Circle Detection Performance Evaluation Revisited

Elisa Barney Smith, Bart Lamiroy

Circle Detection Performance Evaluation Revisited

Elisa H. Barney Smith[1] and Bart Lamiroy[2]

[1] Electrical and Computer Engineering Department
Boise State University
Boise, ID 83725-2075
ebarneysmith@boisestate.edu
[2] Université de Lorraine – Loria (UMR 7503)
Campus Scientifique – BP 239
54506 Vandœuvre-lès-Nancy cedex – France
bart.lamiroy@loria.fr

Abstract. Circle and circular arc detection in images have been a long standing topic in image analysis. It finds numerous applications for both scanned document images as well as in photographic images. As a result, circle detection algorithms are published regularly and benchmarking data sets and contests have been organized on a regular basis over the last decades. Unfortunately, they have not been able to achieve a very clear image establishing which approaches perform best and under what exact conditions.
This paper contributes to circle and arc detection, by providing an open and fully reproducible framework for benchmarking and evaluating circle and circular arc detection methods. It builds upon the current state of the art and commonly used metrics by providing a complementary approach through the introduction of synthetic evaluation data for benchmarking versus two noise types at gradually varying noise levels and new performance metrics that are compatible with previous evaluation approaches.

1 Introduction

Circle and arc detection have been a long standing challenge in the Graphics Recognition community [?] and beyond. New algorithms and approaches are still being published on a regular basis without there being a clear knowledge of their actual performance with respect to the state-of-the-art. This paper is an attempt to provide an overview of past performance evaluation and benchmarking initiatives, and to initiate a more reproducible and complete way of assessing circle and arc detection methods.

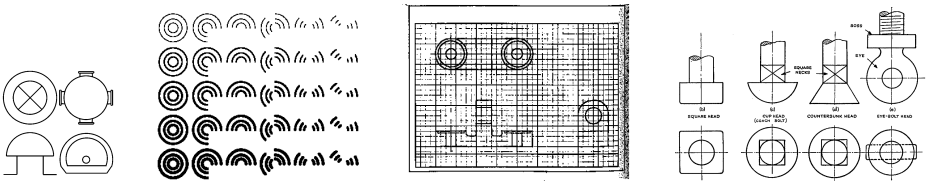
1.1 GREC Arc Detection Contests

The most consistent and complete series of benchmarking initiatives are the Circle and Arc Detection Contests that have been held in conjunction with the

IAPR Graphics Recognition Workshops (GREC). These events have been taken place every other year, and between 2001 and 2013, seven contests have been organised: [?,?,?,?,?,?,?].

For the purpose of objective benchmarking, each of these contests have contributed to providing annotated reference data. These either consist of synthetic data subjected to noise models, or of actually manually annotated scanned documents. The images are sometimes simple circles or parts of circles that do not touch, circles in patterns deliberately designed to fool the algorithm, or machine drawing plans showing the needed end application. Some of them are illustrated in Figure 1.

While different metrics have been proposed to evaluate the performance, the general consensus has been to use the one developed by Liu and Dori [12]. This metric will be detailed further in Section 4.1.



(a) (b) (c) (d)

Fig. 1: Samples from GREC circle and arc detection contests (a) 2003 (b) 2005 (c) 2007 (d) 2009.

1.2 Reproducibilty

We are not going to reproduce an extensive overview of the many exiting algorithms for detecting circles, as will be explained further. The only way for evaluating performances of these multiple approaches consists of testing them on unified benchmarks. However, very often, when a circle detection algorithm is developed, the authors choose some images on which to test it. Often these images are small and simple and the algorithm performs well. When applied in real problems, however, the images for which the algorithm is needed, will not be small and simple. Often they will contain noise from aging while the document has been in storage, or from the acquisition process. These are specifically the cases for which circle detection is needed. Therefore the end user will care not about how the circle detection algorithm performs in the ideal case, but how it performs when under stress, or will like to use the algorithm best suited for their operational conditions (speed, precision, image noise ...)

The contests described in section 1.1 provide an interesting step in the right direction, but fail to fully satisfy the actual goals. This is due to the following reasons:

1. They are time bound, and correspond to snapshots at the time they are run. Although the contests usually provide open access to their reference data sets, and therefore allow replay of the data on algorithms that were unavailable at the time, this condition is not reversible, in a sense that the competing algorithms cannot be applied to other reference data.
2. Getting access to published algorithms that have competed in the past, is very often quite difficult.
3. Changing metrics is impossible.
4. When human annotated reference data is used, often multiple interpretations of ground truth are possible [10], and it is difficult to assess whether difference in precision performance of specific algorithms is due to the quality of the algorithms or quality of the reference data ...

