II. Image Registration

Given a sequence of images, $\mathcal{Q} = \{q_{(1)} \dots q_{(n)}\}$, our first task is to bring these images into registration, producing a new image sequence, $\mathcal{Q}_a$, in which all of the images are aligned. In the image sequence $\mathcal{Q}_a$, the pixels corresponding to a circular feature being inspected occupy the same locations in each image. The registration process that we describe in this section was introduced by Irani and Peleg in [2].

Image registration is accomplished by estimating the motion vector for each image, then shifting each image according to its motion vector. The motion vectors are estimated in an iterative process and expressed with respect to a reference image, $q_{(r)}$, chosen from $\mathcal{Q}$. For a particular image $q_{(c)}$, let $T = (t_x + \rho_x, t_y + \rho_y)$ represent an initial estimate of the motion between images $q_{(r)}$ and $q_{(c)}$, where (t_x, t_y) is the integer part of the motion and (ρ_x, ρ_y) is the fractional part. Motion vector estimation is performed by repeating the following steps. First, we shift the image according to its

Manuscript received February 8, 1995; revised November 30, 1995.

J. M. Reed is with the Artificial Intelligence Laboratory, The University of Michigan, Ann Arbor, MI 48109-2110 USA.

S. Hutchinson is with the Department of Electrical and Computer Engineering, The Beckman Institute for Advanced Science and Technology, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA.

Publisher Item Identifier S 0278-0046(96)02367-2.

motion estimate. Second, we compute a correction to the motion estimate by solving a system of linear equations given by (3). When the changes to the motion estimate are less than a threshold value, motion estimation stops.

Because each motion vector has both an integer and a subpixel part, we shift an image in two steps. In the first part of the shifting operation, $q_{(c)}$ is shifted according to the integer part of the motion. No intensity values are changed during this integer shift; therefore, the intensity of each pixel after shifting by (t_x, t_y) can be expressed as

$$I'(x + t_x, y + t_y) = I(x, y), \tag{1}$$

where $I(x, y)$ is the intensity of the pixel at position (x, y), and $I'(x + t_x, y + t_y)$ is the intensity of the pixel at position $(x + t_x, y + t_y)$ in the shifted image.

In the second part of the shifting process, the subpixel part of the motion is used to estimate new intensities for the shifted pixels. This is accomplished by approximating the intensity function, $I(x, y)$, by a plane, and using bilinear interpolation to compute the intensity values at subpixel locations. To avoid the accumulation of errors, the motion estimation algorithm always shifts the original image $q_{(c)}$.

In the second part of the motion estimation algorithm, a correction to the current motion estimate is calculated by solving a set of linear equations derived by Keren *et al.* in [5]. In general, motion in two dimensions can be described by a translation (a, b) and a rotation, θ, assumed to be about an axis at the center of the image. In terms of the motion vector T, the components of the motion are $a = t_x + \rho_x$ and $b = t_y + \rho_y$. In terms of the parameters a, b, θ, the relationship between $q_{(r)}$ and $q_{(c)}$ is given by

$$q_{(c)}(x, y) = q_{(r)}(x \cos\theta - y \sin\theta + a, y \cos\theta + x \sin\theta + b). \tag{2}$$

As shown in [5], an approximation to the sum-of-squares error between $q_{(r)}$ and $q_{(c)}$ can be derived by linearization of (2). Setting partial derivatives of this approximate error function to zero, we obtain the system

$$\left.\begin{array}{c} \sum D_x^2 a + \sum D_x D_y b + \sum A D_x \theta = \sum D_x D_t \\ \sum D_x D_y a + \sum D_y^2 b + \sum A D_y \theta = \sum D_y D_t \\ \sum A D_x a + \sum A D_y b + \sum A^2 \theta = \sum A D_t \end{array}\right\} \tag{3}$$

where

$$D_x = \frac{\partial q_{(r)}(x, y)}{\partial x}, \qquad D_y = \frac{\partial q_{(r)}(x, y)}{\partial y},$$
$$D_t = q_{(c)}(x, y) - q_{(r)}(x, y), \qquad A = xD_y - yD_x.$$

Solving (3) for a, b, and θ yields a vector that is used to update the current motion estimate at the end of each iteration, until the magnitude of the correction vector is below a predefined threshold. The motion estimation algorithm produces an aligned sequence of images, $\mathcal{Q}_a$, and list of motion vectors, $\mathcal{L}$.

III. Creating the High-Resolution Image

We use the superresolution method of Irani and Peleg [2] to create a high-resolution image using the aligned image sequence, $\mathcal{Q}_a$, and the list of motion vectors, $\mathcal{L}$. The result is a fused image that has twice the resolution of the input images. An alternative approach is given in [6], which describes a system that uses subpixel camera displacements to create the high resolution image.

The superresolution algorithm works by creating an initial estimate of the high-resolution image, $\mathcal{H}$, and then using an iterative refinement procedure to improve that estimate by exploiting known characteristics of the imaging process. The iterative refinement proceeds as follows. A sequence of low resolution images $\mathcal{S} = \{s_{(1)} \cdots s_{(n)}\}$ is created by subsampling $\mathcal{H}$, and then shifting each subsampled image according to the corresponding motion vector in $\mathcal{L}$. If the high-resolution estimate is correct, then the actual and simulated image sequences will be identical, i.e., $\mathcal{S} = \mathcal{Q}$. If $\mathcal{S} \neq \mathcal{Q}$ the difference images between $\mathcal{Q}$ and $\mathcal{S}$ are calculated, creating a sequence of difference images $\mathcal{Q} - \mathcal{S} = \{(q_{(1)} - s_{(1)}) \cdots (q_{(n)} - s_{(n)})\}$. Corrections to the high-resolution estimate are based on the values of these difference images, as described below in Section III-B.

A. Computing Simulated Low Resolution Images

Let $q(x, y)$ represent a low resolution image pixel. Pixel $q(x, y)$ corresponds to a photosite in the CCD camera [1]. The intensity at $q(x, y)$ is determined by the receptive field of the photosite, which is sensitive to light emanating from a scene region that is defined by the center, size, and shape of the photosite's receptive field [7]. The receptive fields of adjacent photosites overlap due to the proximity and spacing of the sites; therefore, light emanating from any given point in the scene influences the intensity of several pixels in the low resolution image. The point spread function of the sensor describes which low resolution pixels are influenced by a given point in the scene. Because CCD cameras use an array of photosites, we assume that the receptive fields and point spread functions of the photosites are identical.

