Proceedings of the 1994 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI'94) • Las Vegas, NV • Oct. 2-5, 1994

# Subpixel Parameter Estimation For Elliptical Shapes Using Image Sequences

James Michael Reed Seth A. Hutchinson
Dept. of Electrical and Computer Engineering
University of Illinois, Urbana, IL 61801

**Abstract: We present a method of ellipse parameter estimation that can be used in performing automated inspection of circular features. In our method, several digital images are taken of each part as it moves past a camera, creating an image sequence. Image enhancement is performed using the image sequence, yielding a high-resolution image. Subpixel edge detection is performed on the high-resolution image, producing a set of data points that is used for ellipse parameter estimation.**

## 1 Introduction

This research presents a method of ellipse parameter estimation that applies image processing algorithms to the problem of automated optical inspection (AOI). Our research was conducted for use in performing 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 our inspection system, circuit boards are moving along a conveyer at a fixed velocity. Above the conveyer, a CCD camera examines each part as it moves past. Although the conveyer is constantly moving, the shutter speed of a typical CCD camera can be set to capture a clear image, for reasonable conveyer speeds.

Several pictures are taken of each part as it passes beneath the camera, creating an image sequence $\mathcal{Q} = \{q_{(1)} \cdots q_{(n)}\}$. The via's 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 [2].

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. The parameters of an ellipse are: the coordinates of the center point, $(X_0, Y_0)$, the length of the major axis, $\mathcal{A}$, the length of the minor axis $\mathcal{B}$, and the angle between the major axis and the horizontal axis of the reference frame, $\Theta$.

Our method combines image enhancement, subpixel edge detection, and subpixel parameter estimation of ellipses to perform the inspection task described above. Figure 1 presents an overview of our method. Given the input image sequence $\mathcal{Q}$, we perform image enhancement using Peleg and Irani's superresolution algorithm [5]. The superresolution algorithm creates a high-resolution image $\mathcal{H}$ that has twice the resolution

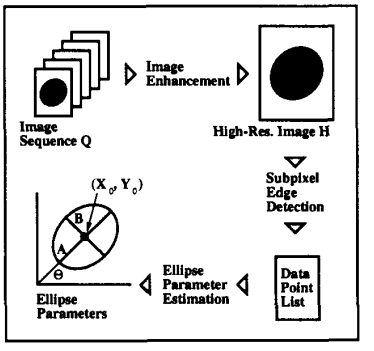


Figure 1: Algorithm overview.

of the individual images in $\mathcal{Q}$. 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 [14]. These data points are used to perform ellipse parameter estimation. The parameter estimation algorithm is noniterative, and operates by minimizing the difference between the areas of two ellipses [11]. 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.

Each phase of the inspection algorithm is discussed in a different section. The superresolution algorithm is covered in Section 2. Subpixel arc-edge detection is discussed in Section 3. Section 4 explains ellipse parameter estimation. Experimental procedures and results are presented in Section 5. Section 6 presents conclusions.

## 2 Image Enhancement

Given the input image stream $\mathcal{Q}$, we create a high-resolution image of the part that is being inspected. Image stream analysis facilitates both noise compensation and resolution improvement. Noise compensation yields an image that is the same resolution as the input images, but is a more faithful representation of the scene. Resolution improvement algorithms use information about the imaging system and the image stream to create a high-resolution image [7][5].

0-7803-2072-7/94 $4.00 © 1994 IEEE

The method we use is based on the work of Peleg and Irani [5], which is an iterative approach to resolution improvement. They propose a superresolution algorithm that creates a high-resolution image that has twice the resolution of the input images. An alternative approach is given in [7], which describes a system that uses subpixel camera displacements to create the high-resolution image.

The superresolution algorithm proceeds as follows. First, we perform image registration on the input image sequence $\mathcal{Q}$. Image registration is the process of bringing the input images into alignment. When the images are aligned, the pixels corresponding to the circular feature being inspected occupy the same locations in each image. Image registration produces a new image sequence, $\mathcal{Q}_a$, in which all of the images are aligned. We use the aligned image sequence to create an initial estimate of the high resolution image. This initial estimate has twice the resolution of the input images, and is created by averaging pixel values from the aligned image sequence. In the last part of the superresolution algorithm, we improve the estimate in an iterative process that is discussed in Section 2.2.

