Machine Vision and Applications (1992) 5:249–263

Machine Vision
and Applications

A Method for Detection of Circular Arcs Based on the Hough Transform

Pär Kierkegaard
Department of Mechanical Engineering, Division of Assembly Technology, University of Linköping, Linköping, Sweden

Abstract: The circular arc is a very useful feature for object detection and recognition in industrial environments. In this paper, a method for detection of circular arcs is described that is based on the Hough transform. The method estimates all five arc parameters and is robust in the presence of a moderate amount of noise. It has a computational and memory complexity of $O(n \cdot m \cdot R)$ and $O(n \cdot m)$ respectively, where n and m are the sizes in the x and y directions and R is the maximum expected arc radius in pixels. Arcs as small as 45 degrees and radii down to 4 pixels can be detected. The computing time is almost independent of the number of circular arcs in the image.

Key Words: Hough transform, feature extraction, circular arcs

1 Introduction

As the development of robot assembly lines is heading towards more flexibility and shorter cycle times, the need for robots and robot vision systems increases. To make robot vision systems easy to use, it is necessary to provide them with a constrained but still general procedure for recognition of predefined objects. This has been found to be a nontrivial task.

A common way to perform object recognition is to detect features that are significant for the object and then use the parameters of the detected features to identify the objects and estimate their position and orientation. Since many industrial objects have outlines which consist of straight lines and circular arcs, the detection of these features is particularly important. Line detection is well known and commonly performed with the Hough transform, but the detection of circles and circular arcs is somewhat more complicated and less obvious (Davies 1985, 1988; Kimme et al. 1975; Qin-Zhong 1989).

The importance of circular arc detection can be understood from the fact that one single arc often defines the position and orientation of the whole object. When partial occlusion occurs, it will in general not be possible to estimate all the object parameters but the identification of circular arcs may still produce very useful information.

1.1 Hough Transforms

The Hough transform is the starting point for this paper. In its original form, it was only good for detecting lines (Hough 1962).

When Duda and Hart rediscovered the Hough transform (Duda and Hart 1975) and also extended the transform to yield the parameterized line description and the circle transform (to be described below), they adopted the name Hough transform also for the new transform types, and similar histogramming methods have been called Hough transform methods ever since. A good survey on Hough transforms is Illingworth and Kittler (1988).

Since 1972, many authors have published transform methods based on accumulator arrays and histogramming. All objects that can be described mathematically with a limited number of parameters can also be transformed to a transform-space with the same number of dimensions as the number of parameters describing the object.

1.2 Circle Detection

The Hough method was extended to *circle detection* by Duda and Hart (Duda and Hart 1975). They

Address reprint requests to: Pär Kierkegaard, Department of Mechanical Engineering, Linköping Institute of Technology, S-58183, Linköping, Sweden.

This work was supported by the Swedish Board for Technical Development, Grant No. 87-01954P.

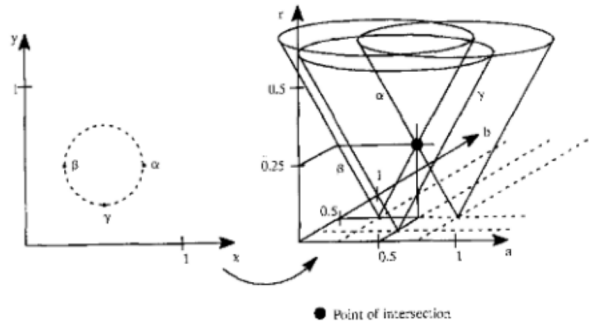

Figure 1. The circle transform. Three points generate one cone each and the intersection of the cones indicate the circle parameter values.

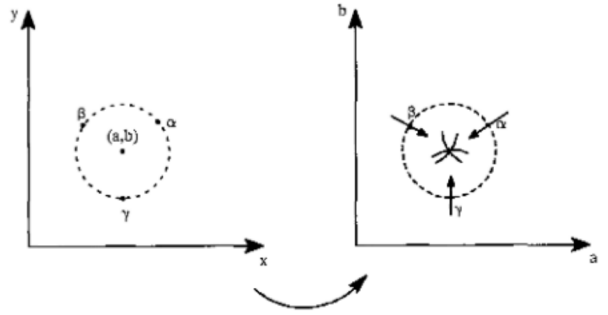

Figure 2. The circle transform with use of the gradient. Radius known.

showed a way to detect circles in the same way by using a three-dimensional transform space and incrementing values on a cone in the 3–D space for each point in the image. All cones generated by the same circle periphery points will intersect in the transform space and the circle parameter values are found by detecting the maxima in the transform space. The location of the maxima indicates the parameter values of the circle which is illustrated in Figure 1. The method is of course extremely computation intensive and has a memory requirement of $O(m{\cdot}n{\cdot}R)$ (image size = $m{\cdot}n$, number of possible radii = R).

If the radius, R, is known a priori, one dimension of the accumulator array can be omitted using only a slice of the accumulator array at $r = R$. A further reduction of the computational burden is obtained by using the gradients (cf. Ballard and Brown's General Hough Transform, (Ballard and Brown 1981)). If the radius is known, a point on a circular arc and the gradient uniquely define a single point as the circle center (a,b) in the parameter space. Using many points on the arc, a cluster will appear at the location of the circle center point. This is described in Kimme et al. (1975) and is shown in Figure 2. In reality,

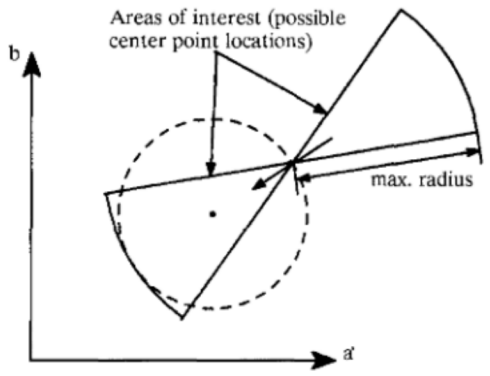

Figure 3. Limitations of the center point location. Radius unknown.

the gradient direction is always limited in precision. Therefore, points on a small arc in the neighborhood of the circle center are incremented.

Thus, if the radius is known a priori, the algorithm has a memory requirement of $O(m{\cdot}n)$ and runs in $O(m{\cdot}n)$ time, since only a few points have to be updated for each image point. To obtain the $O(m{\cdot}n)$ memory requirement and a reasonable time complexity with an unknown radius is more difficult.

A gradient based method for circle detection is described by Davies (1988). If the radius is unknown, we know that the circle center point lies in a circle sector (Figure 3). Only the accumulator cells in this sector should be updated. The angular size of the sector is dependent of the uncertainty of the gradient operator and radius corresponds to the largest radius expected.

In the general case the gradient detector does not give the curvature. Therefore if both dark and light circular objects can occur, the circle center may appear in both sectors shown in Figure 3, otherwise one of them can be pruned.

Suppose now that a better orientation estimate is obtainable due to a better gradient estimator using larger or more kernels. The better the estimate, the smaller the sector and in the extreme case the sector will become a line for small radii. This forms the basis for efficient circle detection, even if the radius is unknown. If values are incremented along a line or a thin sector in the transform space as shown in Figure 4, there will be a cluster at the location for the circle center. It is easy to understand that the possibility of the line passing through the exact center of the circle is heavily dependent on the accuracy of the gradient estimate and the distance between the detected edge point (edgel) and the proposed center point. The likelihood that the line should pass through the true center point is dependent on the

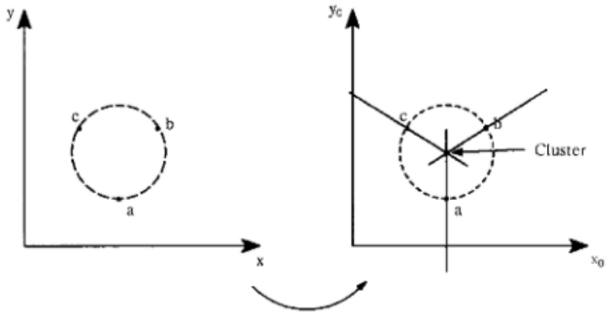

Figure 4. The circle transform with use of the gradient and accumulation along a line. Radius unknown.

radius, and the number of lines generated is equal to the number of edgels of the circle (or circular arc), and is therefore also dependent on the circle radius. Thus, the larger the circle radius, the more lines are generated, and the more prominent the accumulated cluster in the accumulator space. In fact, since the number of lines generated is proportional to the radius, and the probability of the line passing through the circle center point is inversely proportional to the circle radius, the actual number of lines passing through the center point will not vary with the radius at all. Section 3 will verify this observation.