The camera produces a discretized, low resolution version of the original scene. The imaging model is a mathematical representation of the imaging process. We use the model presented by Irani and Peleg in [8]

$$q_{(k)}(x, y) = \sigma_{(k)}(h(f(i, j)) + \eta_{(k)}(i, j)) \tag{4}$$

where $q_{(k)}$ is the kth image of the image stream $\mathcal{Q}$. Equation (4) represents a transformation from the scene f to a digital image $q_{(k)}$. The blurring operator, h, is defined by the point spread function of the sensor. Because we do not actually know the sensor's properties, we assume that the point spread function is a Gaussian smoothing operator. Additive noise is represented by $\eta_{(k)}$. The function $\sigma_{(k)}$ digitizes the image into pixels and quantizes the resulting pixel intensities into discrete gray levels.

The imaging process model (4) describes how low-resolution input images are produced. To create simulated

low-resolution images, we approximate (4) by

$$s^n(x,y) = \sum_{\beta} \mathcal{H}^n(i,j) h^{\mathrm{PSF}}(i - z_x, j - z_y). \quad (5)$$

in which s^n is a low-resolution image, and $\mathcal{H}^n$ is the high-resolution image produced in the nth iteration of the refinement algorithm. Each simulated low resolution pixel $s^n(x,y)$ is the weighted average of the high-resolution pixels, $\mathcal{H}^n(i,j)$, that are in the low-resolution pixel's receptive field; the set of these high-resolution pixels is denoted by β. The point spread function of the imaging system, h in (4), is represented by a mask that is denoted by h^{PSF} in (5). The image coordinates of the center of this mask, (z_x, z_y), are used to select the mask value for a given high-resolution pixel (i,j).

In (4) a blurred version of the continuous image f is discretized to produce input low-resolution images, while in (5) a discretized high-resolution image $\mathcal{H}^n$ is blurred to produce simulated low-resolution images.

Each simulated low-resolution image $s^n_{(i)}$ is created by shifting s^n according to the motion estimate that corresponds to the image $q_{(i)}$ in $\mathcal{Q}$. This shifting operation produces a simulated image that is in congruence with $q_{(i)}$. Simulated images are created for each image in the input image stream.

B. Iterative Refinement

After creating simulated images, improvements to the high-resolution estimate are calculated using the updating formula

$$\mathcal{H}^{n+1}(i,j) = \mathcal{H}^n(i,j) + \left\{ \sum_{k,\alpha} \left(q_{(k)}(x,y) - s^n_{(k)}(x,y) \right) \times \frac{(h^{\mathrm{BP}}(x - z'_x, y - z'_y))^2}{c \sum_{\alpha} h^{\mathrm{BP}}(x - z'_x, y - z'_y)} \right\} \quad (6)$$

where $q_{(k)}$ is the kth image of the input image stream $\mathcal{Q}$, and $s^n_{(k)}$ is the kth simulated low-resolution image. The function h^{BP} represents a back-projection kernel that is used to calculate improvement values for the high resolution estimate. The contribution of the back-projection kernel is normalized using a constant, c. The choice of h^{BP} is discussed in [2]. We use a normalized, 5×5 mask representing a Gaussian operator in our work. The proper back-projection kernel value is chosen using the low-resolution image coordinates of the center of the back-projection kernel, (z_x, z_y). The improvement value for a given high-resolution pixel is the weighted average of the contributions of the low-resolution pixels in its receptive field; the set of all such pixels is denoted by α. Given the high-resolution estimate in the nth iteration of the refinement algorithm $\mathcal{H}^n$, the refinement algorithm creates a new high-resolution estimate, $\mathcal{H}^{n+1}$, by calculating improvement values for the pixels of $\mathcal{H}^n$.

The initial high-resolution estimate $\mathcal{H}$ is created by averaging the values of pixels in the aligned image sequence $\mathcal{Q}_a = \{q_{a(1)} \cdots q_{a(n)}\}$

$$\mathcal{H}(2x,2y) = \mathcal{H}(2x,2y+1) = \mathcal{H}(2x+1,2y) = \mathcal{H}(2x+1,2y+1) = \frac{1}{n} \sum_{q_a \in \mathcal{Q}_a} q_a(x,y). \quad (7)$$

IV. Subpixel Edge Detection

There are many methods of edge detection (see, e.g., [9]). Standard edge operators are easy to implement; however, in their simplest forms they are pixel-level edge detectors, which can only localize edges to the nearest pixel. Although efforts have been made to increase the accuracy of these methods, they cannot be used for subpixel edge detection unless some form of interpolation is used [10]. The limited resolution of early edge detectors led to the development of subpixel edge detection algorithms, which localize edges to within a fraction of a pixel precision. A number of subpixel edge detectors rely on some form of interpolation, e.g., [11]–[13]. Others rely on moment analysis, e.g., [3], [8], [10], [14], [15].

We use the arc-edge detector described in [3]. In the first step, we apply bilevel thresholding and simple edge detection to the high-resolution image $\mathcal{H}$ to create an edge map. The following operations are done on $\mathcal{H}$ at each location specified in the edge map. First, we approximate the circular arc with straight-line segments. The parameters of these line segments are calculated to subpixel accuracy using Tabatabai and Mitchell's moment-based straight-line-edge detector [10]. Given the straight-line approximation of the circular shape, we calculate the coordinates of the points that lie on the circular border curve. These data points are used in the ellipse parameter estimation described in Section V.