## 2.1 Image Registration

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 $(\rho_x, \rho_y)$ is the fractional part. Motion vector estimation is performed by repeating the following steps. First, we shift the image according to its motion estimate. Second, we compute a correction to the motion estimate by solving a system of linear equations given by (1). The correction is added to the current motion estimate, and the process is repeated. When the changes to the motion estimate are less than a threshold value, motion estimation stops.

In general, motion in two dimensions can be described by a translation in the $x$-direction, $a$, a translation in the $y$-direction, $b$, and a rotation, $\theta$. The rotation is assumed to be about an axis located at the center of each image, and the translations are expressed in pixel units. We include the rotational component of the motion $\theta$ for generality; however, we do not use $\theta$ in our experiments because we assume that the motion is purely translational. This is a reasonable assumption because the motion is due to the movement of the conveyer belt in our inspection system. In terms of the motion vector $T$, the components of the motion are $a = t_x + \rho_x$ and $b = t_y + \rho_y$.

Motion vectors for each input image in $\mathcal{Q}$ are estimated in an iterative manner. A derivation of the motion estimation equations is presented by Keren et al. in [8]. Motion parameters are estimated by solving the following system of linear equations

$$\left.\begin{array}{rcl} \sum D_x^2 a + \sum D_x D_y b + \sum A D_x \theta &=& \sum D_x D_t \\ \sum D_x D_y a + \sum D_y^2 b + \sum A D_y \theta &=& \sum D_y D_t \\ \sum A D_x a + \sum A D_y b + \sum A^2 \theta &=& \sum A D_t \end{array}\right\} \quad (1)$$

where

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

In (1), we use these first-order approximations $D_x = q_{(r)}(x+1, y) - q_{(r)}(x,y)$ and $D_y = q_{(r)}(x, y+1) - q_{(r)}(x,y)$. Solving the system of equations described by (1) for $a$, $b$, and $\theta$ yields a vector that is used to update the current motion estimate. Because (1) is based on the first few terms of the Taylor's expansion, motion estimates are only valid for small displacements. The motion estimate is updated at the end of each iteration until the magnitude of the correction vector is below a predefined threshold.

Given the initial estimate, the motion estimation algorithm uses the sequence of equations described in (1) to improve the estimate. The estimation process proceeds quickly and typically converges to a solution in five or six iterations. We estimate the motion vector for each image in the input image sequence $\mathcal{Q}$, creating a list of motion vectors, $\mathcal{L}$, that is used to improve the initial estimate, $\mathcal{H}$.

## 2.2 Creating A High-Resolution Image

Iterative refinement is used to enhance the initial estimate of the high-resolution image. Given an initial estimation of the high resolution image, $\mathcal{H}$, the list of motion vectors, $\mathcal{L}$, and the input image sequence, $\mathcal{Q}$, the following steps are performed for each iteration of the refinement algorithm. A sequence of low resolution images $\mathcal{S} = \{s_{(1)} \ldots s_{(n)}\}$ is created by subsampling $\mathcal{H}$, then shifting the 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., $\mathcal{S} = \mathcal{Q}$. The difference images between $\mathcal{Q}$ and $\mathcal{S}$ are calculated, creating a sequence of difference images:

$$\mathcal{Q} - \mathcal{S} = \{(q_{(1)} - s_{(1)}) \ldots (q_{(n)} - s_{(n)})\}. \quad (2)$$

Corrections to the high-resolution estimate are based on the values of these (2) difference images. Our goal is to find a high-resolution estimate that could have produced the low resolution images in the input image sequence $\mathcal{Q}$. Toward this end, an updating formula that is derived from a model of the imaging system is used to calculate improvements to $\mathcal{H}$. The updating formula is discussed in Section 2.2.3.

### 2.2.1 Initial Estimate

The aligned image sequence $\mathcal{Q}_a$ is created by shifting each input image in $\mathcal{Q}$ by its motion estimate in the final stage of image registration; therefore, the images in $\mathcal{Q}_a$ are aligned such that the pixels corresponding to the elliptical feature occupy the same locations in every image. The initial high-resolution estimate, $\mathcal{H}$, is created by averaging the values of $\mathcal{Q}_a$.

$$avg(\mathcal{Q}_a(x,y)) = \frac{1}{n} \sum_{q_a \in \mathcal{Q}_a} q_a(x,y). \quad (3)$$

Because $\mathcal{H}$ has twice the resolution of $\mathcal{Q}_a$, each pixel in $\mathcal{Q}_a$ corresponds to four pixels in $\mathcal{H}$. The initial high-resolution estimate is improved using a model of the imaging process that is described in section 2.2.3.