In this paper we introduce a method for comparing algorithms and to allow researchers to test their own work using shared and identical experimental conditions. Since the exact quality and quantity of noise is hard to measure on real scanned data, and the ground truth is hard to extract, we present a synthetic ground truth generator that is able to provide both realistic noise and drawing content. The code to create these images and to run the test procedure will be made available on the DAE platform [9].

2 Technical Background

Five different algorithms were used in this study. The choice of the actual algorithms was quite simple: we took those for which we were able to get a reliable implementation. This was either because their authors kindly provided us with their code or binaries, or because the algorithms were sufficiently documented in the supporting publications. A brief overview of each algorithm is presented here. Details for each algorithm are available in the referenced work. For a significant number of published methods, we failed to reach the authors or to correctly reimplement the algorithms based on the published descriptions only.

2.1 Hough 3D

The most well known and probably oldest method for finding a circle in an image the Hough Transform [6, 15]. It is often also used as a baseline for performance comparison. It is based on having a circle

$$(x - x_c)^2 + (y - y_c)^2 = R^2 \tag{1}$$

with unknown parameters x_c, y_c and R. For every known "on" point (x, y) the set of the possible x_c, y_c and R that could produce a circle containing that point are listed. The algorithm creates an accumulator grid for x_c and y_c center coordinates and radius R that are candidate parameter values. Then for every point in the image that is "on", all the x_c, y_c and R that could correspond to it are incremented in the accumulator. The parameters corresponding to the

accumulator bins where there is a peak are the parameters that correspond to circles in the image.

Implementing this requires a set of loops to map each "on" pixel to all the appropriate (x_c, y_c, R) accumulator bins. Finding the peaks heavily depends on the chosen bin resolution and on how many are selected from the accumulator. The number of peaks selected will determine the number of returned detected circles, but this needs also to be parametrized, either by setting an accumulator bin threshold level, or specifying the quantity of circles desired. It takes a fairly high amount of memory to store the accumulator array and also a fair amount of computational time to fill it. These operational requirements make it an undesirable algorithm to use in practice.

A number of well documented techniques for computational complexity reduction exist. Sometimes methods are used to reduce the number of "on" pixels that are used to fill the accumulator array, like those used in Sections 2.2 and 2.3. As a comparison baseline, we have implemented a version of the Hough Transform that uses the basic algorithm with a 3-dimensional accumulator grid. To make the algorithm run in a reasonable amount of time and memory, we introduced some algorithmic accelerations, while making sure the final mapping to the 3D accumulator array from the original points continued as in the original algorithm.

In the 1000×1000 pixel images used in this project, on the order of 20,000 pixels were "on" pixels. For each pixel all eligible (x_c, y_c, R) must be incremented within the accumulator. To reduce this loop size, the image was reduced $5\times$ resulting in a $125\times$ smaller accumulator array. To not lose points because of sub sampling, the reduced image array was set to "on" if any of the pixels in the 5×5 window were "on" in the original image.

All (x_c, y_c, R) values that had an accumulation of 90% of the expected circumference at that radius size were identified. For all potential radius sizes identified in the prefiltering step, the (x_c, y_c, R) values were used as a mask to select (and reduce) "on" points in the original image. The points that satisfied the mask were processed with the full resolution 3D Hough Transform at each candidate radius. As per the Nixon implementation [15], the accumulator array was created and analyzed for each radius value separately in turn reducing memory requirements. The list of (x_c, y_c, R) values that had an accumulation of 90% of the expected circumference at this full resolution were stored. A 3D accumulator again at $5\times$ reduction was used to group these points using a connected component algorithm. The original high-resolution (x_c, y_c, R) values that correspond with each cluster were then used to identify the circle centers and radii. The average x_c, y_c and R values in each cluster were selected as algorithm outputs. The range of radii in each cluster was used to determine stroke width.

This algorithm was implemented in Matlab and takes between 2 and 4 minutes for each 1000 by 1000 image (without pepper noise) on both an older 32 bit laptop and on a server that for other algorithms provided greater than a $3\times$ speedup. When pepper noise is introduced, the run-time increases. At a pepper

noise level of $P = 6$, it was taking 6 hours per image. The results for P$\geq$6 are therefore not shown.

2.2 Hough 2D - Hough Gradient Method

While the Hough 3D algorithm for circles is a standard, there are several variations to reduce its size and complexity. One algorithm related to Hough 3D and often called Hough for circles is based on the idea that the gradient of the edges of a circle will point toward the center of the circle. This algorithm is the one used in the OpenCV package [16].

The algorithm starts by finding all Canny edges. This reduces the search space from the original Hough algorithm. The gradient for each line in the image is calculated. At each "on" pixel location, a line in drawn in the direction of the gradient in a 2D accumulator space. Peaks in this accumulator space are designated as circle centers. Then the L_2 distance from each point to each circle center is calculated. If it is within the allowable radius range, an accumulator is incremented.