1.3 Detection of Circle Radius

Although the circle center is located in Figure 4, the radius is still not known explicitly. Ideally, there is a relation between the radius and the amplitude of the maximum detected, (amplitude $= 2 \cdot \pi \cdot$radius), but since we don't know whether a complete circle or a circular arc has been detected and the amplitude is dependent on the amount of noise in the image, this amplitude information in the transform space is of little use.

A method for radius estimation is mentioned briefly by Illingworth and Kittler (1987). It is based on histogramming the distance from each edge point to the estimated center point candidate ($r^2 = (x - a_0)^2 + (y - b_0)^2$), and finding the maximum in the histogram of r-values. This method is adopted and evaluated in this paper. It will be fully described in Section 2.5–2.6. A comparative study of the method and others can be found in Yuen et al. (1990).

In what follows a Hough-based method is described that efficiently detects circular arcs and calculates all five parameters (x-center, y-center, radius, orientation, and angular size) to be used in object detection. The method uses the circle detection described in Davies (1988) to estimate circle center points (the gradient orientation is used) and the histogramming method described in Illingworth and Kittler (1987) for radius estimation. Illingworth and Kittler's method is augmented to be able to detect an arbitrary number of circles present in the image at the same time, and also to estimate all five parameters describing a circular arc. The method has the following properties:

- Detects circle center points and radii
- Detects circular arc center points, radii, orientations, and angular sizes
- Detects all circles and arcs in an image simultanously the same time
- Separates concentric and excentric circles and circular arcs
- Time complexity $O(n \cdot m \cdot R)$ ($n \cdot m$ = image size, R = maximum expected radius)

2 The Arc Detection Method

2.1 Overview

The method for detection of circular arcs presented in this paper is outlined in Figure 5.

Step by step the following procedures are executed.

a) **Edge detection.** The local derivatives in the x and y directions are computed. The combined result which estimates the gradient magnitude is thresholded.
b) **Hough transform.** Circle centers are detected using the technique of Figure 4.
c) **Peak enhancement.** The image is filtered (convolved) with a large kernel of type Laplacian.
d) **Max finding.** A binarizing procedure which leaves only one pixel within a neighborhood.
e) **Histogramming I.** The points from step d) are used as candidates for circle centers. For each center, the distance (the potential **radius**) to all gradient points is histogrammed.
f), g), h) **Histogram processing.** Peaks in the histograms are detected. Center point candidates without a pronounced peak in their histogram are discarded.
i) **Histogramming II.** Each remaining center point candidate now has a radius and hereby in the gradient image also a specific set of circular arc points. For each center point the **angles** for all radii to this arc are histogrammed.
j), k) **Histogram processing.** The histograms are filtered, thresholded, and converted to angular sectors. The complete feature list of circular arcs is assembled.
l) The feature list is converted to an overlay image, or, alternatively, used for model matching.

In the following sections, the method is going to

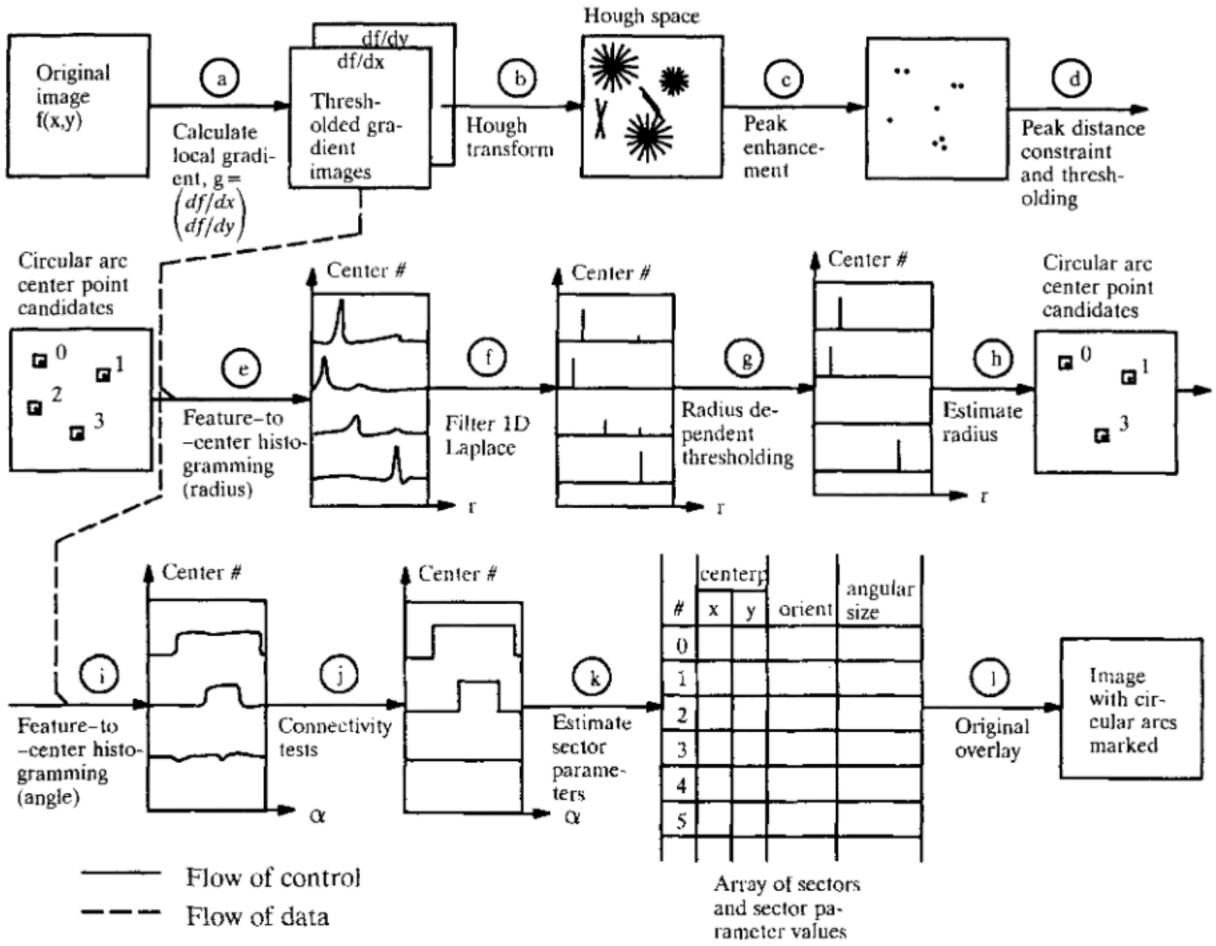

Figure 5. A schematic description of the method.

be penetrated more in detail. Figure 6 shows a test image that will serve as a running example throughout the chapter.

2.2 Edge Detection

The first operation to be performed is to calculate the local gradients in the source image; see Figure 5 step a). Since the whole method is dependent on the accuracy of the edge detection, this step is critical.

Several different edge detection operators have been proposed in the literature (Lyvers 1988). Many of these have such a complex implementation that they are not possible to use in a real-time application. In this project, an operator is needed that gives a reasonably well-defined edge orientation vector in each point of a fairly low computational cost.