### 2.2.2 Imaging Process Model

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 [6]:

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

In (4), $q_{(k)}$ is the $k$th image of the image stream $\mathcal{Q}$. The goal of the superresolution algorithm is to recover the high-resolution image, $f$, of the scene. The intensity of a high-resolution pixel $q_{(k)}(x,y)$ is a function of the intensities of several high-resolution pixels $f(i,j)$. Equation (4) represents a 2D geometric 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 function. 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 digitizing function also accounts for the displacement of the $k$th frame in the image sequence.

### 2.2.3 Enhancing the High-Resolution Estimate

The imaging process model (4) describes how low resolution input images are produced. An expression describing how simulated images are created can also be derived:

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

In (5), $s^n$ is a low resolution image, and $\mathcal{H}^n$ is the high-resolution image produced in the $n$th 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 $\beta$. 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 $(i,j)$.

Each simulated low resolution image $s^n_{(i)}$ is created by shifting $s^n$ (5) 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.

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

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

In (6), $q_{(k)}$ is the $k$th image of the input image stream $\mathcal{Q}$, and $s^n_{(k)}$ is the $k$th image of the simulated image stream that was created in the $n$th iteration of the enhancement algorithm. The function $h^{BP}$ represents a back-projection kernel that is used to calculate improvement values for the high-resolution estimate. The difference between $h^{BP}$ and the point spread function, $h^{PSF}$ in (5), is that $h^{PSF}$ describes properties of the imaging system, while $h^{BP}$ may be chosen arbitrarily [5]. The contribution of the back-projection kernel is normalized using a constant, $c$. The choice of $h^{BP}$ is discussed in [5]. We used a normalized, $5 \times 5$ 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 $\alpha$. Given the high resolution estimate in the $n$th 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 $\mathcal{H}^n$.

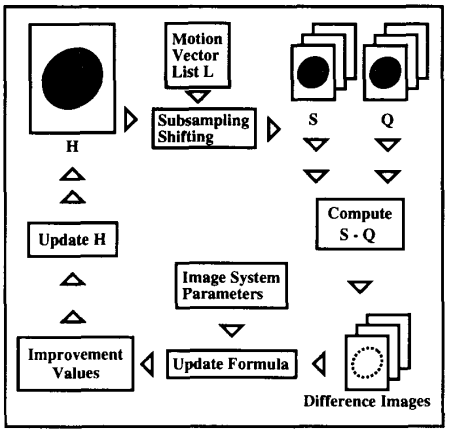


Figure 2: The refinement algorithm.

The following steps are performed in each iteration of the refinement algorithm, illustrated in Figure 2. First, a sequence of simulated images, $\mathcal{S}$, is created using the high resolution estimate $\mathcal{H}$ and the list $\mathcal{L}$ of motion vectors. The difference images (2) are then calculated. The sequence of difference images and several imaging system parameters are used in the updating formula (6) to compute correction values for $\mathcal{H}$. Iterative refinement halts when there are no more changes required in the high-resolution estimate.

The superresolution algorithm produces a high-resolution image $\mathcal{H}$ that has twice the resolution of the input images. The next step in our parameter estimation algorithm is to perform subpixel edge detection on $\mathcal{H}$. This will produce a list of data points that will be used for parameter estimation as described in Section 4.

## 3 Subpixel Edge Detection

Given the high-resolution image $\mathcal{H}$ we use subpixel edge detection to produce a set of data points. There are many methods of edge detection (see e.g. [4]). 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 [12]. The limited resolution of early edge detectors led to the development of subpixel edge detection algorithms. Subpixel edge detectors localize edges to within a fraction of a pixel precision. Subpixel algorithms are warranted when observed features are small enough to be considerably distorted by the digitization process. Tasks requiring subpixel accuracy include photogrammetry, lithography, satellite image processing, manufacturing, and automated inspection of microelectronic components [3].

Our work is based on the arc-edge detector described in [14]. In the first step of our method, 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 [12]. 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 for performing ellipse parameter estimation, which is described in Section 4.

## 3.1 Building the 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 applying thresholding and simple edge detection to $\mathcal{H}$. Tsai's sample moment preserving bilevel thresholding algorithm is used to threshold $\mathcal{H}$ [16]. This algorithm locates the threshold such that the first four sample moments are preserved. The thresholding operation produces a binary image. Simple edge detection, where intensity changes in the binary image are treated as edges, is done on the binary image to produce the edge map.