2.3 Hough Gradients - Jia

This algorithm [8, 7] is very similar to the Hough Gradient algorithm. The source code for it written in Matlab was provided by its authors. The gradient is used to find the intersection of lines and candidate circle centers. Then radii are tried consecutively to see if within a small distance from the radius there are many points with a high gradient. All radii that exceed a threshold are saved. Then the list is parsed to eliminate radii that are consecutive.

The original algorithm was designed for use on small natural scene images. We modified the source code provided by the authors to take advantage of Matlab vector processing accelerations. We also modified the algorithm that searches for the circle's radius to indicate at which radii a large percentage of the circumference contains black pixels, instead of just a high gradient. This also reduced the number of false concentric circles found.

2.4 QGAR Algorithm

This method was published in [11]. It being our own work, access to the source code was straightforward. It is constructed around a robust estimator determining whether there is a circular arc close to a given center (x_c, y_c) and radius σ. However, it needs some initial guess on where to search. It proceeds in three main phases:

1. Generate a high number of possible arc candidates, without consideration of uniqueness, overlapping or exact localization. The arc candidates are obtained by operating a line extraction algorithm on the data, and by considering the circular arc defined by two connected line segments.

2. Verify the quality of each candidate using the approach described below. The output of this verification is a list of genuine arcs, correctly fitted on the image data.
3. Detect and merge multiple and/or partial detections of the same curves as to obtain a set of unique, disjoint arcs.

In order to evaluate the quality of an initial arc guess, the approach is to find a set $P = \{p_i\}$ of all image pixels p_i radially closest to the theoretical discrete circular arc $\mathcal{A}_0$ at iteration 0. P is then used for updating the estimate of the arc parameters, resulting in a new arc estimate $\mathcal{A}_1$ for which the process is iterated until convergence. At convergence a fitness measure, based on a threshold of image pixels lying on the arc determines whether the estimate is to be rejected or not.

2.5 ED Circles Algorithm

The ED Circles algorithm [2] is based on an edge segment detector (ED) algorithm [1] that joins sets of identified edge points into edge segments. This results in an edge map that is not a set of points, but a set of edge segments or pixel chains. Sets of consecutive line segments are evaluated to determine if they are candidates to be parts of lines, circles/arcs, or other. If the angle between the segments is less than 6 degrees they are a considered colinear; if it is greater than 60 degrees, then a corner is likely present, and the segments are separated. Consecutive line segments with an angle between 6 and 60 degrees that turn in the same direction and have consistent angles values are defined as forming a circular arc.

Points corresponding to sets of segments that are candidates for a circle or arc are tested for a good least squares fit. If the fit is adequate, the next segment is a tried for addition of a following segment, otherwise the process stops and what is there is saved. Sets of arcs with calculated radii within constraints and close physical proximity are evaluated to see if they collectively form a circle.

As a final step the candidate lines, circles and arcs are validated with the Helmholz principle [5]. This looks at the probability that any mismatches (false alarms) occurred by chance.

Since EDCircles is designed to work on natural scene images, the edge detector will find both the interior and exterior edge on document images. We therefore modified the output such that, when two concentric circles (circles with center positions having an L_2 distance less than 10 pixels) were found with radii differing by less than 10 pixels, we averaged the center pixel coordinates and radii were averaged. The difference in the radii determines the stroke width.

The authors did not provide the source code for this algorithm, but they have a web portal [3] where images can be tested. They only output detected circles, even though their published algorithm is capable of detecting arcs and lines.

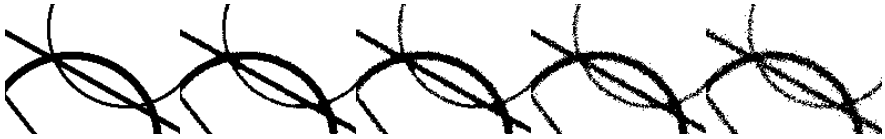
3 Data and Experimental Protocol

In order to provide a perfectly controlled, yet realistic experimental data set, we create images that contain a mixture of circles, arcs and straight lines with known positions and stroke thicknesses, as well as genuine noise. We want to create a large number of synthetic images with several noise levels and type.

Two types of noise were added to the images: edge noise and impulsive or pepper noise. While in physical documents both noise types can occur in the same image samples, the two types of noise are treated separately here to see the effect that each type of noise has on the circle and arc detection algorithms.