A. Building the Initial Edge Map

The arc-edge detector is applied to points of interest in the high resolution image $\mathcal{H}$. These points are indicated on an edge map that is created by performing bilevel thresholding on $\mathcal{H}$, then using simple edge detection to locate edge points in the binary image. We use Tsai's sample moment preserving bilevel thresholding algorithm to threshold $\mathcal{H}$ [16]. Given an input image, this algorithm locates the threshold value t such that the first four sample moments of the image histogram are preserved. The ith sample moment of the image data is

$$M_i = \frac{1}{n} \sum_x \sum_y (\mathcal{H}(x,y))^i \quad (8)$$

where $\mathcal{H}(x,y)$ is the pixel intensity at location (x,y) in $\mathcal{H}$, and n denotes the number of pixels in $\mathcal{H}$. By this definition, $M_0 = 1$. Sample moments can also be computed from the histogram of $\mathcal{H}$

$$M_i = \frac{1}{n} \sum_j n_j z_j^i = \sum_j p_j z_j^i \quad (9)$$

where n is the number of pixels in the image, n_j is the number of pixels in $\mathcal{H}$ with intensity value z_j, and $p_j = \frac{n_j}{n}$.

For bilevel thresholding, let z_b and z_f be the representative intensity values for the two regions of a binary image. The moment-preserving thresholding algorithm selects a threshold such that if the below-threshold pixels are replaced by z_b and the above-threshold pixels are replaced by z_f, then the first four sample moments will be preserved in the resulting bilevel image. Let u_b represent the fraction of below-threshold pixels in $\mathcal{H}$, and u_f represent the fraction of above-threshold pixels

in $\mathcal{H}$, with $u_b + u_f = 1$. The first four sample moments of the bilevel image are given by

$$M_i' = (z_b)^i u_b + (z_f)^i u_f, \quad i = 0, 1, 2, 3. \tag{10}$$

To preserve the first four moments in the bilevel image, $M_i' = M_i$ for $i = 0, 1, 2, 3$, leading to

$$\left.\begin{aligned} u_b + u_f &= 1 \\ z_b u_b + z_f u_f &= \textstyle\sum_j p_j z_j \\ (z_b)^2 u_b + (z_f)^2 u_f &= \textstyle\sum_j p_j z_j^2 \\ (z_b)^3 u_b + (z_f)^3 u_f &= \textstyle\sum_j p_j z_j^3 \end{aligned}\right\}. \tag{11}$$

The system (11) is solved for z_b, z_f, u_b, and u_f. The threshold value t is chosen by examining the histogram of the image. Beginning with the lowest intensity value in the histogram, we begin summing the number of pixels of each intensity value. The threshold value is chosen as the first intensity value that makes this sum of pixels greater than or equal to the fraction of below-threshold pixels, u_b. Given t, we perform thresholding on $\mathcal{H}$ to produce a binary image. Simple edge detection is performed on the binary image to produce the edge map.

B. Straight-Line-Edge Detection

Given the high-resolution estimate $\mathcal{H}$ and the edge map, sample moment preserving straight-line-edge detection is performed on $\mathcal{H}$ at locations specified by the edge map using Tabatabai and Mitchell's subpixel straight-line-edge detector [3], [10]. We summarize the procedure below.

The straight-line-edge detector locates lines that lie within a circular detection area. The detection circle consists of 69 pixels that are weighted to approximate a circle of radius 4.5 pixel units. Each pixel in the detection circle is weighted by the amount of the circle area it occupies. For each location (x, y) stored in the edge map, we center the detection area at location (x, y) and perform straight-line-edge detection in $\mathcal{H}$.

The parameters of the straight-line-edge model are shown in Fig. 1. Let r represent the radius of the detection circle. A straight-line-edge divides the detection circle into the two regions A_1 and A_2. We assume that the edge can be modeled as a step, with pixels of a given intensity on one side of the edge and pixels of a different intensity on the other. These intensities are the characteristic intensities of the regions that border the straight-line-edge.

The two regions that border the edge are described by a_i, the area of the region i; h_i, the characteristic intensity of region i; and p_i, the relative frequency of occurrence of pixels with intensity h_i within the detection circle. The straight-line-edge detector locates the edge by solving for the unknowns a_i, h_i, and p_i.

To facilitate this discussion, consider a line segment drawn from the center of the detection circle to the straight-line-edge, and normal to the straight-line-edge (see Fig. 1). The length of this normal, denoted by L, is the perpendicular distance from the center of the detection circle to the straight-line-edge. The angle of orientation of the straight-line-edge, α, is defined as the angle between the normal and the horizontal axis of the detection circle. In the following discussion, we will refer to the center of gravity, G, of A_2.

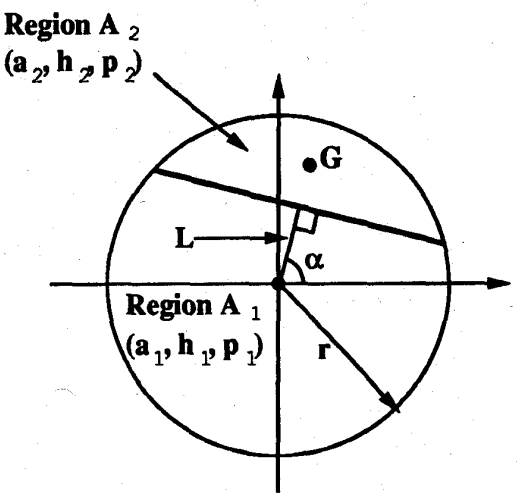

Fig. 1. The parameters of the edge model.

The straight-line-edge detector is also based on sample moments. Edges are located such that the first four sample moments of the image data are preserved. This is similar to the moment-preserving thresholding that was used in creating the edge map. In creating the edge map, the moments were calculated for the entire image $\mathcal{H}$. Here, the moments are calculated for the pixels within the detection circle. The first four sample moments are defined as

$$M_i = \sum_x \sum_y (\mathcal{H}(x, y))^i \omega(x - d_x, y - d_y), \quad i = 0, 1, 2, 3; \quad (x, y) \in \text{detection circle}. \tag{12}$$

The weight of pixel $\mathcal{H}(x, y)$ is denoted by $\omega(x - d_x, y - d_y)$, where (d_x, d_y) is the center of the detection circle. The summation (12) is taken over the pixels in the detection circle.