## 3.2 Straight Line Edge Detection

Given the high-resolution estimate, $\mathcal{H}$, and the edge map, 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 [12] [14].

The straight line edge detector detects lines that are located within a circular detection area that consists of 69 pixels that are weighted to approximate a circle of radius one. 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 Figure 3. Let $r$ represent the radius of the detection circle. A straight-line edge divides the detection circle into two regions, $A_1$ and $A_2$, respectively. 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 ordered triples of the form $(a_i, h_i, p_i)$. The parameters of these

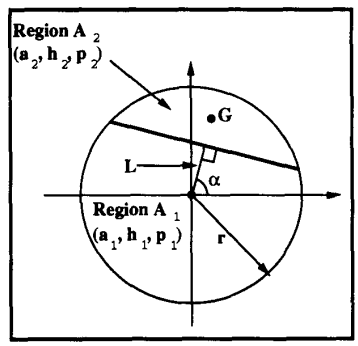


Figure 3: The parameters of the edge model.

triples are as follows. The area of the region $i$ is denoted by $a_i$, and the characteristic intensity of region $i$ is $h_i$. The third parameter, $p_i$, denotes 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 $p_i$. Once an edge has been located, its angle of orientation, $\alpha$, and normal distance from the center of the detection circle, $L$, are calculated.

The straight line edge detector is based on sample moments. Edges are located such that the first four sample moments of the image data, within the detection circle, are preserved. Tabatabai and Mitchell present solutions for the edge parameters, $a_i$, $h_i$, and $p_i$ in [14]. Given these parameters, the angle of orientation, $\alpha$, and the normal distance, $L$, are calculated in the following manner.

The orientation angle is calculated using the center of gravity $G = (G_x, G_y)$ of the data within the detection circle (refer to Figure 3). The coordinates of the center of gravity are calculated using

$$G_x = \frac{\sum_x \sum_y x\mathcal{H}(x,y)\omega(x-d_x, y-d_y)}{M_1} \quad (7)$$

$$G_y = \frac{\sum_x \sum_y y\mathcal{H}(x,y)\omega(x-d_x, y-d_y)}{M_1}. \quad (8)$$

The summations in (7) and (8) are performed for all pixels $(x, y)$ within the detection circle. The function $\omega(x-d_x, y-d_y)$, where $(d_x, d_y)$ are the coordinates of the center of the detection circle, determines the weighting value for $(x, y)$.

Given the coordinates of the center of gravity of the detection circle $(G_x, G_y)$, the angle $\alpha$ can be calculated using either of the following:

$$\sin(\alpha) = \frac{G_y}{\sqrt{G_x^2 + G_y^2}} \quad (9)$$

$$\cos(\alpha) = \frac{G_x}{\sqrt{G_x^2 + G_y^2}}. \quad (10)$$

Referring to Figure 3, the normal distance $L$ is found by calculating the area enclosed between the edge line and the

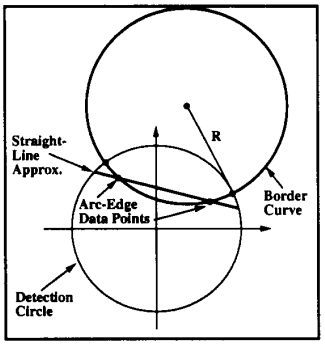


Figure 4: Arc-edge data points.

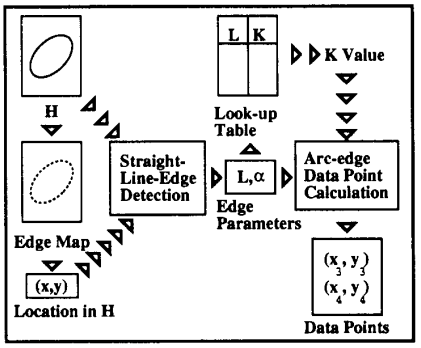


Figure 5: Arc-edge data point calculation.