3.1 Edge Noise

The edge noise is based on image degradations produced by the Baird degradation model [4]. This models the bilevel image acquisition process, which contains blurring from the optics and additive Gaussian noise from the sensors. This is then thresholded with a global threshold to form a bilevel image. With a threshold at 50%, the position of the edge will not change, but corners (*e.g.* at intersections) will round slightly. In the gray-scale image before thresholding, the blurring smooths the transition of the amplitude from white to black. The additive noise on this sloped surface will at times be above the threshold and at times below the threshold. Therefore the noise will cause a variation in the pixels at the edge. The distance from the edge where the noise has this transition effect is called the Noise Spread (NS)[14]. We use 0.5, 1.0, 1.5 and 2.0 pixel NS (*cf.* Figure 2). At larger values, the noise can exceed the threshold and cause isolated or impulsive noise. To separate the effects of the different noise types, all isolated pixels are filtered to white.



(a) (b) (c) (d) (e)

Fig. 2: Samples of images with varying levels of Noise Spread. (a) Original image without noise, (b) NS=0.5, (c) NS=1.0, (d) NS=1.5, (e) NS=2.0.

3.2 Impulsive Noise

The second type of noise is impulsive or pepper noise in which each pixel i is transformed into i_t with a probability

$$\begin{aligned} \mathbf{P}\left(i_t = black|_{i=white}\right) &= \mathbf{P}_{pepper} \\ \mathbf{P}\left(i_t = white|_{i=black}\right) &= \mathbf{P}_{salt}. \end{aligned} \tag{2}$$

This is the noise that is more easy to see and is the type of noise used for the arc detection contest of GREC 2005. The physical source of this noise is the same as the source of the edge noise. It usually appears in systems where the sensor noise or the paper texture is great, and the binarization threshold is low. Edge noise will accompany it in physical systems. For this paper, the noise was added without blurring, as was the case in GREC 2005 and [13]. Therefore the edges have no transition zone between black and white (*i.e.* NS = 0). In this paper we use only pepper noise so the strokes are not affected directly. We empiricially estimated $\mathbf{P}_{pepper}$ from the GREC 2005 dataset and determined eight levels: $\{0.0005, 0.005, 0.026, 0.045, 0.073, 0.11, 0.125, 0.16\}$. These are referenced as pepper noise levels $P = 1..8$ in the remainder of the paper (*cf.* Figure 3).

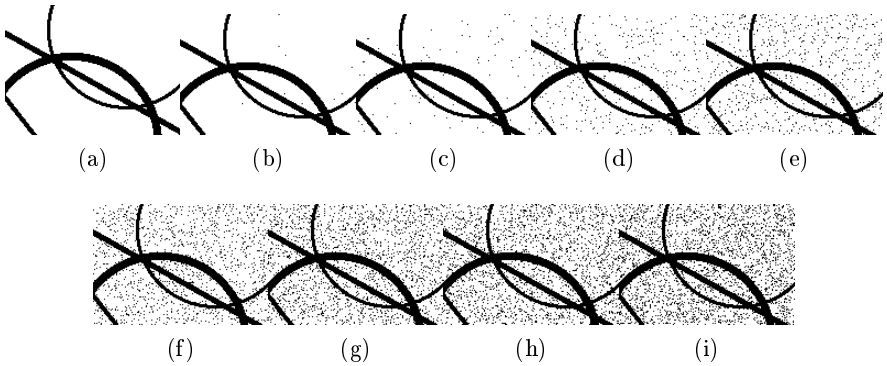


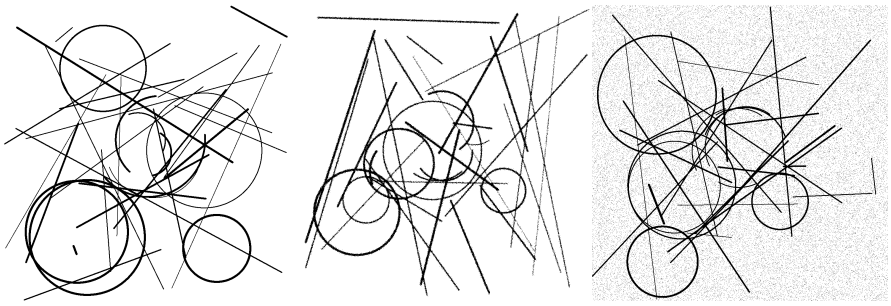
Fig. 3: Samples of images with varying levels of pepper noise. (a) Original image without noise, (b)-(i) $\mathbf{P}_{pepper}$ = 0.0005, 0.005, 0.026, 0.045, 0.073, 0.11, 0.125, 0.16.

3.3 Image Data

Images generated for this study all contain 5 circles, 5 arcs and 25 line segments. The circles have a random center and radius, drawn from a uniform distribution $U[100, 900]$ for the center coordinates, and $U[50, 200]$ for their radius. The arcs have a supplementary starting angle within $U[0, 360]$ and a span within $U[30, 180]$ degrees. The line segment end points were drawn from a random distribution $U[1, 1000]$. Each component has a randomly chosen stroke width between 2 and 7 pixels. To generate a line with a thickness, the object was created with a one pixel thickness and then morphologically dilated by a circle of the appropriate diameter.