The equations that describe the sample moments used in edge detection are similar to the equations used for bilevel thresholding in Section IV-A. To preserve the first four sample moments, the following relations must be satisfied

$$\left.\begin{aligned} p_1 + p_2 &= M_0 \\ h_1 p_1 + h_2 p_2 &= M_1 \\ (h_1)^2 p_1 + (h_2)^2 p_2 &= M_2 \\ (h_1)^3 p_1 + (h_2)^3 p_2 &= M_3 \end{aligned}\right\} \tag{13}$$

where h_i, and p_i are described above.

Tabatabai and Mitchell present the following solutions for (13) in [10]. After computing the first four moments, the frequencies of occurrence p_1 and p_2 are calculated

$$p_1 = 1 - p_2 \tag{14}$$

$$p_2 = \frac{1}{2}\left[1 + \bar{s}\sqrt{\frac{1}{4 + \bar{s}}}\right] \tag{15}$$

where

$$\bar{s} = \frac{M_3 + 2M_1^3 + 2M_1 M_2}{\bar{\sigma}^3} \tag{16}$$

$$\bar{\sigma} = M_2 - M_1^2. \tag{17}$$

The characteristic intensities are calculated using p_1 and p_2

$$h_1 = M_1 - \bar{\sigma}\sqrt{\frac{p_2}{p_1}} \tag{18}$$

$$h_2 = M_1 + \bar{\sigma}\sqrt{\frac{p_1}{p_2}}. \tag{19}$$

After computing p_1, p_2, h_1, and h_2 for the pixels within the detection circle, we determine whether an edge is present by using a threshold on the difference between the characteristic intensities of A_1 and A_2. A lower bound for this threshold is derived using (18) and (19)

$$\frac{(h_1 - h_2)^2}{\bar{\sigma}^2} = \frac{1}{p_1 p_2}. \tag{20}$$

Because $p_1 + p_2 = 1$, and $p_1 p_2 \leq \frac{1}{4}$,

$$(h_1 - h_2)^2 \geq 4\bar{\sigma}^2, \tag{21}$$

which yields

$$|h_1 - h_2| \geq 2\bar{\sigma}. \tag{22}$$

If (22) is satisfied, then we say that there is an edge within the detection circle.

Once an edge has been detected, its angle of orientation, α, and normal distance from the center of the detection circle, L, are calculated. Refer to Fig. 1. Edges are described by their normal equation with respect to the coordinate system of the detection circle

$$\begin{array}{ll} x\cos\alpha + y\sin\alpha = -L & \text{if } p_1 \leq p_2 \\ x\cos\alpha + y\sin\alpha = L & \text{otherwise.} \end{array} \tag{23}$$

Values for $\cos\alpha$ and $\sin\alpha$ are calculated using the center of gravity $G = (G_x, G_y)$ of the data within the detection circle. The coordinates of the center of gravity are calculated using

$$G_x = \frac{1}{M_1}\sum_x\sum_y x\mathcal{H}(x,y)\omega(x - d_x, y - d_y) \tag{24}$$

$$G_y = \frac{1}{M_1}\sum_x\sum_y y\mathcal{H}(x,y)\omega(x - d_x, y - d_y). \tag{25}$$

The summations in (24) and (25) are performed for all pixels (x, y) within the detection circle. The angle α can be calculated using

$$\tan(\alpha) = \frac{G_y}{G_x}. \tag{26}$$

Referring to Fig. 1, the normal distance L is found by calculating the area enclosed between the edge line and the detection circle, i.e., the area, a_2, of region A_2. As shown in [3], this leads to the following solutions for L,

$$r^2 \arcsin\left(\frac{\sqrt{r^2 - L^2}}{r}\right) - L\sqrt{r^2 - L^2} - a_2 = 0 \quad \text{if } p_1 \geq p_2 \tag{27}$$

$$0.5\pi r^2 - L\sqrt{r^2 - L^2} - r^2 \arcsin\left(\frac{L}{r}\right) - a_2 = 0 \quad \text{otherwise.} \tag{28}$$

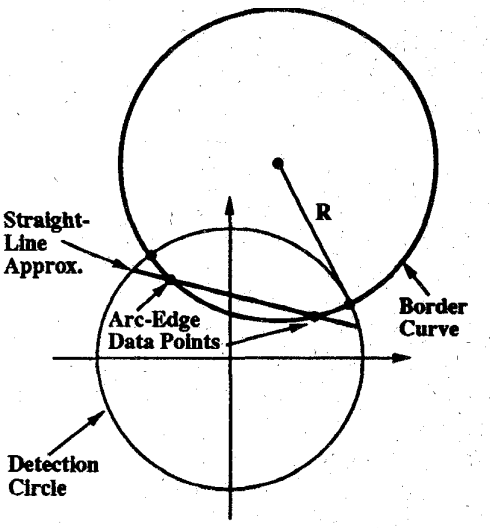

Fig. 2. Arc-edge data points.

C. Arc-Edge Data Point Calculation

In the previous section the circular border curve was approximated using line segments defined by the parameters α and L. In this section, we will show how these parameters are used to calculate the coordinates of the data points that lie on the circular border curve. These data points will be called arc-edge data points. As shown in Fig. 2, arc-edge data points are the intersection points of the straight-line approximation of the border curve and the border curve itself.

We assume that the border curve can be approximated by a circle of radius R, which can be described by

$$(x - X_0)^2 + (y - Y_0)^2 = R^2 \tag{29}$$

where (X_0, Y_0) are the coordinates of the center of the circle. In [3], Tchoukanov *et al.* derive a geometric relationship between the locations of the arc-edge data points and the parameters of the approximating line, based on the assumption that the position of the arc-edge data point is a weak function of R (i.e., the choice of R has so little effect on the position of the arc-edge data points that it can be ignored). This assumption is used to derive equations that allow us to calculate the coordinates of the arc-edge data points. The following derivation of these equations follows that of Tchoukanov *et al.* in [3].

Fig. 3 illustrates the arc-edge data points in greater detail. The circular border curve is centered at the point (X_0, Y_0), has radius R, and intersects the detection circle at the points (x_1, y_1) and (x_2, y_2). The arc-edge data points are located at (x_3, y_3) and (x_4, y_4). We now derive equations to calculate the coordinates of (x_3, y_3) and (x_4, y_4), given L and α.