detection circle, i.e., the area of region $A_2$. The area of this region is

$$a_2 = p_2 \pi r^2 = \int_{A_2} dxdy. \quad (11)$$

The left side of (11) is the area of $A_2$ written in terms of $p_2$ and the radius of the detection circle $r$. Using $a_2$, (11) can be solved to yield one of the following solutions for $L$:

$$r^2 \arcsin\left(\frac{\sqrt{r^2-L^2}}{r}\right) - L\sqrt{r^2-L^2} - a_2 = 0 \text{ if } p_1 \geq p_2 \quad (12)$$

$$0.5\pi r^2 - L\sqrt{r^2-L^2} - r^2 \arcsin\left(\frac{L}{r}\right) - a_2 = 0 \text{ otherwise.} \quad (13)$$

We use either (9), or (10), to calculate $\alpha$, then calculate $L$ with (12), or (13). In the next section, we will show how to use $L$ and $\alpha$ to calculate the coordinates of data points that lie on the circular border curve. These data points will be used for ellipse parameter estimation in Section 4.

### 3.3 Arc-edge Data Point Calculation

In the previous section the circular border curve was approximated using line segments that are defined by the parameters $\alpha$ 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 Figure 4, arc-edge data points are the intersection points of the straight-line approximation of the border curve and the border curve itself.

In [14], Tchoukanov and Safaee Rad show that there is a geometric relationship between the locations of the arc-edge data points and the parameters of the approximating line. This relationship is based on the assumption that the position of the arc-edge data points is a weak function of the radius of the border curve $R$, 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 in [14].

Let $K$ represent the relative position of an arc-edge data point $(x_1, y_1)$; then this assumption can be written as

$$K = \frac{x_1'}{\sqrt{r^2-L^2}}. \quad (14)$$

Evidence for the validity of (14) is provided in [14]. Off line, we create a look-up table of averaged $K$ values using (14). This table contains average $K$ values for $R$ varying in the range $(4.5 \ldots 300)$ pixel units, and is indexed on values of $L$, for $L$ values in the range $(-4.5 \ldots 4.5)$ pixel units.

The coordinates of the arc-edge data points $(x_1, y_1)$ and $(x_2, y_2)$ are calculated with

$$x_1 = L\cos(\alpha) + K\sin(\alpha)\sqrt{r^2-L^2} \quad (15)$$

$$y_1 = L\sin(\alpha) - K\cos(\alpha)\sqrt{r^2-L^2} \quad (16)$$

$$x_2 = L\cos(\alpha) - K\sin(\alpha)\sqrt{r^2-L^2} \quad (17)$$

$$y_2 = L\sin(\alpha) + K\cos(\alpha)\sqrt{r^2-L^2} \quad (18)$$

Refer to [14] for the derivation of equations (15), (16), (17), and (18).

The complete subpixel arc-edge detector is outlined in Figure 5. 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 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 $\alpha$. Now we have values of $L$, $\alpha$, and $K$ for each straight-line edge approximating the circular border curve. The coordinates of the arc-edge data points are calculated with (15), (16), (17), and (18).

## 4 Ellipse Parameter Estimation

Ellipse parameter estimation is performed using the list of data points produced by the arc-edge detector. Let $\mathcal{P} = \{P_1 \ldots P_n\}$ represent the list of $n$ data points, where each data point $P_i$ is a subpixel value of the form $(X, Y)$. 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 $\Theta$ of the ellipse that fits the data points.

Tsuji and Matsumoto use a modified hough transformation to locate ellipses [17]. Takiyama and Ono describe another method of ellipse parameter estimation algorithm in [13]. This algorithm represents an ellipse as a quadratic equation with constraints, and finds ellipse parameters using an iter-

ative process. Another iterative method is proposed by Nagata, Tamura, and Ishibashi [9]. This algorithm uses a recursive least-squares operator to find ellipse parameters. In contrast to these iterative methods, Safaee-Rad et al. propose a non-iterative parameter estimation algorithm in [11]. This algorithm is based on a new error function that minimizes the difference between the areas of an ideal ellipse and an estimated ellipse. This new error function allows accurate parameter estimation in two passes.

We use the area-based parameter estimation algorithm described by Safaee-Rad et al. [11]. 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. The weighted data points are used to find the parameters of the ellipse. This method is non-iterative and produces results that compare favorably with iterative techniques [11].

## 4.1 The Error Function

Parameter estimation is accomplished by minimizing an error function. The characteristics of the error function affect the accuracy of the parameter estimation process. The error functions discussed here are derived from the general equation of an ellipse:

$$\mathcal{W}(X,Y) = aX^2 + bXY + cY^2 + dX + eY + f = 0. \quad (19)$$

Equation (19) is valid for all points $(X,Y)$ on ellipse contour. $\mathcal{W}(X,Y)$ is negative for points in the interior of the ellipse, and positive for points outside of the ellipse. Derivations for the equations presented in this section are provided in [10].