The first idea was to apply a small 3·3 gradient estimation kernel, calculate the magnitude, and then apply a larger kernel if the response from the small kernel is larger than a certain threshold. This would give good accuracy in position from the small kernel and a good estimate of the orientation from the larger kernel. This turned out to be a good idea in the case of noise free images, but when noise was added to the image, the result from the small 3·3 operator became irrelevant. After some experiments, we choose a 7·7 separable gradient estimation operator (Figure 8), as a tradeoff between accuracy in orientation estimate, accuracy in position, and computational effort.

An angular error comparison study was done and the operator was found to have a maximum angular error of 1.6 degrees, compared with 4.5 degrees for the ordinary 3*3 Sobel kernel. The $\frac{d}{dx}$ and $\frac{d}{dy}$ kernels can be decomposed as shown in Figure 8 (Danielson and Seger 1989). To compute f_x and f_y in one output position takes 14 additions and two subtractions, all of them employing nearest neighbors.

Since the gradient response in homogeneous areas is small and does not give significant information about object structure, the gradient is only saved if it is prominent. Let f_x, f_y be the kernel

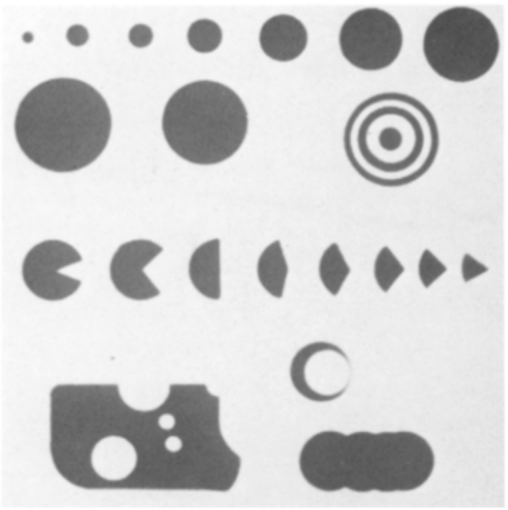

Figure 6. The 256 · 256 test image.

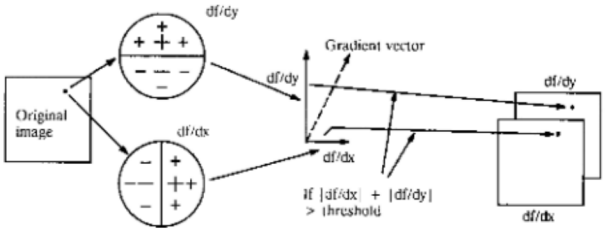

Figure 7. Gradient detection and thresholding.

responses. Then the thresholded gradient, $\vec{g}$, should be computed as

$$\vec{g}(x, y) = \{\sqrt{(f_x(x, y))^2 + (f_y((x, y))^2} > T\} \begin{pmatrix} f_x(x, y) \\ f_y(x, y) \end{pmatrix}$$

where the expression {. . .} evaluates to 0 (false) or 1 (true). To save time one can use a simpler approximate gradient strength calculation:

$$\vec{g}(x, y) = \{|f_x(x, y)| + |f_y(x, y)| > T\} \begin{pmatrix} f_x(x, y) \\ f_y(x, y) \end{pmatrix}$$

The thresholded image is not thinned to unit pixel width because this would lead to decreased signal-to-noise ratio in later stages (this will be clear in Section 2.3). The gradient estimation procedure is described in Figure 7.

The result of the edge detection applied to the test image is shown in Figure 9. Here the magnitude is shown, instead of the two gradient images.

2.3 The First Transform

Circle center points are detected in the way described by Davies (1988). The method works as follows (step b) in Figure 5);

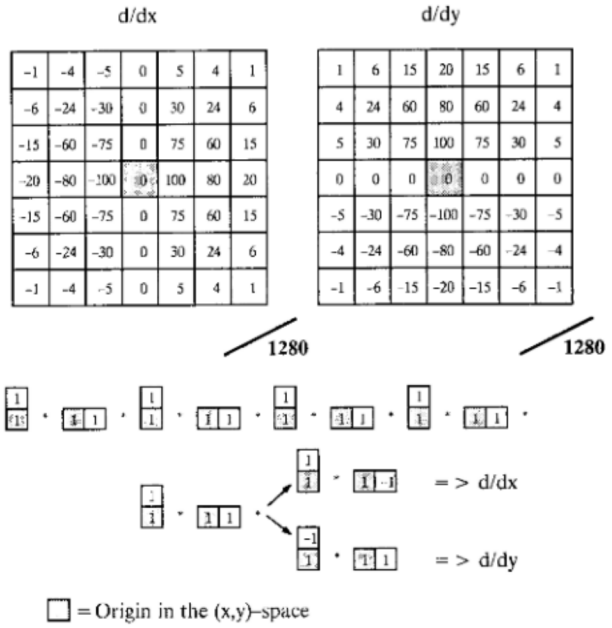

Figure 8. The two gradient estimation kernels and their decomposition.

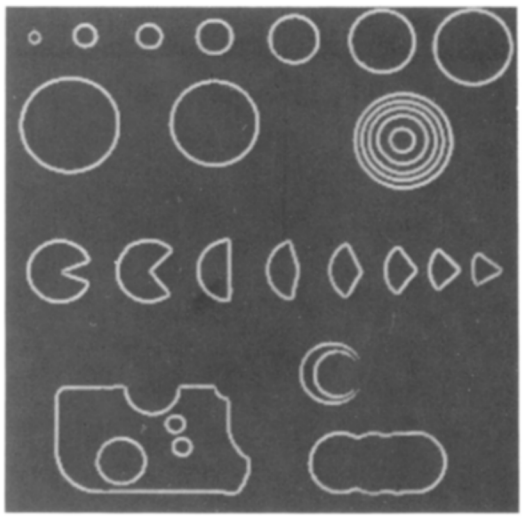

Figure 9. The test image filtered and thresholded (gradient magnitude shown).

For each edgel (points that have valid gradients according to Section 2.2) we use the gradient to increment values along a line perpendicular to the edge (in the gradient direction) in the transform space (Figure 10). The line is drawn at both sides of the edge point to detect both dark and light disks and disk sectors. When all points have been transformed, the transform space contains clusters, representing possible locations of circle center points. Note that if the gradient image had been thinned in

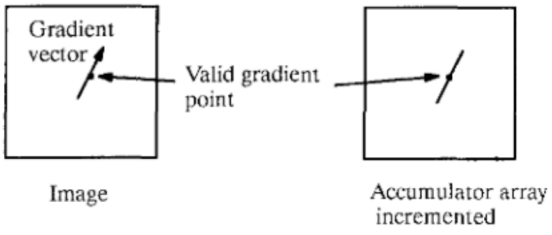

Figure 10. The incrementation caused by a valid gradient point.

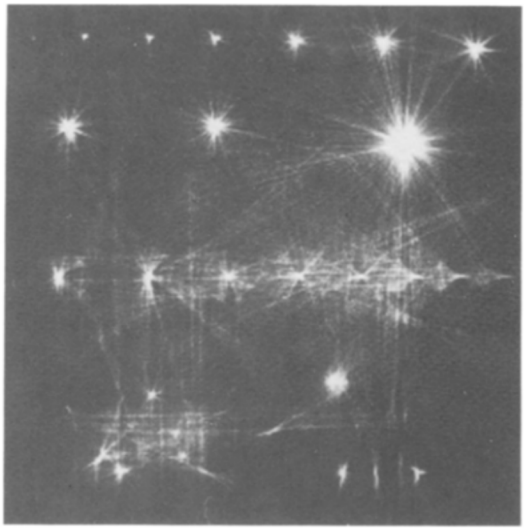

Figure 11. The Hough transformed test image.

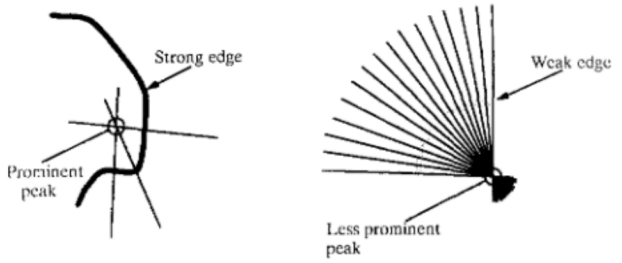

Figure 12. Two equally strong peaks generated of which one is false and caused by differentiated increments in the transform space.