To facilitate this derivation, the detection circle is rotated counterclockwise through an angle of $(0.5\pi - \alpha)$ to align the normal L with the Y-axis. Let (X_0', Y_0') be the coordinates of the center of the rotated circular arc. The coordinates of the rotated arc-edge points, (x_3', y_3') and $(-x_3', y_3')$, are given by

$$x_3' = x_3\sin\alpha - y_3\cos\alpha \tag{30}$$

$$y_3' = L = x_3\cos\alpha + y_3\sin\alpha. \tag{31}$$

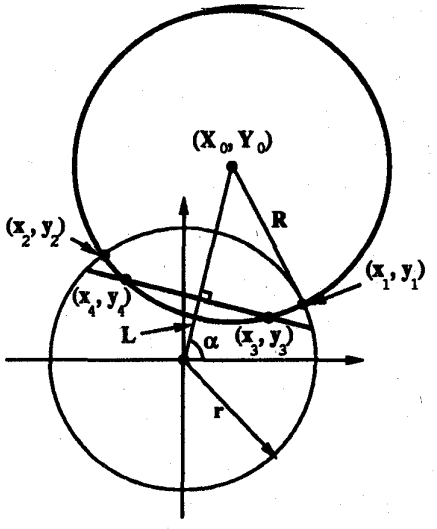

Fig. 3. Parameters used in arc-edge data point calculation.

The area of region A_2 is

$$a_2 = p_2 r^2 \pi = 2 \int_0^{x_1'} \int_{Y_0' - \sqrt{R^2 - x^2}}^{\sqrt{r^2 - L^2}} dx dy. \tag{32}$$

Tchoukanov *et al.* [3] derive the following equations for x_3' by solving the integral in (32)

$$\left.\begin{aligned} r^2 \arcsin \frac{x_1'}{r} + R^2 \arcsin \frac{x_1'}{r} - x_1' Y_0' - a_2 = 0 \\ \text{if } p_1 \geq p_2 \\ 0.5\pi r^2 + r^2 \arccos \frac{x_1'}{r} + R^2 \arcsin \frac{x_1'}{r} - x_1' Y_0 - a_2 = 0 \\ \text{(otherwise)} \end{aligned}\right\} \tag{33}$$

where

$$Y_0' = L + \sqrt{R^2 - x_3'^2} \tag{34}$$

$$x_1' = \sqrt{r^2 - \left(\frac{r^2 - L^2 + 2LY_0' - x_3'^2}{2Y_0'} \right)^2}. \tag{35}$$

Tchoukanov *et al.* have used the approximation

$$K = \frac{x_3'}{\sqrt{r^2 - L^2}} \tag{36}$$

where a look-up table of K values is created off-line, indexed on values of L in the range $(-4.5 \cdots 4.5)$ pixel units. The coordinates of the arc-edge data points (x_3, y_3) and (x_4, y_4) are calculated with

$$x_3 = L \cos \alpha + K \sqrt{r^2 - L^2} \sin \alpha \tag{37}$$

$$y_3 = L \sin \alpha - K \sqrt{r^2 - L^2} \cos \alpha \tag{38}$$

$$x_4 = L \cos \alpha - K \sqrt{r^2 - L^2} \sin \alpha \tag{39}$$

$$y_4 = L \sin \alpha + K \sqrt{r^2 - L^2} \cos \alpha. \tag{40}$$

The complete subpixel arc-edge detector is illustrated in Fig. 4. Given an input image $\mathcal{H}$, we perform bilevel thresholding and simple edge detection to create an edge map. Moment-preserving straight-line-edge detection is performed

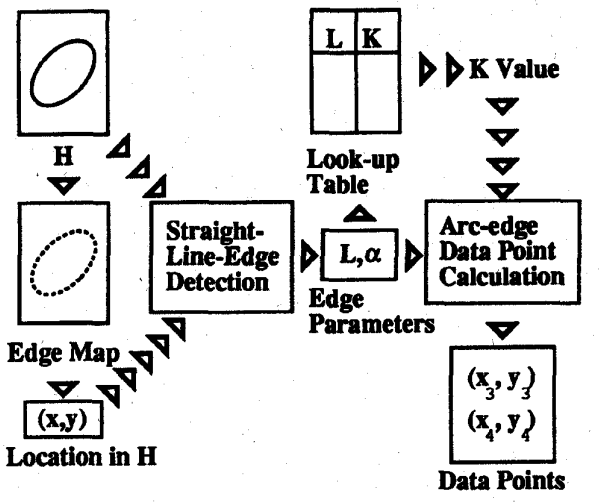

Fig. 4. Arc-edge data point calculation.

on $\mathcal{H}$ at locations specified in the edge map, producing a list of straight-line segments that approximate the circular border curve. Each line segment is defined by its normal distance from the center of the detection circle L and its angle of orientation α. The coordinates of the arc-edge data points are calculated with (37)–(39), and (40).

V. Ellipse Parameter Estimation

Under perspective projection, a circle in the scene will appear as an ellipse in the digital image; therefore, we use the data point list that was described in Section IV to perform ellipse parameter estimation. Let $\mathcal{P} = \{P_1 \cdots P_n\}$ represent the list of n data points. Given $\mathcal{P}$, our task is to estimate the center point coordinates (X_0, Y_0), the major axis length $\mathcal{A}$, minor axis length $\mathcal{B}$, and the angle or orientation Θ of the ellipse that fits the data points.

Various methods have been reported for ellipse parameter estimation, including [4], [17]–[19]. We use the area-based parameter estimation algorithm described by Safaee-Rad *et al.* [4]. Parameter estimation proceeds as follows. In the first step, we estimate the parameters of an initial optimal ellipse. These parameters are used to generate weights for the data points. The weights normalize the contribution of each data point to the parameter estimation. In the final step, the weighted data points are used to find the parameters of the ellipse.