One method of fitting an elliptical shape to a list of data points would be to use the minimum-squares error criterion which minimizes the following error function:

$$\mathcal{J}_0 = \sum_{i=1}^{n} [\mathcal{W}(X_i, Y_i)]^2. \quad (20)$$

The goal is to find the set of parameters $\{a \ldots f\}$ that minimizes $\mathcal{J}_0$. Bookstein investigated the behavior of $\mathcal{J}_0$ for the general case of (19), and has proven the following relationship [1]. If two data points are equidistant from the optimal ellipse, with one lying along the ellipse's major axis and the other along its minor axis, then the contribution of the data point lying along the minor axis will be greater. Therefore, to perform accurate parameter estimation, the contributions of the data points must be normalized.

To calculate ellipse parameters in a non-iterative process, (19) can be normalized with respect to the constraint $f$, i.e. $f = 1$. Under this constraint, the general equation for an ellipse becomes:

$$\mathcal{W}(X,Y) = aX^2 + bXY + cY^2 + dX + eY + 1 = 0. \quad (21)$$

The corresponding minimum squares error function is:

$$\mathcal{J}_1 = \sum_{i=1}^{n} [\mathcal{W}(X_i, Y_i)]^2_{f=1}. \quad (22)$$

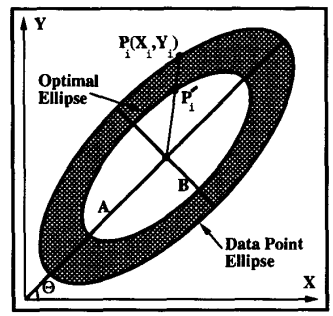


Figure 6: Area difference between concentric ellipses.

Minimizing (22) produces a set of linear equations that can be solved in a non-iterative fashion; however, the contributions of the data points are not normalized.

To compute ellipse parameters more efficiently, Safaee Rad et. al. introduce an error function, $\mathcal{J}_2$, that minimizes the difference of areas of two concentric ellipses, and normalizes the contributions of the data points [11]. Let $\mathcal{P} = \{P_1 \ldots P_n\}$ be the set of data points, $(\mathcal{A}, \mathcal{B}, \theta, X_0, Y_0)$ be the optimal parameters of the ellipse that 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 $(X_i, Y_i)$. These ellipses will be referred to as the optimal ellipse and the data point ellipse, respectively. The two ellipses are concentric and have the same eccentricity and orientation (see Figure 6). 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:

$$e_i = D - D'. \quad (23)$$

To calculate the error (23) we will 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 $v_i$ be the distance from the center of the ellipse to the point $P_i$ and $v_i'$ 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 ellispes can be derived from (23)

$$e_i = \pi(\mathcal{AB}) \left(1 - \frac{{v_i'}^2}{{v_i}^2}\right). \quad (24)$$

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. If we define $\delta_i = v_i' - v_i$, then the weighting factor for a data point $P_i$ can be expressed as

$$\psi_i = \left(\frac{v_i}{\mathcal{A}}\right) \left(\frac{1 + \delta_i/2\mathcal{A}}{1 + \delta_i/2v_i}\right). \quad (25)$$

Because $\delta_i$ is usually much smaller than $d_i$ or $\mathcal{A}$, (25) can be approximated by:

$$\psi_i \approx \left(\frac{v_i}{\mathcal{A}}\right) \left(1 + \frac{\delta_i}{2\mathcal{A}} - \frac{\delta_i}{2v_i}\right) \approx \frac{v_i}{\mathcal{A}}. \quad (26)$$

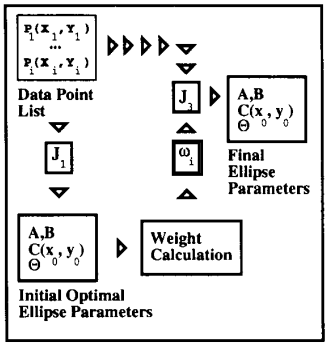


Figure 7: Ellipse parameter estimation algorithm.