We created 10 different randomly generated line drawings of size 1000×1000 (*cf.* Figure 4). They were created initially at $4\times$ the resolution so the blurring process that creates the edge noise could be done with discrete convolution,

and then downsampled to return the images to the "original" resolution. 10 instances of noise were added to each of the edge noise images at each noise level. 5 instances of noise were added to each of the impulse noise images. This produced a total of 100 noisy images for each noise level for edge noise and 50 for each impulse noise level. There were 400 total noisy images for each type of noise. There were 10 images without noise used as reference.



(a) (b) (c)

Fig. 4: Sample test images. (a) without noise (b) NS=2.0 (c) Pepper=0.045%.

4 Experiments and Results

We compared the performance of five circle detection algorithms versus two types of noise at varying noise levels. We used two metrics to quantify their performance. We next discuss the metrics, and then show the performance of the algorithms.

4.1 Performance Metrics

Two performance metrics were used to evaluate the quality of the circle detection. The first is ArcEval by Liu and Dori [12], which is the one used in past GREC arc detection contests. It contains measurements of the amount of positive match between the estimated circles and the negative or false alarm mismatch between the two images. While the original paper includes multiple metrics we only consider the vector performance metrics, D_v, F_v, and VRI. The implementation we are using is the executable from the 2009 GREC contest.

The detection rate is

$$D_v = \frac{\sum_{g \in V_g} Q_v(g) l(g)}{\sum_{k \in V_d} l(g)}. \tag{3}$$

The quantity $Q_v(g)$ is the vector detection quantity and is made from a combination of the quality of the match in the endpoints, overlap distance, line width, line style and line shape. It can take values from 0 to 1, and a value of 1 is desired. V_g is the set of ground truth vec objects, and V_d is the set of detected vec objects. $l()$ is the length of the vec object stroke. The false alarm rate

$$F_v = \frac{\sum_{k \in V_d} F_v(k) l(k)}{\sum_{k \in V_d} l(k)} \tag{4}$$

is the length-weighted sum of the false alarm factors $F_v(k) = 1 - Q_v(k)$, meaning one minus the match of a line to the image that shouldn't have been there, or a 100% match to the background. We desire F_v to be zero. Similar to the F-measure in retrieval, the detection and false alarm rates are combined, but in a weighted arithmetic mean

$$VRI = \beta D_v + (1 - \beta)(1 - F_v). \tag{5}$$

The weight factor β is set to 0.5. We also want VRI to be 1.

The Liu–Dori metrics rely on the percent overlap of the detected strokes with the ground truth strokes. Some of the circle detection algorithms produce a list of circles that are plausible indicating the algorithm did indeed find real circles and not random noise, but have a one or two pixel error in the circle center position and/or the radius length. This can result in zero stroke overlap and thus a score of 0 for V_p, 1 for V_{fa} and 0 for VRI. This does not accurately reflect the algorithms' performance. Thus a second evaluation metric was created. This metric looks at the percent overlap of the circle area. To liken it to the Liu–Dori metric, a version for positive overlap, C_d, and false-alarm non-overlap, C_f, were created and then averaged. This starts with the overlap percentage calculated by

$$Ov(C_1, C_2) = \frac{\overline{C_1 \bigcap C_2}}{max(\overline{C_1}, \overline{C_2})}, \tag{6}$$

which is the area of the overlap between circle 1, C_1, and circle 2, C_2, divided by the maximum area of the circles individually. The overlap area is calculated algebraically [17] based on the vec data.

C_d measures for each ground truth circle, whether there is a detected circle that strongly overlaps ($\geq 50\%$) it and if so, their average overlap metric:

$$C_d = \sum_{g \in V_{gt}} Ov(C_{d^*}, C_{gt})/N_{gt}. \tag{7}$$

C_f measures the percentage of detected circles that do not have a match ($\geq 50\%$) with a ground truth circle

$$C_f = 1 - \sum_{d \in V_d} Ov(C_d, C_{gt})/N_d. \tag{8}$$

For each circle this requires finding which ground truth circle is the best match to it, so it will only be used once in the match calculations. For instance, we do not

want several circle overlapping one ground truth circle to all count as positive matches. The excess ones are false alarms and should penalize the algorithm. As in Eq 5 the detection and false alarm rates are averaged

