https://doi.org/10.2352/EI.2022.34.10.IPAS-345

Rapid circle detection through fusion of summative statistics of edge components

Scott Craver and Pheona Angoy; Binghamton University; Binghamton, NY USA

Abstract

Circle detection of edge images can involve significant time and memory requirements, particularly if the circles have unknown radii over a large range. We describe an algorithm that processes an edge image in a single linear pass, compiling statistics of connected components that can be used by two distinct least square methods. Because the compiled statistics are all sums, these components can then be quickly merged without any further examination of image pixels. Fusing multiple circle detectors allows more powerful circle detection. The resulting algorithm is of linear complexity in the number of image pixels, and quadratic complexity in a much smaller number of cluster statistics.

Introduction

Detection of circles in computer images is a long-standing problem with a number of proposed algorithms, which can involve significant time and memory requirements. Our goal is a reliable algorithm with low time complexity, that can detect multiple circles, partial or occluded circles and circular arcs, and circles with a wide range of radii.

Existing algorithms include brute-force search algorithms such as the Circle Hough Transform [1], and randomized algorithms such as randomized circle Hough, or gradient vector pairs citerch,gvp. Such complexity is needed because most of a computer image's content will be non-circle content, and circle content may consist of multiple circles. If one only has a data set of points known to be on one circle, there are simple and rapid methods of estimating a circle of best fit, two of which we summarize below; but the challenge then is separating and classifying those data points in a natural image.

In this paper, we consider natural images that are subjected to edge processing, in our case Canny edge detection [4], producing a binary image whose foreground pixels are edge pixels. Our proposed method slightly amends Canny edge detection, both to exclude vertex pixels and to save gradient vectors computed in the edge detection process. We then segment the edge image into connected components with a single pass over the edge pixels, and accumulate summative statistics for each component. These statistics can be used to determine if a component is a likely circular arc; subsequently, components can be quickly merged if their statistics produce a match to the same arc, resulting in an image consisting of likely circles.

Least Squares methods

Our algorithm combines one or more least squares methods as kernels for classifying groups of pixels. Least squares methods provide a very fast, accurate and optimal estimate of a circle that best fits a data set; however, they work when one is given points on a single circular arc, as opposed to a data set comprising multiple circles overwhelmed by non-circle (noise) pixels.

We observe, however, that least squares methods often employ summative statistics, for example statistics of the form $\sum_k x_k^a y_k^b$ for data points at locations (x_k, y_k). These sums are not only easy to accumulate in a single pass over the data, but they are also easy to combine. If we can segment a computer image into components, such that a component may be mostly edge data from a single circle, a least squares method may be applied to that segment; but in addition, two segments on the same circle can be rapidly identified by adding their summative statistics and recomputing the circle estimate and goodness of fit. Crucially, this does not require another pass over the data set to perform the circle fit.

If multiple least squares methods employ the same set of summative statistics, it is also possible to use several, for improved detection, with a modest increase in computational complexity.

Our requirements for a least squares kernel are therefore:

1. A least squares estimate must be computable only from aggregate pixel statistics, such as sums, that can be accumulated in a single pass over the pixels and combined by a simple operation.
2. The least squares estimate must combine these statistics by a simple formula of constant-time complexity.
3. The estimate must include not only circle parameters but the MSE of the estimate.

We describe two methods below, that we use in our algorithm. One identifies a circle of best fit in terms of pixel location alone; the other uses luminance gradient information, and we thus hypothesize that fusing both will provide improved performance.

The Kåsa method

The Kåsa method of circle fitting inputs a collection of data points at locations $\{(x_1, y_1), \cdots, (x_n, y_n)\}$ and fits a circle that minimizes the total error $\sum_{k=1}^{n} {E_k}^2$, where the error is defined as:

$$E_k = \left\| (x_k, y_k) - (c_x, c_y) \right\|^2 - R^2$$

This error is *not* the distance from a point to the circle, but an alternate metric chosen so that the resulting estimate is easily computable. With this error metric the estimate is:

$$
\begin{aligned}
C_x &= \frac{1}{2D}\begin{vmatrix} \sum(x_k^3+x_k y_k^2) & \sum(y_k^3+x_k^2 y_k) & \sum(y_k^2+x_k^2) \\ \sum x_k y_k & \sum y_k^2 & \sum y_k \\ \sum x_k & \sum y_k & n \end{vmatrix} \\
C_y &= \frac{1}{2D}\begin{vmatrix} \sum(y_k^3+y_k x_k^2) & \sum(x_k^3+y_k^2 x_k) & \sum(x_k^2+y_k^2) \\ \sum x_k y_k & \sum x_k^2 & \sum x_k \\ \sum y_k & \sum x_k & n \end{vmatrix} \\
D &= \begin{vmatrix} \sum x_k^2 & \sum y_k x_k & \sum x_k \\ \sum x_k y_k & \sum y_k^2 & \sum y_k \\ \sum x_k & \sum y_k & n \end{vmatrix} \\
nR^2 &= \sum x_k^2 + nc_x^2 - 2\left(\sum x_k\right)^2 + \sum y_k^2 + nc_y^2 - 2\left(\sum y_k\right)^2
\end{aligned}
$$

The total error of the estimate can also be expressed in terms of summative statistics, as:

$$
\begin{aligned}
\text{Err} &= \sum x_k^4 + \sum y_k^4 + 2\sum x_k^2 y_k^2 \\
&\quad -4c_x\left(\sum x^3 + \sum xy^2\right) + 4c_y\left(\sum y^3 + \sum yx^2\right) \\
&\quad +4\left(c_x^2\sum x^2 + c_y^2\sum y^2 + 2c_x c_y\sum xy\right) \\
&\quad +2A\left(\sum x^2 + \sum y^2 - 2c_x\sum x - 2c_y\sum y\right) + nA^2
\end{aligned}
$$

...where $A = c_x^2 + c_y^2 - R^2$. We observed that this total error function can be written as $\text{Err} = EnR^2$, where E denotes an error metric that is consistent across arcs of varying sizes.

Gradient regression

Suppose edge pixels are determined from luminance gradients in a computer image, as is the case in our problem. If an edge follows a circular arc, normalized luminance gradients at the edge points will satisfy a linear relationship with the edge coordinates. To see this, consider without loss of generality a disk whose luminance is darker than its background. The luminance gradient ∇ at the edge will point outward. At an edge point (x,y) we have

$$
\begin{aligned}
(x-c_x)^2 + (y-c_y)^2 &= R^2 \\
\nabla &= A(x-c_x, y-c_y) \\
\nabla/\|\nabla\| &= A(x-c_x, y-c_y)/AR \\
&= \frac{1}{R}(x,y) - \frac{1}{R}(c_x,c_y)
\end{aligned}
$$

If we collect the coordinates (x_k,y_k) and normalized gradient values which we label $\vec{n}_k = (nx_k, ny_k)$, then we can perform linear regression to minimize the error

$$
E = \sum_k e_k^T e_k, \qquad e_k = \left(R\begin{bmatrix} nx_k \\ ny_k \end{bmatrix} - \begin{bmatrix} x_k \\ y_k \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}\right)
$$

Computing the partial derivatives with respect to R, c_x and c_y and setting them to zero, we have the following equations:

$$
\begin{aligned}
\frac{1}{R} &= \frac{\sum nx_k x_k + \sum ny_k y_k - \sum nx_k \sum x_k/n - \sum ny_k \sum y_k/n}{\sum x_k^2 + \sum y_k^2 - \left(\sum x_k\right)^2/n - \left(\sum y_k\right)^2/n} \\
c_x &= \left(\sum nx_k - \frac{1}{R}\sum x_k\right)/n \\
c_y &= \left(\sum ny_k - \frac{1}{R}\sum y_k\right)/n
\end{aligned}
$$

Despite their apparently complexity, all of these estimates are computed from summations assembled from a linear pass over the edge pixel data; in terms of asymptotic complexity, both of these algorithms run in $O(n)$ time for an image of n pixels, and the above formulas run in constant time if the sums are precomputed. In particular, if one has two datasets whose sums are precomputed, applying either least squares method to the combined dataset requires a constant-time operation of adding the corresponding sums.

Gradient computation

Gradient regression requires that we have, for each edge pixel, an estimate of the luminance gradient direction (normalized gradient) at that point. Such gradients are computed as a stage in the Canny edge detection algorithm, although they are immediately quantized to one of four orientations: vertical, horizontal, diagonal northeast, and diagonal northwest [4]. However, it is a simple matter to save the estimated gradient data prior to quantization, to accumulate the sums.