Using the weights described by (26), the new error function is defined as:

$$\mathcal{J}_2 = \sum_{i=1}^{n} \left[\psi_i \frac{1}{\pi \mathcal{AB}}(D - D_i')\right]^2 = \sum_{i=1}^{n} [\psi_i \mathcal{W}(X_i, Y_i)]^2_{f=1}. \quad (27)$$

Error function $\mathcal{J}_2$ minimizes the area-based error function described (23) and normalizes the contributions of the data points. We will use this error function to calculate the final values of the ellipse parameters.

## 4.2 Calculating Ellipse Parameters

The error function described by (27) minimize the area based error criteria (23) and normalize the contributions of the data points. Equations for the ellipse parameters can be derived by taking the first derivatives of $\mathcal{J}_2$ with respect to the five unknowns $(a, b, c, d, e)$ to yield a system of five linear equations with five unknowns. The solution of this system of equations is the vector $V^T = (a, b, c, d, e)$. The five parameters of an ellipse can then be calculated using the following:

$$X_0 = \frac{2cd - be}{b^2 - 4ac} \quad (28)$$

$$Y_0 = \frac{2ae - bd}{b^2 - 4ac} \quad (29)$$

$$\theta = \arctan\left[\frac{(c-a) + \sqrt{(c-a)^2 + b^2}}{b}\right] \quad (30)$$

$$\mathcal{A}^2 = \left[\frac{2(1 - F_s)}{b^2 - 4ac}\right][(c+a) + \sqrt{(c-a)^2 + b^2}] \quad (31)$$

$$\mathcal{B}^2 = \left[\frac{2(1 - F_s)}{b^2 - 4ac}\right][(c+a) - \sqrt{(c-a)^2 + b^2}], \quad (32)$$

where

$$F_s = \frac{bde - ae^2 - cd^2}{b^2 - 4ac}. \quad (33)$$

Both $\mathcal{J}_1$ and $\mathcal{J}_2$ are used in estimating the ellipse parameters. As shown in figure 7, $\mathcal{J}_1$ is used to generate the parameters of an initial optimal ellipse. These parameters are used to estimate the value of $\delta_i$ and $d_i$ for each data point. The data point weighting values $\omega_i$ are calculated using $\delta_i$ and $d_i$. The weighted data points are used in $\mathcal{J}_2$ to find the parameters of the final optimal ellipse. This method of parameter estimation is non-iterative and produces good results.

Table 1: Results for a 1/8-in diameter circle.

| Estimated Ellipse Parameters: Circle Radius 0.0625 in | | | | | |
|---|---|---|---|---|---|
| Method | $X_0$ | $Y_0$ | $\mathcal{A}$ (in) | $\mathcal{B}$ (in) | $\Theta$ |
| Simple | -30.615 | -5.680 | 0.079 | 0.077 | 81.610° |
| Canny 1 | -30.974 | -5.437 | 0.075 | 0.078 | 77.449° |
| Canny 2 | -30.974 | -5.463 | 0.075 | 0.077 | 82.069° |
| Arc Edge | -30.866 | -5.479 | 0.050 | 0.047 | 79.381° |
| High Res. | -30.750 | -5.528 | 0.064 | 0.063 | 83.800° |

# 5 Experiments

In the previous chapters we have described a method of parameter estimation for circular shapes. The three algorithms used in this method, superresolution, arc-edge data point detection, and ellipse parameter estimation, were implemented in C++. These programs were compiled on Sun Sparc workstations using the Free Software Foundation's Gnu C++ compiler.

Our method of ellipse parameter estimation was tested on real image sequences. Each image was produced in the following manner. A picture of a circle was produced using postscript and printed using a high-resolution, i.e., 600 dots per inch, Hewlett Packard Laser Jet 4 SI printer. The picture of the circle was digitized using a Sony model XC-77 video camera and Datacube MaxVideo 20 image processing equipment. Each input image has dimensions $512 \times 480$ pixels, and pixel intensities are in the range $(0 \ldots 255)$. Image sequences of circles with diameters of one-eighth inch, one-fourth inch, one-half inch, one inch, and two inches were used to test our inspection system. The accuracy of our parameter estimation method was measured using a camera calibration method described by Tsai in [15].

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, i.e., (1024 × 960) 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, i.e., (512 × 480), 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 Chapter 3 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 4.