		5	16	22	16	5		
	8	32	56	64	56	32	8	
5	32	56	32	6	32	56	32	5
16	56	32	-120	-224	-120	32	56	16
22	64	6	-224	-376	-224	6	64	22
16	56	32	-120	-224	-120	32	56	16
5	32	56	32	6	32	56	32	5
	8	32	56	64	56	32	8	
		5	16	22	16	5		

2304

Figure 13. The 9·9 peak enhancement kernel.

step a), the peaks in the accumulator array would have been less prominent, since fewer lines would have been accumulated.

It has been proposed that the transform should increment the values with a number proportional to the gradient strength instead of unity. One could expect that this would result in a transform space with more prominent peaks. This was tried and was found not to be the case. One explanation could be the following.

If the incrementation is gradient-magnitude dependent, a prominent edge point would result in a more prominent line in the transform space than a weak edge point. There is, however, no evidence that a prominent edge has a higher possibility to belong to an arc than a weaker one. This means that if a prominent, noncircular curve is present, the detected maxima might belong to crossings between lines caused by the curve rather than by lines generated from a weaker circular arc. To clarify this, see Figure 12.

2.4 Maxima Detection

As can be seen in Figure 11, the Hough space may contain many peaks, representing center points of the circles and arcs. To detect these peaks, the transform space is first filtered with a 9·9 Laplacian-like peak detecting filter (step c) in Figure 5). The filter enhances existing prominent peaks and suppresses small fluctuations. The kernel used is shown in Figure 13 and is borrowed from Danielsson and Seger (1990). This kernel is separable and seems to work very well. Further optimization of this filter is quite possible but hardly worthwhile. The most important thing is that no significant peaks should be omitted. False peaks are a less severe problem since they will most likely be eliminated in later steps. Figure 14 shows the result after peak enhancement for our running test image.

When comparing Figure 14 with the original image (Figure 6) it is easy to correlate the peaks in Figure 14 to the center points of the circles and circular arcs in Figure 6. To make the computer do this, the image in Figure 14 should be reduced to

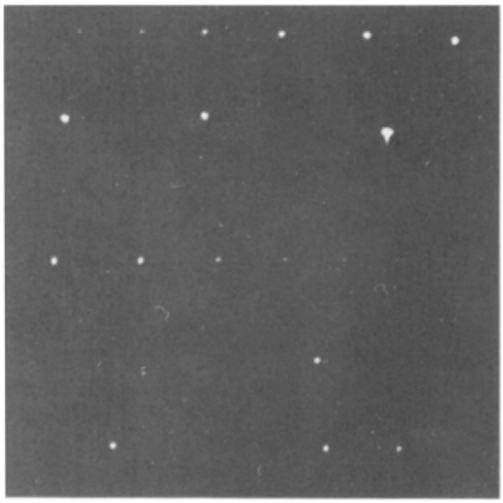

Figure 14. The Hough transformed test image with the enhanced peaks.

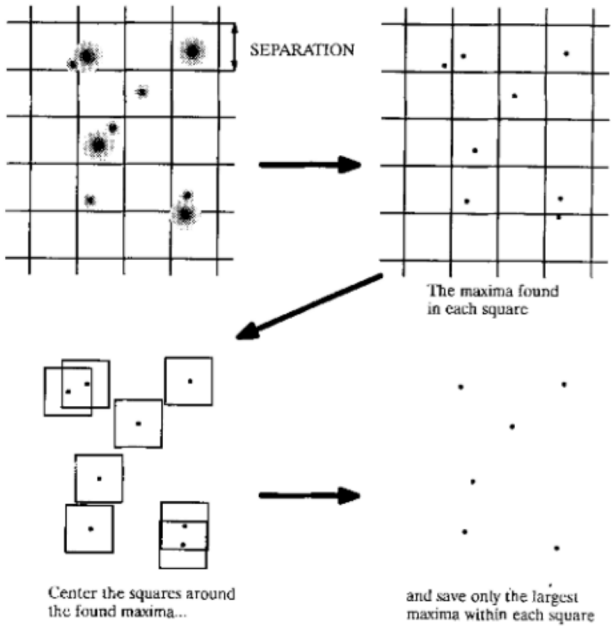

Figure 15. The maxima detection principle.

contain only single pixel peaks (step d) in Figure 5). This task is performed in two steps to save time. In the first step the peak image is divided into small squares, each having a size equal to SEPARATION·SEPARATION, where SEPARATION is the minimum expected distance between any two circle centers. Then the maximum pixel value is found within all squares and the rest of the pixels are set to zero. If the maximum pixel value is below a certain threshold, even this pixel is set equal to zero. In the next step, the neighborhood of the remaining pixels is checked (the square is moved to put the pixel in the center of the square). If the pixel is the largest within its neighborhood, it is considered to be a potential circular arc center point candidate (see Figure 15). This method has proven to be fast and reliable in the sense that very few maxima are missed, even though some false maxima are detected due to noise. This is very difficult to avoid, but will cause no problem as we shall see later on. The center points found in the test image are shown in Figure 16.

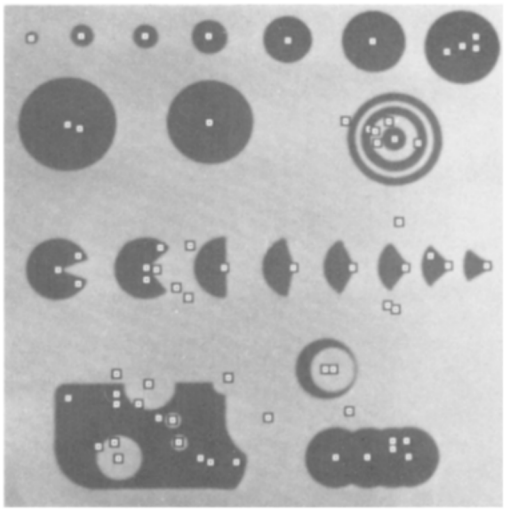

Figure 16. Found center points marked in the test image.

2.5 Radius Histogramming

Once the center points have been detected, the circle and arc radii are estimated. This is performed in the histogramming step e) in Figure 5. For each valid gradient point (edgel) we do the following (Figure 17):

1. Assign a histogram array to each detected center point candidate with R number of entries (R = maximum circle radius).
2. Find all center point candidates within a square of size $(2 \cdot \text{MAXRADIUS})^2$ and the current pixel in the center.
3. For each such center point compute the angle ϕ between the vector $\vec{C}$, pointing from the pixel towards the center point and the gradient vector, $\vec{G}$. The angle ϕ is computed as:

$$\phi = \arccos\left(\frac{\vec{C} * \vec{G}}{|\vec{C}| * |\vec{G}|}\right)$$

4. If $\phi < 0.1$ rad, the center point and the feature point is considered to belong to the same circle.

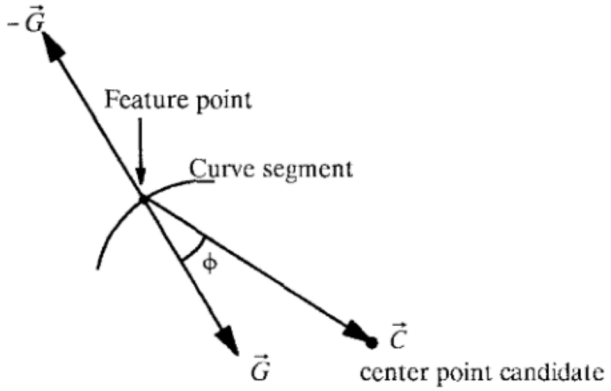

Figure 17. A found feature point, its gradient vectors, and its relation to a neighboring center point candidate.