Overall algorithm

Our algorithm is illustrated in figure 1. We first commit edge detection on a computer image, and then segment the edge pixels into connected components. For each component, as we assemble it, we assemble a collection of sixteen summative statistics, based on pixel coordinates and gradient values. At that point, the image can be discarded, as the rest of the algorithm only operates on the summative statistics of the assembled components. Components with few pixels are discarded.

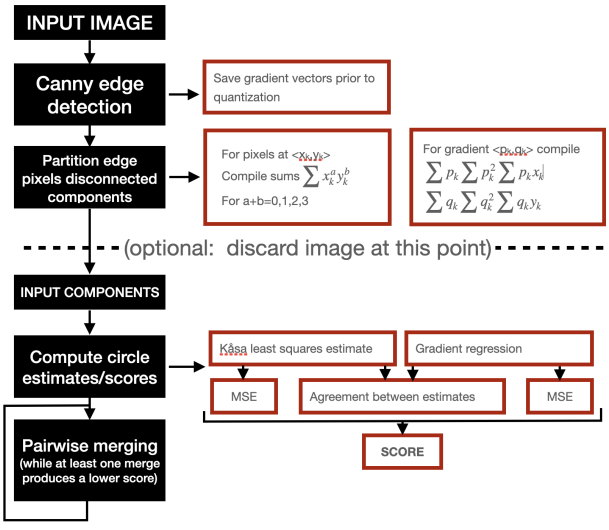

Figure 1. *The overall structure of the proposed method.*

Given a set of edge pixel components, we test each component using least squares estimates, determining which have a high goodness of fit to a circular arc. In a final phase, we attempt to merge components, adding their statistics, if doing so increases their goodness of fit.

Segmentation into connected components

The segmentation algorithm we employ requires only a single pass over the image, and it (along with accumulating statistics) can be performed as Canny edge detection classifies edge pixels. The algorithm is illustrated in figure 2. The image is traversed top to bottom, left to right, with each foreground (edge) pixel labeled with a component. When a foreground pixel is encountered, we check the four pixels above and to its left, in other words neighboring pixels that have already been evaluated by our traversal. We then switch based on the foreground pixel's neighbors:

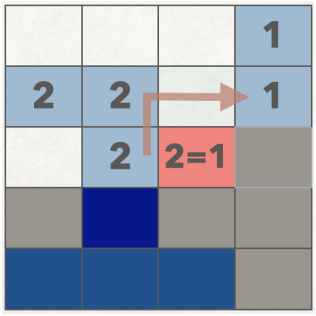

Figure 2. *Image traversal to assign edge pixels to connected components.*

- If there are no foreground pixels there, our new foreground pixel is labeled as belonging to a new component, which is allocated with its own statistical accumulators initialized to 0.
- If neighboring pixels are found labeled as belonging to a specific component, our new foreground pixel is also labeled as belonging to that component, and its location and gradient are used to update the statistical accumulators for that component.
- If neighboring pixels are found belonging to more than one component, the components are first merged. This consists of adding one components statistics to the other, and updating a reference table so that pixels with both component labels are mapped to a single label for the merged component. Then, the new foreground pixel's location and gradient are used to update the accumulators of the merged component, as above.

At the end of this pass over the image, we then have a set of statistical accumulators for every connected component, providing summative statistics for each. We discard those components that have few pixels; in our experiments any component with fewer than 12 pixels is discarded.

Classification and merging

Once our components are assembled, our next step is classifying likely arcs. The summative statistics per component are used to compute the centers, radii, and goodness of fit for a circle, using both the Kåsa method and gradient regression as described above. For the Kåsa method, we use the normalized error $E = \mathrm{Err}/nR^2$. In experiment we observe that for certain circular arcs, E is approximately 12.5 times the mean squared error for gradient regression. We thus use the score $S = (E/12.5)^2 + (MSE)^2$, where MSE denotes the mean square error under gradient regression.

There are other statistics and heuristics that can be employed when classifying a segment as a likely circle. One is the radius size, or center location: if the circle's center is far from the image rectangle, it means that our cluster may be a line segment rather than an estimable circular arc. The "arc length" of a segment can be estimated using the number of pixels in the segment versus its radius, and those segments of small arc length can be discarded as containing insufficient information to classify them as circular. Another heuristic method available to us is comparing the circle estimate of our two fused methods, to determine if they agree sufficiently. However, for our initial exploration, we only use the combined errors of circle fit, which we compare to a threshold.