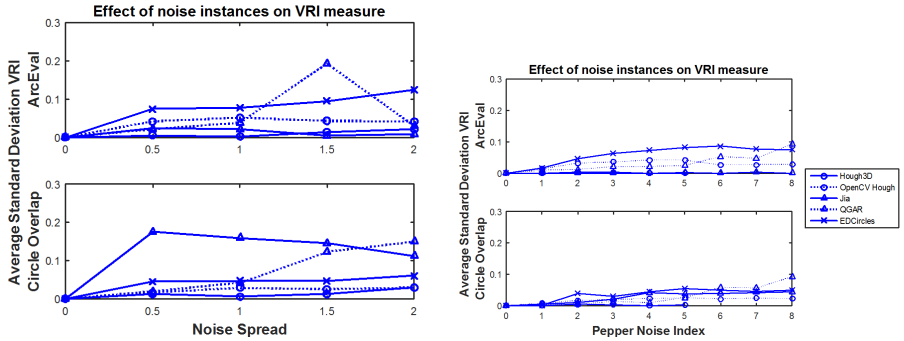
$$VRI_C = \beta D_v + (1 - \beta)(1 - F_v). \tag{9}$$

This metric is currently implemented only for circles. Detected arcs are ignored, and lack of properly detected arcs is not penalized.

4.2 Results

The 10 original and 800 noisy images were processed by each of the five circle detection algorithms. The match between the estimated results and the ground truth was calculated with the two methods described in Section 4.1. For each noise level and image combination the average across the noise instances was calculated.

Multiple noise instances were used for each image and noise level (10 instances for edge noise and 5 instances for pepper noise) to reduce the chance of specific noise pixels significantly affecting the results. Figure 5 shows the standard deviation measured across the noise instances and then averaged across the 10 different images. The standard deviation of the performance measurements was very low between noise instances. The different noise instances have a smaller effect for the pepper noise.



(a) (b)

Fig. 5: The standard deviation of VRI across the (a) 10 edge spread instances and (b) 5 pepper noise instance averaged over 10 images.

Edge noise results: The average VRI for both metrics across the 10 instances and 10 images for each algorithm was calculated for each noise level. The results are shown in Figure 6. As expected, the performance of the algorithms decreased

as the amount of noise was increased, but for most algorithms not in as extreme a fashion as expected. The Hough 3D algorithm was overall the best performer even though it did not detect arcs. The Qgar circle detection algorithm had the highest performance scores at low noise levels. The EDCircles algorithm was least sensitive to the noise, maintaining a VRI score of 0.7 for the ArcEval metric and a VRI score of 0.9 with the circle overlap method. The OpenCV Hough and the Jia algorithm scored very low with the Liu–Dori metrics because of small radius estimate errors resulting in strokes not physically overlapping. While they did not perform strongly with the circle overlap metric, it can be seen that they do detect some circles with reasonable accuracy.

Pepper noise results: The average VRI across the 5 pepper noise instances and 10 images is shown in Figure 7. A similar trend is seen relative to increased pepper noise level as was seen for edge noise. Hough 3D is the strongest and Qgar is most affected by noise. The algorithms were not affected as much by the pepper noise as they were by the edge noise.

Figure 7(a) also includes the results for the Liu–Dori VRI metric from their paper [13]. It has the same pepper noise levels, but also includes salt noise and is on different image content. It gives some level of comparison as their algorithm is not available to test here.

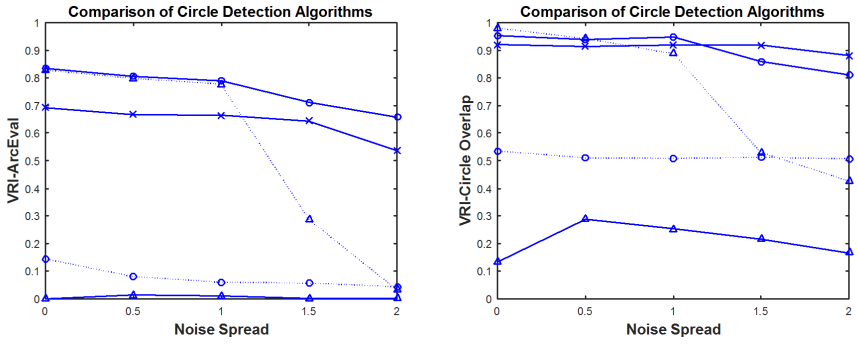
Full circles only: The OpenCV Hough and the Jia method were only designed to detect full circles. While EDCircles is capable of detecting arcs, their online service only returns circles and ellipses. Therefore to evaluate their performance, all the arcs were removed from the ground truth vec files for consideration by the Liu-Dori evaluation metric. Those results are shown in Figure 8. The results have the same shape with and without the arcs. The arcs would have added a false alarm quantity to all images the same across all noise levels appearing as a bias. Their removal reduced that in those algorithms.

Samples of detection results: In Figure 9 the arcs and circles identified by two of the algorithms can be seen. The Qgar algorithm is the only one that detects arcs. The Jia algorithm sometimes detects arcs as circles, but has many false alarms. Thus the difference in the VRI scores for the Liu–Dori metric versus the circle overlap metrics can be better understood.