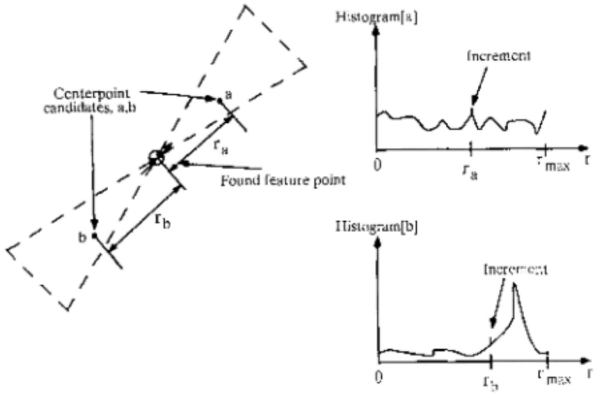

Figure 18. A feature point and two possible center points are found. The distances between the feature point and the center points are calculated and the histograms are incremented. The searched area is shown within dotted lines.

The threshold 0.1 radians has been found to work well in the present application. If the angle satisfies the condition, the histogram is incremented by one at the entry equal to $|\vec{C}|$ (Figure 18). This distance is equal to the radius of the circle if the pixel belongs to the circle periphery.

2.6 *r*-Histograms

When all valid pixels have given their contributions to the histogram arrays, we need to detect peaks in these arrays. A peak indicates that many edgels are situated at the same distance from the center point. This distance is indicated by the index in the array at which the peak occurs. To detect the peaks, a method similar to the approach described in Section 2.5 is used. First a Laplacian-type, one-dimensional filter is applied to the array. This filter exists in three

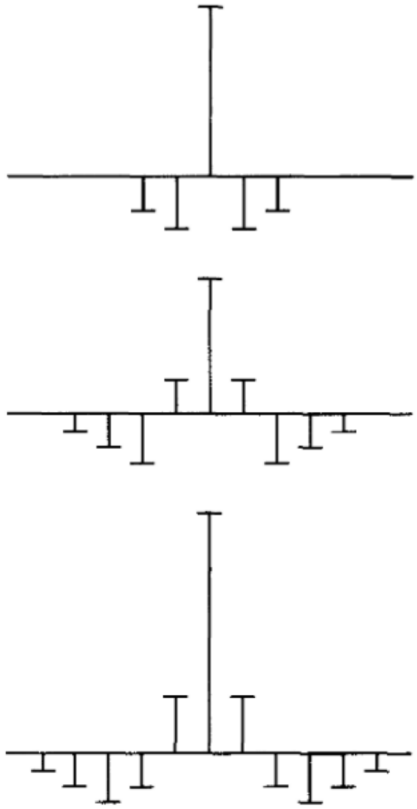

Figure 19. Filters used for peak detection in the radial histograms. Three different filter kernels are used for different radii.

different versions (Figure 19). Since large circles give rise to more widespread peaks in the histogram, they are more reliably detected with larger kernels.

If in the filtered histogram an element is found to be larger than its four closest neighbors, it is considered to represent a valid radius and thus a valid circle or circular arc. Note that a peak in the image that is caused by a lot of spurious line and arc segments as in Figure 20a does not give a peak in the *r*-histogram and will be eliminated in this step. Hence most of those maxima that are erroneously detected in Section 2.4 will disappear. The result from this step is an array of records describing possible circular arcs and circles with circle center coordinates and radii.

2.7 Angle Histogramming

To completely define a circular arc, we need another two parameters, shown in Figure 21. These parameters are:

ϕ = arc orientation = orientation of the midpoint radius
$\Delta\phi$ = arc size

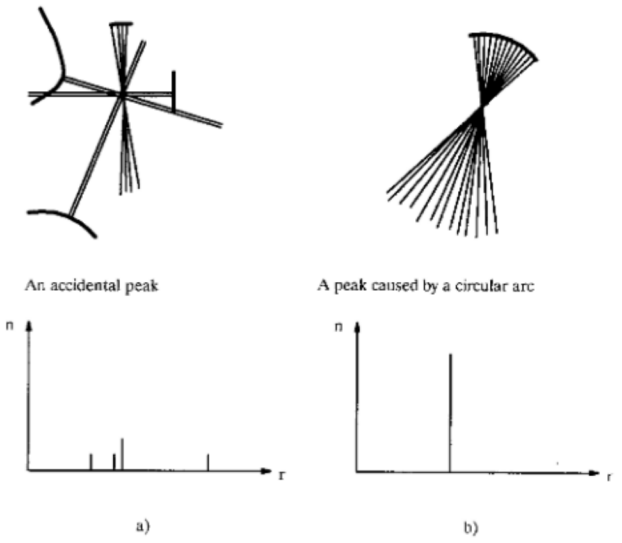

Figure 20. Histograms caused by an accidental peak and a circular arc, respectively.

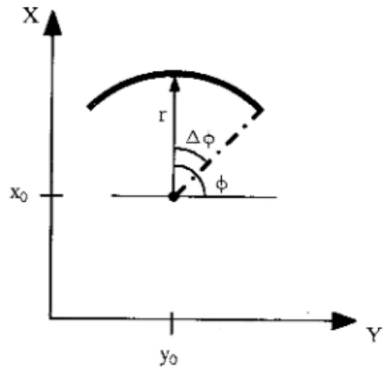

Figure 21. The definition of a circular arc.

The simplest way of estimating these parameters would be to overlay the already estimated circle data, center point and radius, and mask out the feature points along the circle periphery. For each circle we could then measure the angle for each radius to the periphery points and collect a histogram over the angles.

Another way of obtaining this histogram is to adopt an approach similar to the radius estimation method just described. This method was chosen for implementation and it consists of the following steps:

1. Establish a histogram array for each circle detected in the previous step with 360/(angle resolution) entries.
2. For each feature point, calculate the center points which lie in the correct gradient direction. For each center point, calculate the angle for the vector pointing from the feature point towards the center point and increment the histogram array for this angle.
3. When the whole image has been processed, check all histogram arrays for runs of constant level which should indicate true arcs. Note that one image peak can generate more than one arc.

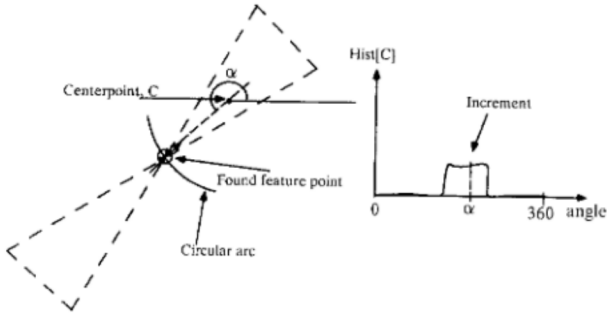

Figure 22. A feature point and a center point candidate is found. The distance between the feature point and the center point is calculated and the histogram array is incremented. The searched area is shown within dotted lines.

In Figure 22 the angle α is the angular position of the feature point in the circle whose center point candidate is indicated in the figure. This angle generates an increment in the histogram connected to the center point candidate. To introduce an increment, the feature point must satisfy two constraints. The first is that the radius should have approximately the same orientation as the gradient (same criterion as in Section 2.5), and the second is that the distance between the feature point and the center point candidate must be close to the radius connected to the center point candidate. The histogram to the right in Figure 22 is connected to the center point candidate in the left part of the figure. A feature point is found, the direction to the center point candidate (α) is calculated and the histogram is incremented at the entry equal to α.

The resulting histogram will contain a cluster for the ϕ-values where there exist relevant arc segments. For a complete circle, the cluster will cover the complete histogram and for smaller arcs, the length of the cluster can be directly interpreted as the angular size.

The information gathering in the angle histogramming overlaps and reiterates some of the steps in the radius histogramming. A possible alternative would be to save more information available during the radius histogramming. (Already here are center points connected to each feature pixel and the angle

to every such center point could be stored in a dynamic data structure. In effect, this would buy time for memory but the idea has not been implemented here.)