Results for a one-eighth inch diameter circle are given in Table 1. The coordinates of the ellipse center point, $(X_0, Y_0)$ are with respect to the center of a low-resolution image and

Table 2: Average percent error.

| Average Percent Error of Circle Radius Estimates. | | | | | |
|---|---|---|---|---|---|
| | Circle Diameter in Inches. | | | | |
| Method | 0.125 | 0.25 | 0.5 | 1 | 2 |
| Simple | 23.431 | 10.316 | 5.767 | 2.669 | 1.517 |
| Canny 1 | 22.348 | 11.007 | 5.926 | 2.973 | 1.443 |
| Canny 2 | 20.936 | 10.591 | 5.876 | 2.912 | 1.430 |
| Arc Edge | 22.400 | 11.533 | 5.149 | 2.518 | 1.605 |
| High Res. | 1.566 | 0.853 | 1.041 | 1.756 | 1.482 |

are given in pixel units. The axis lengths are given in inches, and the orientation angle is measured in degrees.

Table 2 lists the average percent error of the axis length estimates. The percent error, $\Delta$, is calculated accordingly:

$$\Delta = \frac{(|\mathcal{A} - \text{Actual } \mathcal{A}| + |\mathcal{B} - \text{Actual } \mathcal{B}|)}{2 \times \text{Actual Radius}} \times 100$$

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 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.

## 6 Conclusions

We have presented a new method of parameter estimation for circular shapes that uses image sequences. In this method, an image sequence of the circle being inspected is used to create a 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. This parameter estimation algorithm finds the center point of the ellipse, the major axis length, the minor axis length, and the ellipse's angle of orientation with respect to the horizontal axis of a local reference frame in the image. 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.

Our method was tested on real image sequences and produced good results. Experiments have shown that our method produces the most improvement in estimation accuracy when we are inspecting small circles. One motivation for our research was the task of inspecting small features, such as via holes in printed circuit boards. An automated optical inspection system for performing such tasks would have to inspect as many circles as possible, while maintaining a high degree of accuracy. The parameter estimation algorithm described in this work is ideally suited to such a system because it provides the greatest improvement in accuracy when inspecting small features. For large circles, subpixel accuracy and superresolution do not provide much improvement over simpler methods.

## References


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

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

[3] S. Ghosal and R. Mehrotra. Orthogonal moment operators for subpixel edge detection. *Pattern Recognition*, 26(2):295–306, February 1993.

[4] R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision, Vols. I and II.* Addison-Wesley Pub. Co., Inc., Reading, MA, 1993.

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

[6] M. Irani and S. Peleg. Image sequence enhancement using multiple motions analysis. *IEEE*, pages 216–221, March 1992.

[7] G. Jacquemod, C. Odet, and R.Goutte. Image resolution enhancement using subpixel camera displacement. *Signal Processing*, 26(1):139–146, January 1992.

[8] D. Keren, S. Peleg, and R. Brada. Image sequence enhancement using sub-pixel displacements. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 742–746, 1988.

[9] T. Nagata, H. Tamura, and K. Ishibashi. Detection of an ellipse by use of a recursive least-squares estimator. *Journal of Robotic Systems*, 2:163–177, February 1985.

[10] J. M. Reed. Subpixel parameter estimation for circular shapes using image sequences. Master's thesis, University of Illinois at Urbana-Champaign, Aug 1994.

[11] R. Safaee-Rad, I. Tchoukanov, B. Benhabib, and K.C. Smith. Accurate parameter estimation of quadratic curves from grey-level images. *CVGIP: Image Understanding*, 54(2):259–274, September 1991.

[12] A.J. Tabatabai and O. R. Mitchell. Edge location to subpixel values in digital imagery. *PAMI*, 6(2):188–200, March 1984.

[13] R. Takiyama and N. Ono. An iterative procedure to fit an ellipse to a set of points. *IEICE Transactions on Communications Electronics Information and Systems*, 74(10):3041–3045, October 1991.

[14] I. Tchoukanov, R. Safaee-Rad, B. Benhabib, and K.C. Smith. Sub-pixel edge detection for accurate estimation of elliptical shape parameters. *CSME Mechanical Engineering Forum 1990 (Univ. of Toronto, 1990), Proceedings*, pages 313–318, 1990.

[15] R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf television cameras and lenses. *IEEE Journal of Robotics and Automation*, RA-3(4):323–344, Aug 1987.

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

[17] S. Tsuji and F. Matsumoto. Detection of ellipses by a modified Hough tranformation. *IEEE Transactions on Computers*, 27(8):777–781, August 1978.