Implementation and testing

To test our algorithm, we produced pseudo-random zig-zag images to use as non-circle data. An example is illustrated in figure 3. These are wedges of random location and angular span, which are combined by exclusive-or in a binary image plane. We can also add disks and disk wedges to these images, isolating their edges to collect statistics for arcs versus non-arcs.

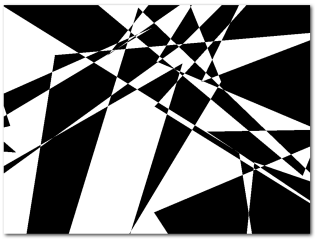

Figure 3. *A generated zig-zag image for testing the algorithm.*

Figure 4 shows a plot of Kåsa error metric E on the horizontal axis, versus gradient error on the vertical axis, for zig-zag images versus generated circular arcs. The striking pattern for circular images corresponds to our observation that E for many true arcs is about 12.5 times the gradient error.

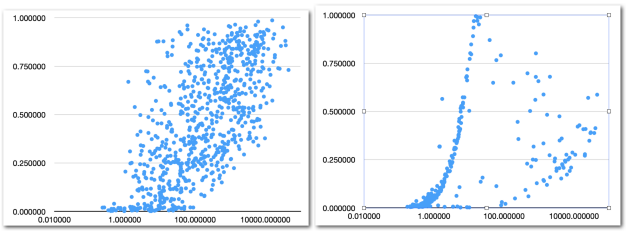

Figure 4. *Error rates for zig-zag images (left) and circular arcs (right).*

Upon inspecting the segmented images, we noticed that segmentation would mistakenly combine into polygons multiple edge

elements corresponding to distinct edges. These polygons produce the risk of false positives, and an edge fusing with an arc produces the risk of a false negative. We thus modified the Canny edge detector with one extra, final step in its edge pixel classification: a candidate edge pixel is compared to its immediate neighbors, and rejected if neighboring pixels have at least two orientiations distinct from that of the candidate pixel. This removes "tee" intersections as well as sharp bends in the image plane; figure 5 shows the segmentation before and after the addition of this step. Figure 6 shows the plot of E versus gradient MSE for zig-zags and arcs after this vertex culling step: the result is data that is much better behaved. Vertex culling was also observed to separate circles in natural images, from adjoining edges that previously confounded our attempts to identify them.

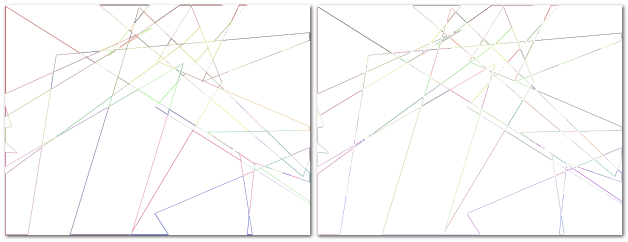

Figure 5. *Segmentation of zig-zag image without (left) and with (right) vertex culling.*

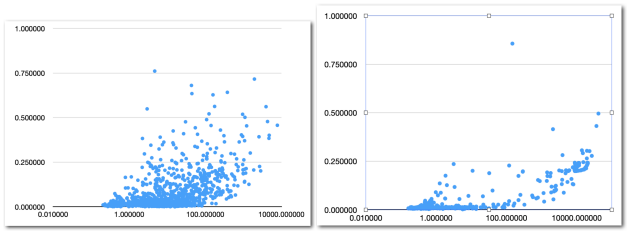

Figure 6. *Error rates for zig-zag images (left) and circular arcs (right), after vertex culling.*

Finally, we show the result of this algorithm on natural images, below. The figure shows, for each image, its edge components, components that satisfy an error threshold of $0.01n$, where n is the number of pixels in the cluster; and finally the result of combining those clusters if their error metric improves by doing so.

Discussion and conclusions

We have described an overall approach to circle finding in computer images, that fuse multiple least-squares methods with edge segmentation, exploiting the ability to swiftly combine segments without requiring additional passes over the image.

The complexity of our algorithm is not much more than the linear time operation of edge detection (linear in the number of image pixels). Most of the work of the algorithm can be performed in the edge detection process, as edge pixels can be segmented and statistics assembled as those edge pixels are identified.

The subsequent processing is per cluster, and indeed the image can be discarded once it is segmented. Our results show that per-segment classification is already quite powerful by itself, and subsequent merging of clusters clearly improves the identification of circular arcs in the image.

