Pattern Analysis and Applications
https://doi.org/10.1007/s10044-021-01007-6

THEORETICAL ADVANCES

# Multiple circle detection in images: a simple evolutionary algorithm approach and a new benchmark of images

**Miguel R. González[1] · Miguel E. Martínez[1] · María Cosío-León[2] · Humberto Cervantes[1] · Carlos A. Brizuela[3]**

Received: 29 May 2020 / Accepted: 7 July 2021


**Abstract**
The circle detection problem focuses on finding all circle shapes within a given image. In fact, circle detection has several applications in real-life problems arising in agriculture, ophthalmology, and oceanography, among others. Despite many approaches having been proposed to deal with this problem, our work is motivated by two main issues: (1) the limitation of a recently proposed evolutionary algorithm and (2) the lack of benchmark images to fairly compare current approaches. To address the first issue, we introduce an effective evolutionary algorithm with a pre-processing noise reduction step. The proposed evolutionary algorithm's goal is to match several randomly generated circles with a point cloud extracted from an edge map of the original image. These circles are individuals in the population where the fittest one in the last generation is a detected circle. Henceforth, by removing the points corresponding to such circle and repeating the process, all circles within the image can be detected. We propose and make publicly available a set of synthetic, hand-drawn, and real images with different features to address the second issue. To assess our approach's performance, we apply it to the set of proposed images that include challenging features. Experimental results show that our method is competitive compared with the well-known Circle Hough Transform and as well as with EDCircles.

**Keywords** Circle detection · Genetic algorithm · Noise reduction · Point cloud · Edge map

## 1 Introduction

Circle detection within images has been extensively studied in the past decades [5, 9, 11, 18, 19, 32–35, 37] and it has proven vital for inferring geometric properties. Moreover, circle detection is a useful tool in such fields as agriculture to recognize and estimate the size of circular shaped fruits [22], astronomy to detect arc-like astronomical objects [15], ophthalmology to detect both iris and pupil [28], and oceanography to define the position and size of eddies [24]. Different techniques have been proven to accomplish circle detection. One of them is the Circle Hough Transform (CHT) [16]. Unfortunately, CHT needs a large 3D accumulation array and a considerable storage space, which impacts the computational cost [37]. Many other alternatives have been proposed to reduce this cost; for instance, Cuevas et al. [9] modeled circle detection as an optimization problem. They use a cost function to fit a set of points that form a circle previously computed from three selected points and a Learning Automaton (LA) as an adaptive decision-making method for selecting these points. Yuan et al. [35] used a power histogram and the circle power theorem, identifying edge pixels using a Canny edge detector to construct multiple power histograms from multiple random reference points derived from these edge points. They detected peaks in each histogram using the peak detection signal and cross-validating with the multiple power histogram to finally compute centers and radii for all detected circles.

Heuristic methods have also solved other optimization models. Cuevas et al. [8] defined a multi-modal optimization problem and used the Artificial Bee Colony (ABC) algorithm to find near-optimal solutions. They include the use of memory in the ABC process so that the algorithm can hold critical information and avoid getting stuck at local

✉ Carlos A. Brizuela
cbrizuel@cicese.mx

[1] Facultad de Ingeniería Arquitectura y Diseño, Universidad Autónoma de Baja California, Carretera Transpeninsular 3917, Ensenada, México

[2] Universidad Politécnica de Pachuca, Carretera Ciudad Sahagún-Pachuca Km. 20, Pachuca, México

[3] CICESE Research Center, Carretera Ensenada-Tijuana No. 3918, Ensenada, México

Published online: 17 July 2021

optimum and candidate circles are defined by three points. Genetic Algorithms (GA) have also been used to deal with this problem [32]. They have proven to be a useful tool to extract geometric primitives [27] mainly because their crossover operators can improve the extracted geometric objects' quality by appropriately combining them. Ayala-Ramirez et al. [32] used a GA whose primary input is the number of circles to be detected using the Sobel filter to create an edge image and generate a population of solutions that evolves and outputs the fittest candidate circle. The number of circles to be detected establishes the number of times this process is iterated.