It should also be noted that if we can be sure that no concentric arcs are present, it would be possible to estimate the arc data already during the radius histogramming step. We could simply assign a second histogram array to each center point and accumulate the angle in this array exactly in the same way as was just described for the separate angle histogramming. This would be more noise-sensitive however, since any valid pixel having the right gradient orientation and being within the maximum radius distance from the center point would give its angular contribution to the angle histogram. Hence it depends on the application whether this approach could be applied or not.

If the radius histogramming was to be implemented in hardware (which it should in a real-time environment) it would be quite easy to use the hardware also for the angular histogramming.

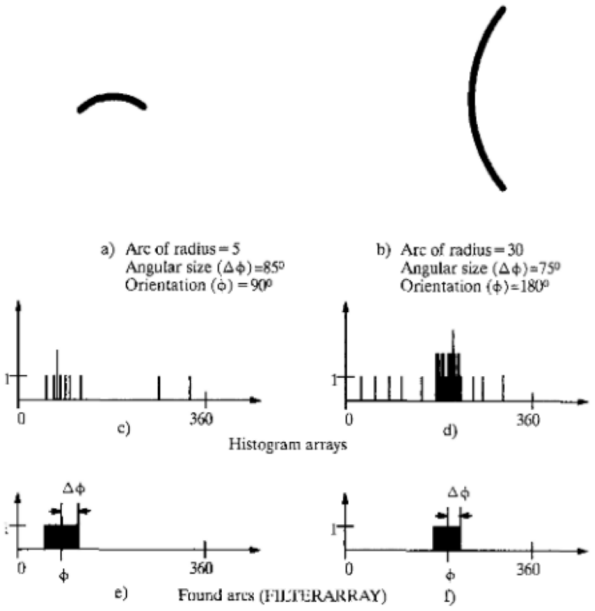

Figure 23. The distribution of two angular histograms and the selected arcs.

2.8 Detecting Arc Parameters

The two arc parameters ϕ and $\Delta\phi$ should now be derived which proves to be a nontrivial task. Some of the problems to be solved are the following.

The density of the accumulator array is radius dependent, since large circles contain more pixels per degree than small circles. Therefore, some scaling function must be applied to verify if an element in the array belongs to an arc or not. In Figure 23c the values in the histogram are quite spread out and many zeros can be found between the one's, but it is still a valid arc. In Figure 23d on the contrary, there are no gaps at all where the arc is and it is very easy to detect.

To obtain a correct parameter estimation, the circle radius r must be taken into account, and we must allow large gaps in the histograms for small circles, while smaller gaps should be permitted for larger arcs. The maximum gap allowed here is set of $25/r + 1$. Theoretically, there are $2\cdot\pi\cdot r$ edge elements (and thus $2\cdot\pi\cdot r$ increments in the histogram array) on a complete circle. In reality, some edge elements are missed due to noise, but since an edge is detected not only on the actual edge itself, but also on the close neighborhood to the edge, the actual number of line elements will be close to $2\cdot\pi\cdot r$. The angular resolution in the histogram array is chosen to 3.6°, which gives 100 entries. This results in an angular resolution equal to $2\cdot\pi\cdot r/100$ increments per entry. For $r = 5$ this gives 0.63 increments per entry and gaps will occur in the histogram array. The chosen maximum gap allowed is considerably larger than the expected maximum gap, but this has been found to be necessary, especially in images with a high noise level.

The arc detection consists of the following procedure.

1. Allocate a FILTERARRAY with the same number of entries as the angular histogram arrays.
2. For each of the histogram arrays, called CURRENT below, do the following:
   2.1 Set all the elements in FILTERARRAY to zero.
   2.2 Set a variable MAXGAP equal to 25/radius(CURRENT) + 1. This is the maximum gap tolerated between any two non-zero elements in CURRENT.
   2.3 Set GAP to zero. Set INDEX to zero.
   2.4 Scan through CURRENT and for each element, CURRENT[INDEX], do the following:
      2.4.1 If CURRENT[INDEX] > 0 set GAP to zero and FILTERARRAY[INDEX] = 1.
      2.4.2 If CURRENT[INDEX] > 0 set GAP = GAP + 1. If GAP < = MAXGAP set FILTERARRAY[INDEX] = 1.
      2.4.3 Increment INDEX.
   2.5 Set INDEX to zero. Set ARCLENGTH to zero. Set FOUND = 0.
   2.6 Scan through FILTERARRAY and for each element do the following:
      2.6.1 If FILTERARRAY[INDEX] = 1 in-

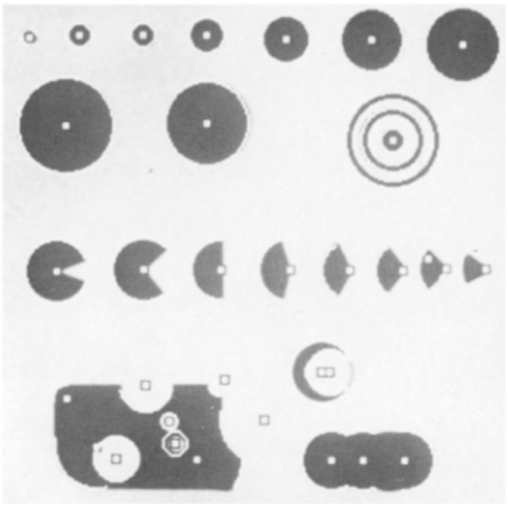

Figure 24. Found circular arcs marked in the test image.

crement ARCLENGTH
2.6.2 If FILTERARRAY[INDEX] = 1 and ARCLENGTH > MINARCLENGTH set FOUND = 1
2.6.3 If FILTERARRAY[INDEX] = 0 and FOUND = 1 we have found an arc. The parameters are:
DELTAFI = (ARCLENGTH-MAXGAP)/2.
FI = INDEX-MAXGAP-DELTAFI.
2.6.4 If FILTERARRAY[INDEX] = 0 set FOUND = 0.

We also have to detect when the arc passes through FI = 0. A wrap-around is easily detected and the sum of the sub-arcs can be used as a complete arc.

To make the function of the above algorithm clear, an angle histogram is plotted in Figure 23 together with FILTARRAY and the detected circular arcs. The found arcs are shown in Figure 24 overlaid on the test image.

3 Results and Experiments

The following properties of the suggested procedure should be tested:

- How small are the smallest detected circles and arc angles as a function of the noise level?
- How many percent of the arcs are detected correctly as a function of the noise level?
- How many false circles are introduced as a function of the noise level?

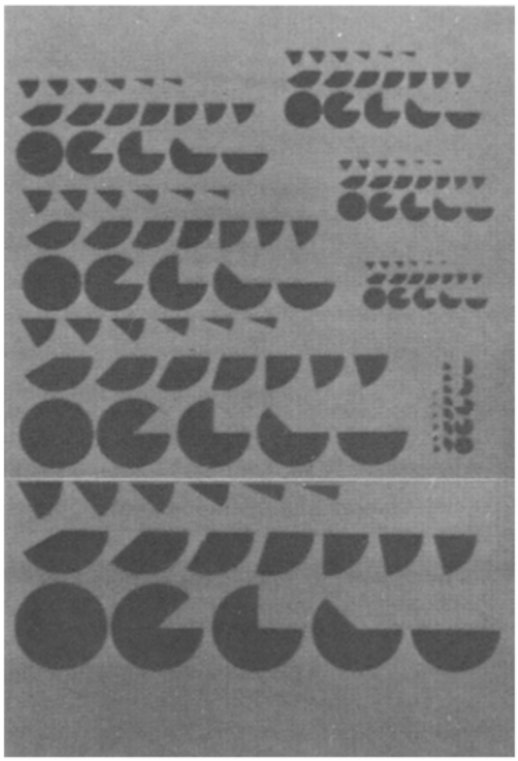

Figure 25. The test image containing circular arcs of different radii and size.