We write the implicit equation of an ellipse as

$$\mathcal{W}(X, Y) = 0 \tag{41}$$

where

$$\mathcal{W}(X, Y) = aX^2 + bXY + cY^2 + dX + eY + 1. \tag{42}$$

The typical approach to fitting an ellipse to data points is to minimize an error residual $\mathcal{J}_0$, given by

$$\mathcal{J}_0 = \sum_{i=1}^{n} [\mathcal{W}(X_i, Y_i)]^2. \tag{43}$$

It is well known that (42) does not give the geometric distance from the point (X, Y) to the ellipse given by (41), and

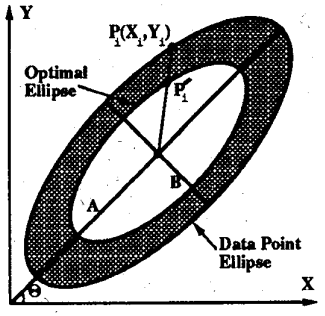

Fig. 5. The area difference between two concentric ellipses.

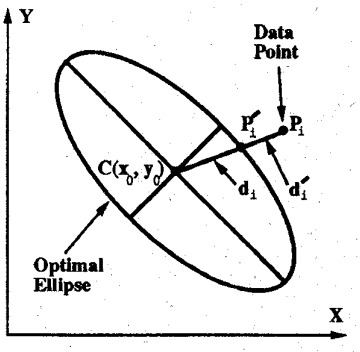

Fig. 6. Distances d_i and d_i' of an ellipse.

therefore, that minimizing (43) gives an ellipse that minimizes the *algebraic* distance from the ellipse to the data points, not the *geometric* distance. To correct for this, Safaee Rad *et al.* introduce an error function that weights each data point's contribution in the minimization according to the ratio between the point's geometric and algebraic distance to the ellipse [4]. We summarize their derivation here.

First, we will derive a new error function, $\mathcal{J}_1$, that is based on the difference in the areas of two concentric ellipses with equal eccentricity. We then show that minimizing $\mathcal{J}_1$ is equivalent to minimizing (43). Finally, we construct $\mathcal{J}_1'$ by weighting the data points based on the ratio of geometric versus algebraic distance.

Let $(\mathcal{A}, \mathcal{B}, \theta, X_0, Y_0)$ be the parameters of the ellipse that best fits the data points, and $(\mathcal{A}', \mathcal{B}', \theta, X_0, Y_0)$ be the parameters of the ellipse passing through a given data point $P_i = (X_i, Y_i)$. These ellipses will be referred to as the optimal ellipse and the data point ellipse. The two ellipses are concentric and have the same eccentricity and orientation (see Fig. 5).

If D' is the area of the data point ellipse and D is the area of the optimal ellipse, then an error function can be defined as the difference between these areas as

$$e_i = D - D'. \tag{44}$$

Consider a line that passes through a data point $P_i = (X_i, Y_i)$ and the center point (X_0, Y_0). Let $P_i' = (X_i', Y_i')$ be the intersection point of this line and the optimal ellipse. To aid in this discussion, we define the following quantities. Let d_i' be the distance from the center of the ellipse to the point P_i' and let d_i be the distance from the center of the ellipse to the point P_i.

Given that the two concentric ellipses are similar, i.e., they have the same orientation angle and eccentricity, an expression for the area difference of the ellipses can be derived from (44) as follows

$$\begin{aligned}\Delta \text{ Area} \doteq e_i &= D - D' \\ &= \pi\mathcal{A}\mathcal{B} - \pi\mathcal{A}'\mathcal{B}' \\ &= \pi\mathcal{A}'\mathcal{B}'\left(\frac{\mathcal{A}\mathcal{B}}{\mathcal{A}'\mathcal{B}'}\right)\left(1 - \frac{\mathcal{A}'\mathcal{B}'}{\mathcal{A}\mathcal{B}}\right) \\ &= \pi(\mathcal{A}'\mathcal{B}')\frac{d_i^2}{d_i'^2}\left(1 - \frac{d_i'^2}{d_i^2}\right).\end{aligned} \tag{45}$$

Bookstein has shown that the following proportionality holds [20]

$$\mathcal{W}(X_i, Y_i) \propto \left[\frac{\delta_i}{d_i'}\left(\frac{\delta_i}{d_i'} + 2\right)\right] \tag{46}$$

where $\delta_i = d_i - d_i'$. After a bit of algebraic manipulation, we obtain

$$\left[\frac{\delta_i}{d_i'}\left(\frac{\delta_i}{d_i'} + 2\right)\right] = \frac{d_i^2}{d_i'^2}\left(1 - \frac{d_i'^2}{d_i^2}\right). \tag{47}$$

Therefore, (45) is proportional to (42), and consequently, we may now define a new error function, $\mathcal{J}_1$, in terms of (45)

$$\mathcal{J}_1 = \sum_{i=1}^{n}\left[\frac{1}{\pi\mathcal{A}'\mathcal{B}'}(D - D_i')\right]^2 = \sum_{i=1}^{n}\left[\frac{e_i}{\pi\mathcal{A}'\mathcal{B}'}\right]^2. \tag{48}$$

Thus, minimizing (48) is equivalent to minimizing (43).

Distance d_i' is maximum for points along the ellipse's major axis and minimum for points along its minor axis; therefore, (45) is maximum for data points near the minor axis of the ellipse and minimum for data points near the major axis of the ellipse. For this reason, the contributions of the data points are normalized by defining a weighting factor that is a function of each data point's position relative to the major axis of the optimal ellipse. Let d_i be the distance of a data point P_i to the optimal ellipse, i.e., distance $|\overline{P_iP_i'}|$ in Fig. 6. The geometric distance from the center of the optimal ellipse to the optimal ellipse, distance $\overline{(X_0, Y_0)P'}$ in Fig. 6, is represented by d_i'. Using the expression for the error e_i of a data point given by (45), after some algebraic manipulation, we obtain

$$\begin{aligned}e_i &= \pi(\mathcal{A}'\mathcal{B}')\frac{d_i^2}{d_i'^2}\left(1 - \frac{d_i'^2}{d_i^2}\right) \\ &= \pi(\mathcal{A}'\mathcal{B}')\frac{d_i^2}{d_i'^2}\left[\frac{\delta^2 + 2\delta d_i'}{d_i^2}\right].\end{aligned} \tag{49}$$