One question left unanswered by this analysis is: how many segments are produced by this process in general? A naive attempt to merge them requires a number of trials that grows quadradically with the number of clusters. Thus far, our observations show natural images on the order of 1000x700 producing hundreds of candidate clusters; we suspect that the number of clusters grows sub-linearly with the number of pixels, but future work is needed to confirm this.

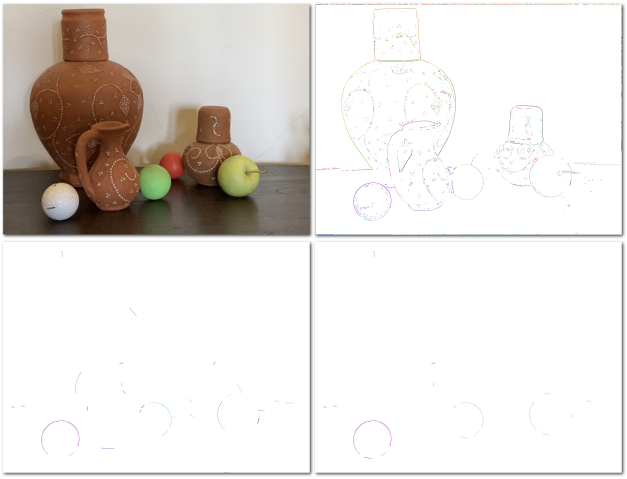

Figure 7. *Circle detection example. Upper right, an edge-segmeneted image; below left, segments below threshold; below right, segments after merging.*

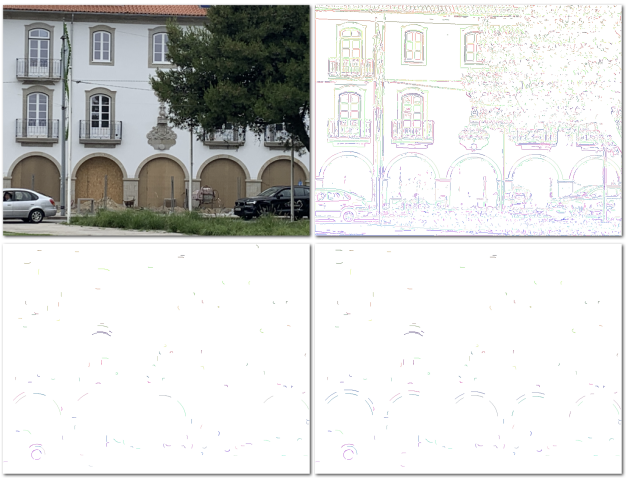

Figure 8. *Circle detection example. Upper right, an edge-segmeneted image; below left, segments below threshold; below right, segments after merging.*

Future work

The largest open question implied by our results is how we can best classify circular arcs based on their computed statistics. While we see promising performance by combining only the mean squared errors of our two least-squares kernels, there is significant potential to improve this by combining these or perhaps other metrics into a more sophisticated detection region.

In addition, we have yet to explore heuristic methods to discard unlikely segments, for example those whose estimated circle centers, radii or arc lengths are not likely to correspond to circles in natural images.

Finally, an obvious avenue for future work is combining other or more least squares kernels into this algorithm, to determine if or how much they may improve performance when their statistics are fused.

Acknowledgments

This research is the product of a summer research internship program supported by the Louis Stokes Alliance for Minority Participation (LSAMP).

References

[1] ER Davies, A modified Hough scheme for general circle location, Pattern Recognition Letters, vol 7 no 1, pg. 37–43. (1988).

[2] A Pekka Kultanen, Lei Xu and Erkki Oja, Randomized hough transform (rht), Proc. 10th International Conference on Pattern Recognition, vol 1, pg. 631-635. (1990).

[3] A.A. Rad, K. Faez and N. Qaragozlou, Fast circle detection using gradient pair vectors, Proc. VIIth Digital Image Computing: Techniques and Applications, 897-887. (2003.)

[4] McIlhagga, William, The Canny edge detector revisited. International Journal of Computer Vision 91.3, pg. 251-261. (2011).

Author Biography

Scott Craver received his PhD in Electrical Engineering from Princeton University in 2004, and is an assistant professor of Electrical and Computer engineering at Binghamton University in Binghamton, NY. Pheona Angoy is an Electrical and Computer engineering major at Binghamton University in Binghamton, NY.