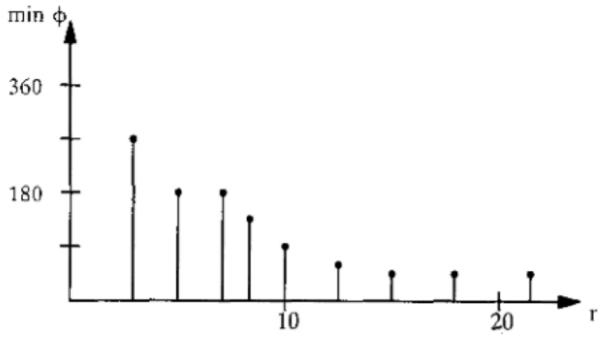

Figure 26. The arc detection program run on the images in Figure 25. Minimum detected arc size as a function of radius. No noise present.

In addition the procedure should also be applied to a few images of industrial parts.

The test image in Figure 25 are used to estimate the minimum radius and angular size detectable, as a function of noise. The image consists of 162 arcs with radii varying from 3 to 22 pixels and angular sizes from 20 to 360 degrees. If the method is run directly on this image, the detection result is shown in Figure 26.

From the diagram in Figure 26 we can conclude

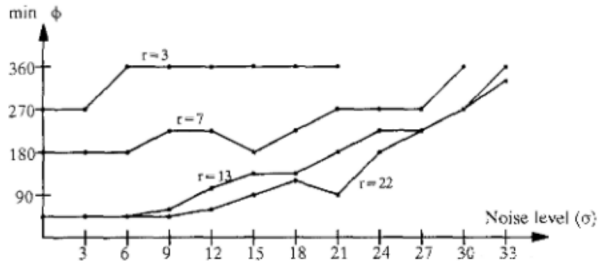

Figure 27. Minimum detected arc angle as a function of noise level for four different radii.

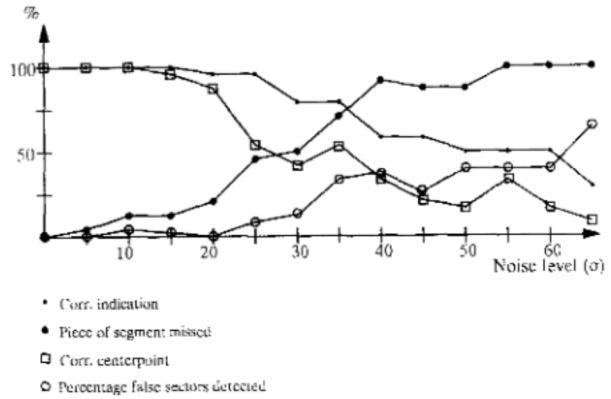

Figure 28. A few properties tested as a function of noise level.

that arcs as small as 45 degrees can be detected reliably if the radius is larger than 13. The method is not very radius dependent for larger circles. This is what could be expected since the number of feature points for each circle is proportional to circle radius and arc angle. For small circles there are not enough line segments to create a peak prominent enough to be detected, hence a larger segment is needed.

Next, we added some noise to the image. This is Gaussian distributed, $N(x, \sigma)$ around the true pixel value, x, and σ is varied. In the diagram shown in Figure 27, the results from 12 measurements and 12 different noise levels are shown.

For complete circles, the noise level can be as high as 30 and most circles are still detected. If small arcs are to be detected reliably, the noise level has to be much lower (below 12) and some of the detected arcs have a radius estimate that is not very accurate. If the center point or radius deviate more than the three pixels from the correct value, the arc is not considered to be reliable enough and is not taken into account.

A few tests were done using the test image shown in Figure 9. The results are shown in Figure 28. All

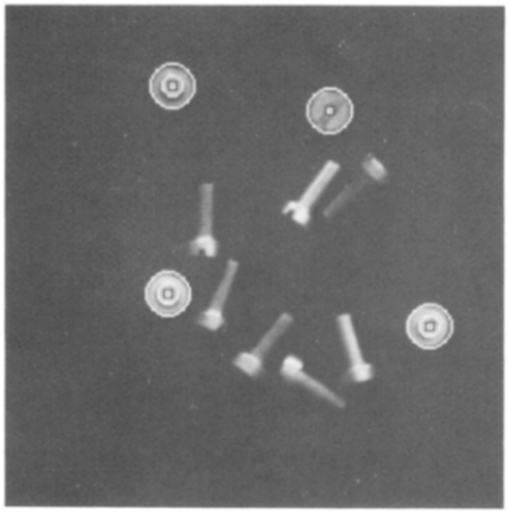

Figure 29. Metal bolts.

circular arcs smaller than 90 degrees or having a radius smaller than 5 pixels are neglected.
The resulting image is interpreted in the following way:

- An arc is correctly indicated if a detected arc is a part of it and the center position and arc radius do not deviate more than three pixels from the correct value.
- A detected arc has a correct center point if the center point does not deviate more than one pixel from the true arc center point.
- A piece of segment is missed if one arc is detected as several arcs having the same radius. This will give arcs with incorrect ϕ and $\Delta\phi$ values.
- The percentage of false arcs detected is the quotient between the number of arcs detected that have no relevance and the total number of indicated arcs.

To test the real case performance we tried the method on a few images containing industrial parts. The results are shown in Figures 29 to 34. The circle center points are found with an accuracy of ± 1 pixel in almost all cases. In Figures 31 and 32, the detected circles do not always follow the edges of the objects. This is especially prominent for large objects. The reason for this is that the objects become slightly ellipse-shaped due to projection transformations and that a large gradient estimation kernel is used where there is edge information even at some distance from the edge. This problem could, and should of course, be solved by using a better edge detection operator, e.g., Canny's edge detection (Canny 1983). If the circle center points are very close (as in Figure 32,

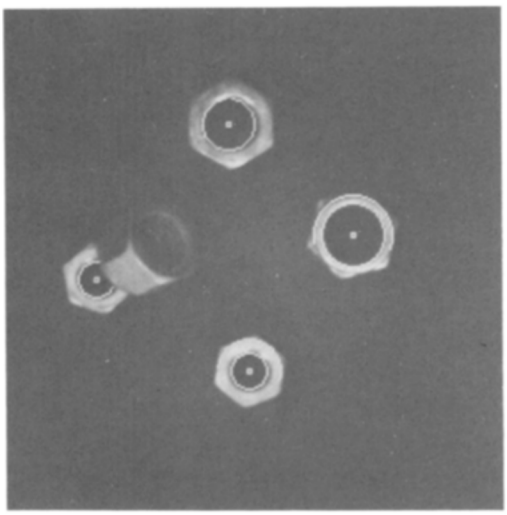

Figure 30. Metal nuts.

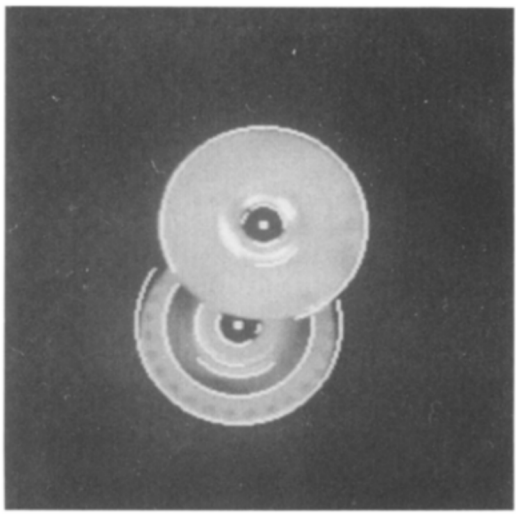

Figure 32. Gears.

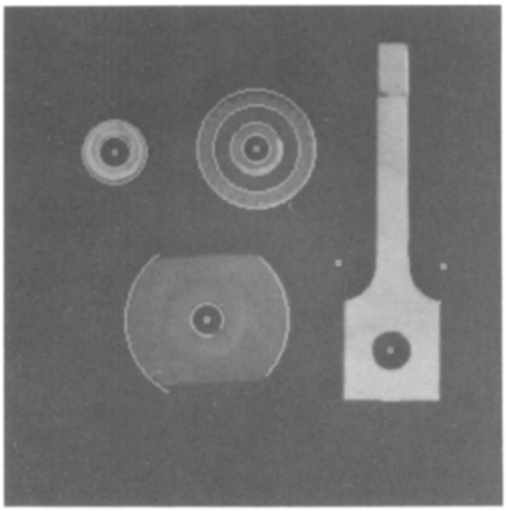

Figure 31. Some industrial parts.