Equation (49) is the general expression for the error due to data point P_i. This error will be a minimum when P_i is near the major axis of the ellipse. If P_i is on the major axis of the optimal ellipse and has the same distance d_i, the error will be given by e_2

$$e_2 = \pi(\mathcal{A}'\mathcal{B}')\frac{A^2}{A'^2}\left[\frac{\delta^2 + 2\delta A'}{A^2}\right]. \tag{50}$$

The ratio e_2/e_i is given by

$$\frac{e_2}{e_i} = \frac{A^2}{A'^2}\frac{d_i'^2}{d_i^2}\frac{\left[\frac{\delta^2+2\delta A'}{A^2}\right]}{\left[\frac{\delta^2+2\delta d_i'}{d_i^2}\right]} = \frac{d_i'}{A'}\frac{\left(\frac{\delta}{2A'}+1\right)}{\left(\frac{\delta}{2d_i'}+1\right)} \approx \frac{d_i'}{A'} = \psi_i \tag{51}$$

where the final approximation follows because typically δ is much smaller than either A' or d_i'. We now construct an error function using these weights as follows

$$\mathcal{J}_1' = \sum_{i=1}^{n}\left[\psi_i\frac{1}{\pi\mathcal{A}\mathcal{B}}(D-D_i')\right]^2 = \sum_{i=1}^{n}[\psi_i\mathcal{W}(X_i,Y_i)]^2. \tag{52}$$

The error function described in (52) minimizes the area-based error function described by (44), and normalizes the contributions of the data points.

Equations for the ellipse parameters can be derived by computing the derivatives of $\mathcal{J}_1'$ with respect to the unknowns, leading to

$$X_0 = \frac{2cd-be}{b^2-4ac} \tag{53}$$

$$Y_0 = \frac{2ae-bd}{b^2-4ac} \tag{54}$$

$$\theta = \arctan\left[\frac{(c-a)+\sqrt{(c-a)^2+b^2}}{b}\right] \tag{55}$$

$$\mathcal{A}^2 = \left[\frac{2(1-F_s)}{b^2-4ac}\right][(c+a)+\sqrt{(c-a)^2+b^2}] \tag{56}$$

$$\mathcal{B}^2 = \left[\frac{2(1-F_s)}{b^2-4ac}\right][(c+a)-\sqrt{(c-a)^2+b^2}], \tag{57}$$

where

$$F_s = \frac{bde-ae^2-cd^2}{b^2-4ac}. \tag{58}$$

Thus, we have a two-stage algorithm. In the first stage, $\mathcal{J}_1$ is used to generate the parameters of an initial estimate of the ellipse. These parameters are then used to estimate the value of A' and d_i' for each data point. The data point weighting values ψ_i are calculated using A' and d_i'. The weighted data points are used in $\mathcal{J}_1'$, in the second stage, to find the parameters of the final optimal ellipse. This method of parameter estimation is noniterative and produces good results.

VI. Experimental Results

Our method of ellipse parameter estimation was tested on real image sequences of circles with diameters of one-eighth inch, one-fourth inch, one-half inch, one inch, and two inches. The high-resolution parameter estimation method produced good results for each circle. Fig. 7 shows the results of parameter estimation for a one-inch-diameter circle.

For each image sequence, our algorithms were used to estimate the major and minor axis lengths for the image ellipses. Given these estimates, in pixel units, an estimate of the axis length in inches is calculated in the following manner. First, the endpoints of each axis are located in the image. Then, Tsai's camera calibration [7] method was used to compute the world coordinates that correspond to the axis endpoints. Finally, each axis length is calculated as half the Euclidean distance between the world coordinates of its endpoints.

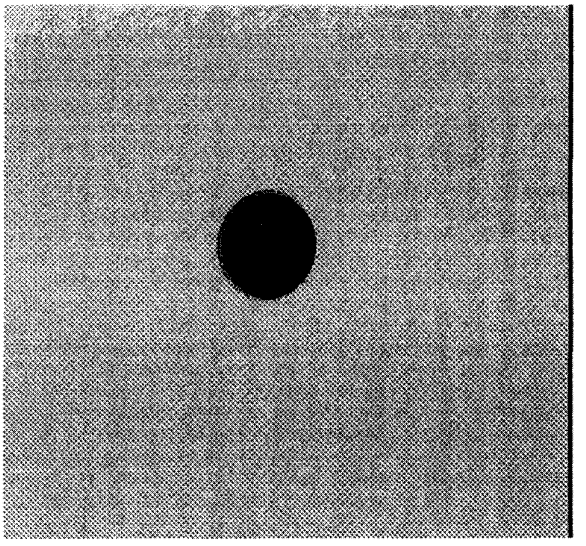

Fig. 7. Estimated ellipse plotted onto 1-in-diameter circle.

For comparison, results for five different methods of edge detection were obtained. In simple edge detection, we perform moment-based binary thresholding, see [16], on the high-resolution image produced by the superresolution algorithm. In the binary image, pixels that border regions of different intensities are treated as data points. Canny 1 edge detection is the Canny edge detector without subpixel interpolation. Canny 2 edge detection is the Canny edge detector with subpixel interpolation. The interpolation is accurate to within a tenth of a pixel dimension. Both the Canny 1 and Canny 2 edge detectors were run on low resolution input images. To investigate the benefit of performing superresolution, the arc-edge detector was also run on low-resolution input images. High-resolution edge detection is the method of edge detection that was described in Section IV and is used in our method of ellipse parameter estimation. This edge detection method performs arc-edge data point detection on the high-resolution image produced by the superresolution algorithm. In each experiment, parameter estimation was performed using the area-based algorithm described in Section V.

Table I lists the average percent error of the axis length estimates. The percent error, Δ, is calculated using

$$\Delta = \frac{(|\text{Estimated } \mathcal{A}-\mathcal{R}|+|\text{Estimated } \mathcal{B}-\mathcal{R}|)}{2\times\mathcal{R}}\times 100 \tag{59}$$

where $\mathcal{R}$ is the actual radius of the circle.