5 Conclusion

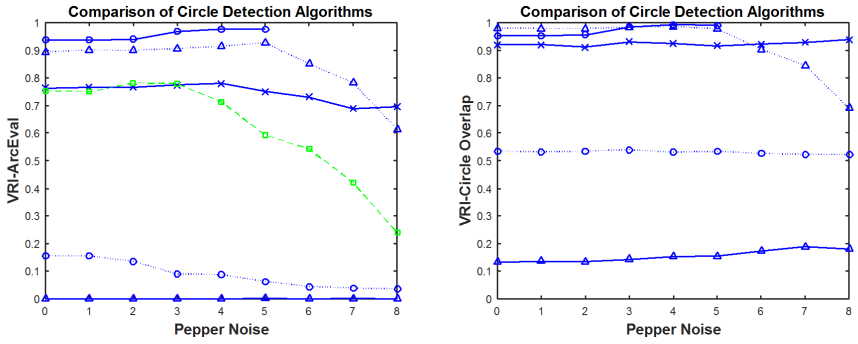
We have expanded the existing generally admitted framework for evaluating the robustness of circle and arc detection algorithms to different types of noise and have introduced a complementary performance metric based on circle overlap.

The main contribution of this paper is that it provides a finer way to compare algorithms, or to evaluate an individual individually to assess its suitability under



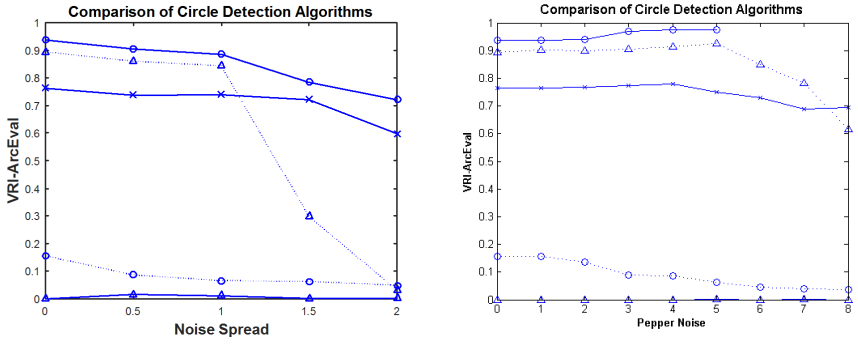
(a) (b)

Fig. 6: The average VRI across the 10 instances and 10 images for edge noise (a) Liu–Dori metric (b) Circle overlap metric. The legend is as in Figure 5.



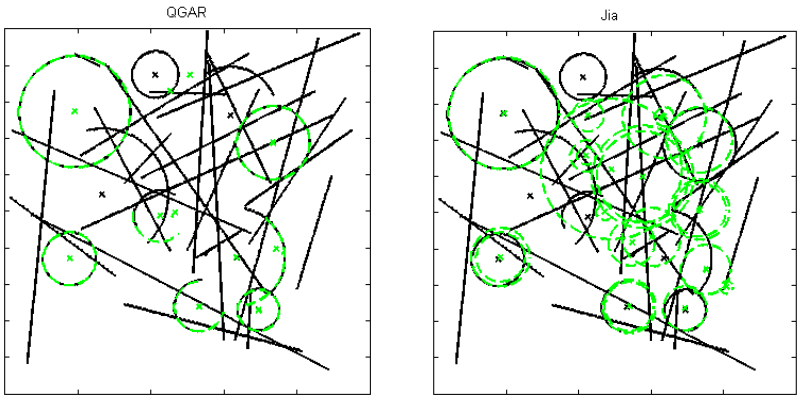
(a) (b)

Fig. 7: The average VRI across the 5 instances and 10 images for pepper noise (a) Liu–Dori metric (b) Circle overlap metric. The legend is as in Figure 5. In Figure (a) the dashed line with square symbols is results from paper [13]



(a) (b)

Fig. 8: The average Liu–Dori VRI across the 10 images for (a) edge and (b) pepper noise when arcs are eliminated from the ground truth. The legend is as in Figure 5.



(a) (b)

Fig. 9: The original image and the detected results for (a) Qgar and (b) Jia. Image 8, NS=0.5, noise instance 2.

well defined operational conditions, without the cost of manually annotating scanned documents.

We have provided two data sets, one for edge noise and one for pepper noise, and the source code for generating new sets is equally publicly available.

From our experiments on various state-of-the-art circle detection algorithms we can conclude that:

- Our noise model produces very consistent performance behaviours over all evaluated models.
- Our new evaluation metric, based on circle overlap surfaces, provides performance readings that are consistent with the usual Liu-Dori metrics. However, results are not identical, and underscore that the choice of metrics is important in regard to the application context.
- Reproducibility and traceability of published results remains a difficult issue. Not all contacted authors were willing or capable of providing source code or binaries of their published work. Sometimes provided code was not consistent with the claimed results, or needed debugging. Reprogramming from the published algorithm description was also, in one case, challenging.
  As a result, there are only 2 cases where we are certain to test against the actual, previously published algorithms.