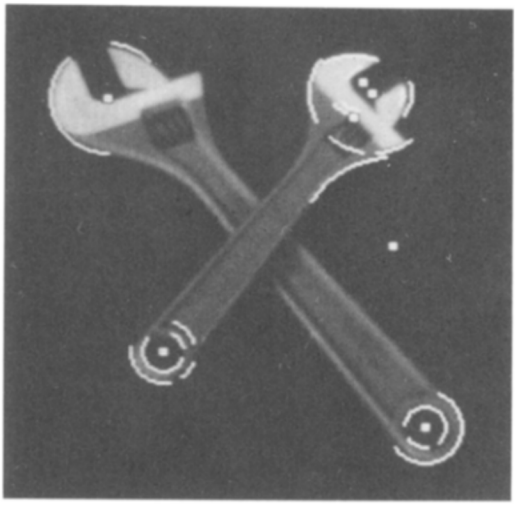

Figure 33. Wrenches.

upper wheel) the circle with the larger radius will most likely be detected, since a larger number of edge points contribute to the peak in the center.

3.1 Processing Times

The processing time for an empty image was 15 s on a SUN SPARCstation 2. It is mainly the gradient filtering (7·7) and the maxima detection filtering (9·9) that are time consuming. The processing time for the test image was 31 s and 40 arcs were detected.

The program is not at all optimized. The gradient estimation kernels used are separable, but this was not exploited. A separation of the kernels would probably make the filtering run considerably faster. Furthermore, there are conventional hardware filtering processors available today that could be used. In a real-time application, special purpose hardware would be useful for performing the Hough transform and the radius and angle histogramming. If the same hardware could be used for detection of other well-known features, the total performance-to-cost ratio of the system would be more favorable.

4 Conclusions

A circular arc detection method based on the Hough transform has been described. The method uses only a two-dimensional transform space and extracts the

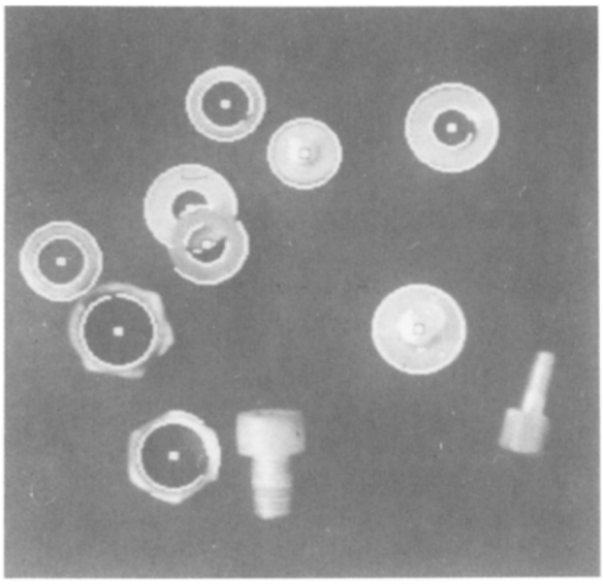

Figure 34. Washers, nuts, and bolts.

five parameters in three transform steps. The method is found to work well for noise-free images and images with moderate noise levels. All five parameters associated with circular arcs are estimated, but angular size and arc orientation are only accurate at small noise levels. For arcs with small angular size, the radius is difficult to estimate accurately, especially at a high noise level.

The number of false arcs detected does not seem to be a problem, at least for low noise levels and angular sizes larger than 45 degrees.

If the method is to be used commercially, speed requirements would probably suggest that both the filtering and the Hough transformations should be implemented in special purpose hardware.

4.1 Improvements

The method can be improved in a number of ways, for instance the following.

- In images with a high noise level, a consistency operation should be applied after the gradient estimation filtering. This would suppress pixels with a gradient caused by noise, and probably improve the angular estimate for true gradient pixels.
- In the present application, separable convolution kernels were used, but the separation was not implemented. Such an implementation would save a lot of time at the filtering stage.
- The detection of maxima in the Hough space could probably be improved a lot.
- In our experiments, we used 256·256 images for memory requirement reasons. If the image resolution could be increased, much better performance could be expected.
- A confidence number can easily be calculated that tells how distinct a detected arc is. The calculation could take into account such properties as filter response, radius, angular size, number of detected edge elements per arc length, etc. Such a number would be of great use in higher level computation.
- As mentioned in Section 2.7, some time could be saved if more information was saved in the radius histogramming step to be used in the angle histogramming. However, if the Hough transforms are implemented in hardware, this hardware could be used for both the radius histogramming and the angular histogramming steps, and thus the processing time for the angular histogramming would not be a problem.
- The method can not handle arcs whose center points lie outside the image. To handle this, the transform space has to be augmented beyond the border of the original image.

4.2 Future Work

During the work described in this paper, a number of questions and new ideas have arisen that might be subjects of future research. The most important is the execution time problem: Is it possible to optimize the algorithms to make the program run in a reasonable time on conventional hardware or is it necessary to design a special purpose transform hardware that performs the actual time-consuming transformations? If so, how should this hardware be designed to fulfill the requirements? Could it be used to detect other features in a similar way?

Ellipses could possibly be detected with an extension of the described method. An ellipse can be partitioned into a number of less perfect circular arcs. These arcs can be detected with the method described in this paper and then combined to evaluate a description of an ellipse. The question is just how this description can be derived from the parameters of the circular arcs and how large a segment of the ellipse must there be for it to be detectable.

4.3 Program Availability

The algorithms for the method described in this paper were developed on a SUN workstation. The source-code is written in C and is available on request (send a 3½″ floppy, name, and address to the author).

Acknowledgments. This work was supported by the Swedish National Board for Technical Development No. 753-

87-01954. The author also wants to thank Prof. Per-Erik Danielsson and Dr. Klas Gralén.

References

Ballard, Brown (1981) Generalizing the Hough transform to detect arbitrary shapes. Pattern Recognition, 13(2):111–122

Canny (1983) Finding edges and lines in images. TR No 720, AI Lab, MIT, Cambridge, Mass.

Danielsson PE, Seger O (1990) Rotation invariance in gradient and higher order derivative detectors. Computer Vision, Graphics and Image Processing 49:198–221

Danielsson PE, Seger O (1989) Generalized and separable sobel operators. In: Freeman H (ed) Machine vision-acquiring and interpreting the 3D scene. Academic Press, New York, pp 347–379

Davies ER (1985) Radial histograms as an aid in the inspection of circular objects. IEE Proc. 132D(4):158–163

Davies ER (1988) A modified Hough scheme for general circle location. Pattern Recognition Letters 7:37–43

Duda RO, Hart PE (1975) Use of the Hough transform to detect lines and curves in pictures. CACM 18(9):509–517

Hough PVC (1962) Methods and means for recognizing complex patterns. U.S. Patent 3,069,654

Illingworth J, Kittler J (1987) The adaptive Hough transform. IEEE Transactions on Pattern Analysis and Machine Intelligence 9(5):690–697

Illingworth J, Kittler J (1988) A survey of the Hough transform. Computer Vision, Graphics and Image Processing 44:87–116

Kimme C, Ballard DH, Slansky J (1975) Finding circles by an array of accumulators. CACM 18(1): 120–122

Lyvers EP (1988) Precision edge contrast and orientation estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence 10(6):927–937

Yuen HK, Princen J, Illingworth J, Kittler J (1990) Comparative study of Hough transform methods for circle finding. Image and Vision Computing 8(1):71–77

Qin-Zhong Ye (1989) Contributions to the development of machine vision algorithms. Linköping Studies in Science and Technology, Dissertations no. 201, Linköping, Sweden, part II