The high-resolution method consistently provides accurate parameter estimates. The greatest benefit is seen when inspecting small circles. The high-resolution method benefits from both subpixel accuracy and superresolution. The improvement

TABLE I
AVERAGE PERCENT ERROR

Average Percent Error of Circle Radius Estimates.					
Method	Circle Diameter in Inches.				
	0.125	0.25	0.5	1	2
Simple	23.4313	10.3163	5.76656	2.6687	1.51665
Canny 1	22.3480	11.0073	5.92641	2.97288	1.44266
Canny 2	20.9358	10.5905	5.87558	2.91241	1.42978
Arc Edge	22.4000	11.5330	5.14937	2.51762	1.60457
High Res.	1.5660	0.8528	1.04134	1.75634	1.48155

gained by using the high-resolution inspection method decreases as the radius of the circle increases. For large circles, the methods provide virtually the same results.

VII. CONCLUSION

We have presented a new method of parameter estimation for circular shapes that uses image sequences. In this method, an image sequence is used to create a fused, high-resolution image. A moment-based edge detector that locates points that lie along a circular arc with subpixel accuracy is used to locate data points in the high-resolution image, creating a data point list. Given the data point list, parameter estimation is performed using an area-based ellipse parameter estimation algorithm. Once the ellipse parameters have been estimated, camera calibration techniques are used to translate distances in the image plane into distances in the real-world results.

REFERENCES

[1] K. Fu, R. Gonzalez, and C. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*. New York: McGraw-Hill, 1987.

[2] M. Irani and S. Peleg, "Improving resolution by image registration," *CVGIP: Graphical Models and Image Processing*, pp. 231–239, May 1991.

[3] I. Techoukanov, R. Safaee-Rad, B. Benhabib, and K. Smith, "Subpixel edge detection for accurate estimation of elliptical shape parameters," in *CSME Mech. Eng. Forum 1990, Proc.*, Univ. of Toronto, 1990, pp. 313–318.

[4] ——, "Accurate parameter estimation of quadratic curves from grey-level images," *CVGIP: Image Understanding*, vol. 54, pp. 259–274, Sept. 1991.

[5] D. Keren, S. Peleg, and R. Brada, "Image sequence enhancement using sub-pixel displacements," in *IEEE Conf. Computer Vision and Pattern Recognition*, 1988, pp. 742–746.

[6] G. Jacquemod, C. Odet, and R. Goutte, "Image resolution enhancement using subpixel camera displacement," *Signal Processing*, vol. 26, pp. 139–146, Jan. 1992.

[7] R. Y. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf television cameras and lenses," *IEEE J. Robot. Automat.*, vol. RA-3, pp. 323–344, Aug. 1987.

[8] M. Irani and S. Pelegm, "Image sequence enhancement using multiple motions analysis," *IEEE*, pp. 216–221, Mar. 1992.

[9] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*, vols. I and II. Reading, MA: Addison-Wesley, 1993.

[10] A. Tabatabai and O. R. Mitchell, "Edge location to subpixel values in digital imagery," *PAMI*, vol. 6, pp. 188–200, Mar. 1984.

[11] W. Niblack, D. Petkovic, and M. Flickner, "Projection-based high accuracy measurement of straight line edges," *Machine Vision and Applicat.*, pp. 183–199, Jan. 1988.

[12] W. Niblack and D. Petkovic, "On improving the accuracy of the Hough transform," *Machine Vision and Applicat.*, pp. 87–106, Mar. 1990.

[13] J. Carbone, "Subpixel interpolation wih charge injection devices (cid's)," *SPIE: Optical Sensors and Electronic Photography*, vol. 1071, pp. 80–89, 1989.

[14] E. P. Lyvers, O. R. Mitchell, M. L. Akey, and A. P. Reeves, "Subpixel measurements using a moment-based edge operator," *IEEE Trans Pattern Anal. Machine Intell.*, vol. 1, pp. 1293–1309, Dec. 1989.

[15] S. Ghosal and R. Mehrotra, "Orthogonal moment operators for subpixel edge detection," *Pattern Recognition*, vol. 26, pp. 295–306, Feb. 1993.

[16] W. Tsai, "Moment-preserving thresholding: A new approach," *Computer Vision, Graphics and Image Processing*, vol. 29, pp. 377–393, Mar. 1985.

[17] S. Tsuji and F. Matsumoto, "Detection of ellipses by a modified Hough transformation," *IEEE Trans. Computers*, vol. C-27, pp. 777–781, Aug. 1978.

[18] R. Takiyama and N. Ono, "An iterative procedure to fit an ellipse to a set of points," *IEICE Trans. Communicat. Electron. Inform., Syst.*, vol. 74, pp. 3041–3045, Oct. 1991.

[19] T. Nagata, H. Tamura, and K. Ishibashi, "Detection of an ellipse by use of a recursive least-squares estimator," *J. Robot. Syst.*, vol. 2, pp. 163–177, Feb. 1985.

[20] F. L. Bookstein, "Fitting conic sections to scattered data," *Computer Graphics and Image Processing*, vol. 9, pp. 59–71, 1979.

James M. Reed (S'95) received the B.S. degree in computer engineering in 1990 and the M.S. degree in electrical engineering in 1994 from the University of Illinois at Urbana-Champaign. He is currently pursuing the Ph.D. degree in computer science at the University of Michigan, Ann Arbor.

In the course of completing undergraduate studies at the University of Illinois, he developed a strong interest in the fields of artificial intelligence, robotics, and computer vision. This interest led him to perform research in the area of automated optical inspection during Masters studies. Learning how computer technology and artificial intelligence techniques can be used to benefit the industrial community led him to consider the question of how computer technology and AI techniques can be used to benefit the educational community. Toward this end, he is currently doing research in the area of educational technology as a member of the Highly-Interactive Computing Research Group at the University of Michigan.

Seth Hutchinson (S'85–M'88) received the Ph.D. degree from Purdue University in West Lafayette, IN, in 1988.

He spent 1989 as a Visiting Assistant Professor of Electrical Engineering at Purdue University. In 1990 Dr. Hutchinson joined the faculty at the University of Illinois in Urbana-Champaign, where he is currently an Assistant Professor in the Department of Electrical and Computer Engineering, the Coordinated Science Laboratory, and the Beckman Institute for Advanced Science and Technology.