6 Future Work

Future work could include variations in the image deformations. Elongation of the circles to gradually make them more elliptical will show the robustness of the algorithm, but introduces an interpretative bias, as it will require us to define the "best" circle fitting an ellipse.

The number of extraneous lines can be varied. There are currently 25 lines interfering with the circles and arcs. For some algorithms this is not a problem. For other algorithms, intersecting lines significantly reduce performance. Analyzing how the performance is degraded by extraneous lines is interesting, as is the detection of filled circles versus the current non-filled circles.

Furthermore, in some of our experiments we have observed unexpected behaviour of the Liu–Dori metric. We will investigate this matter further to determine whether this is an inherent shortcoming of the method, or has some other cause.

Integrating the image generation and the performance evaluation software into the DAE [9] platform will allow all researchers to evaluate their algorithms under this framework and will allow them to duplicate the results in this paper relative to those algorithms.

Acknowledgment

The authors would like to thank *Télécom Nancy* students G. Humbert and Y. Jardin for their preliminary work, and all authors who were solicited and who kindly provided their source code, binary code or other contribution allowing us to fully evaluate their methods. E. Barney Smith was funded under the "*Chercheur d'excellence*" program by the *Région Lorraine*.

References

1. Akinlar, C., Topal, C.: EDLines: A real-time line segment detector with a false detection control. Pattern Recognition Letters 32(13), 1633–1642 (2011)
2. Akinlar, C., Topal, C.: EDCircles: A real-time circle detector with a false detection control. Pattern Recognition 46, 725–740 (2013)
3. Akinlar, C., Topal, C.: EDCircles Web Interface (2015 (accessed 6 May 2015)), `http://ceng.anadolu.edu.tr/CV/EDCircles/demo.aspx`
4. Baird, H.S.: Document image defect models. In: Proc. IAPR Workshop on Syntactic and Structural Pattern Recognition, in Murray Hill, NJ. pp. 13–15 (June 1990), reprinted: Structured Document Image Analysis, H.S.Baird, H. Bunke and K. Yamamoto(eds), Springer-Verlag, NY, 1992
5. Desolneux, A., Moisan, L., Morel, J.: From Gestalt theory to image analysis: A probabilisitc approach. Springer (200)
6. Hough, P.: Methods and means for recognizing complex patterns (1962)
7. Jia, L.Q., Peng, C.Z.: A new circle detection method based on parallel operator. In: 2012 International Conference on Machine Learning and Cybernetics (ICMLC),. vol. 3, pp. 1085–1090 (2012)
8. Jia, L.Q., Peng, C.Z., Liu, H.M., heng Wang, Z.: A fast randomized circle detection algorithm. In: 2011 4th International Congress on Image and Signal Processing (CISP). vol. 2, pp. 820–823 (2011)
9. Lamiroy, B., Lopresti, D.: An open architecture for end-to-end document analysis benchmarking. In: Document Analysis and Recognition (ICDAR), 2011 International Conference on. pp. 42–47 (Sept 2011)

10. Lamiroy, B.: Interpretation, Evaluation and the Semantic Gap ... What if we Were on a Side-Track? In: Lamiroy, B., Ogier, J.M. (eds.) 10th IAPR International Workshop on Graphics Recognition, GREC 2013. vol. 8746, pp. 213–226. Springer, Bethlehem, PA, United States (Aug 2013)
11. Lamiroy, B., Guebbas, Y.: Robust and precise circular arc detection. In: 8th International Workshop on Graphics Recognition. Achievements, Challenges, and Evolution, GREC 2009. Lecture Notes in Computer Science, vol. 6020, pp. 49–60. Springer-Verlag, La Rochelle, France (July 2010)
12. Liu, W., Dori, D.: A protocol for performance evaluation of line detection algorithms. Machine Vision and Applications 9, 240–250 (1997)
13. Liu, W., Zhai, J., Dori, D., Long, T.: System for performance evaluation of arc segmentation algorithms. In: Proc. CVPR Workshop Empirical Evaluation in Computer Vision (2001)
14. McGillivary, C., Hale, C., Barney Smith, E.H.: Edge noise in document images. In: Proceedings of the 3rd Workshop on Analytics for Noisy Unstructured Text Data. pp. 17–24 (2009)
15. Nixon, M., Aguado, A.: Feature Extraction & Image Processing. Academic Press, 2 edn. (2008)
16. OpenCV: Hough Circle Transform (2015 (accessed 13 May 2015)), http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/hough_circle/hough_circle.html
17. Wolfram MathWorld: Circle-Circle Intersection (2015 (accessed 13 May 2015)), http://mathworld.wolfram.com/Circle-CircleIntersection.html