One disadvantage of the GA method [32] is that it requires the number of circles to be found before running the algorithm, which may be a drawback when the end-user has no prior knowledge of the number of circles in the image, as when counting berries within grape bunches [25]. The first contribution of this work aims to compensate for this limitation. To this end, we designed a simple evolutionary algorithm that does not require the input of the exact number of circles it has to find and which is provided with a step by step explanation. Even though any population based heuristics needs to generate a circle (an individual) and assess its quality, so far, only two approaches have been proposed for doing it [31, 32]. We compared these approaches and selected the most suitable to use in the proposed algorithm.

However, despite the number of existing circles detection methods, we could not find any comparative research. One substantial limitation to perform such a comparison is the lack of a set of benchmark images. In this regard, our second contribution is a set of benchmark images made publicly available for future comparative studies.

The rest of the paper is organized as follows: Sect. 2 defines the problem and highlights the relevant work. Section 3 explains the proposed approach, while Sect. 4 presents the proposed benchmark of images. Section 5 details the experimental setup, as well as the results and metrics used to assess the accuracy of the method. Finally, Sect. 6 states the conclusions and provides some ideas for future research.

## 2 Problem statement

As mentioned above, the Circle Detection Problem (CDP) consists of locating all circular shapes within images. A circumference can be described as a set of center coordinates and a radius $(x_0, y_0, r)$ that can be computed through the circumference equation, using three points located on its perimeter. The formal circle representation used in this work was presented in [6] and used in [8, 9, 11, 20, 32, 33]. Figure 1 illustrates this representation. Given three points,

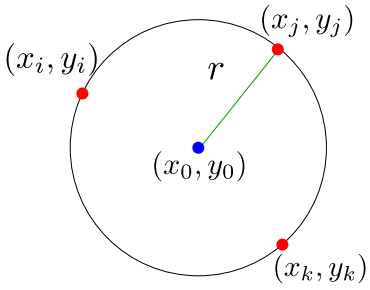



**Fig. 1** Illustration of three points passing through the circumference's perimeter

Eqs. 1, 2, and 3 show the computing process for center coordinates and the radius, respectively.

$$x_0 = \frac{\begin{vmatrix} x_j^2 + y_j^2 - (x_i^2 + y_i^2) & 2(y_j - y_i) \\ x_k^2 + y_k^2 - (x_i^2 + y_i^2) & 2(y_k - y_i) \end{vmatrix}}{4((x_j - x_i)(y_k - y_i) - (x_k - x_i)(y_j - y_i))} \tag{1}$$

$$y_0 = \frac{\begin{vmatrix} 2(x_j - x_i) & x_j^2 + y_j^2 - (x_i^2 + y_i^2) \\ 2(x_k - x_i) & x_k^2 + y_k^2 - (x_i^2 + y_i^2) \end{vmatrix}}{4((x_j - x_i)(y_k - y_i) - (x_k - x_i)(y_j - y_i))} \tag{2}$$

$$r = \sqrt{(x_{i,j \text{ or } k} - x_0)^2 + (y_{i,j \text{ or } k} - y_0)^2} \tag{3}$$

An image can be represented as a matrix $\boldsymbol{A}$ with $N$ rows and $M$ columns, where each matrix element corresponds to the intensity level of a pixel in a gray-scale image. First, since RGB images are commonly used as an input for this method, a gray-scale transformation is included. Let $\boldsymbol{A}_{gs}$ be a gray-scale image from image $\boldsymbol{A}$ by using the following equation [10]:

$$\boldsymbol{A}_{\mathbf{gs}} = 0.299\boldsymbol{A}_R + 0.587\boldsymbol{A}_G + 0.114\boldsymbol{A}_B \tag{4}$$

Second, a noise reduction stage can be introduced by applying a blur filter, e.g. median filter [29]; let $\boldsymbol{A}_{\text{nr}}$ be this filtered image. Finally, the edge map can be computed via an edge detector such as the Canny edge detector [4]; we can formally define an edge map as:

$$\boldsymbol{E}(i,j) = \begin{cases} 1, & \text{if } \boldsymbol{A}_{\text{nr}}(i,j) \text{ is an edge pixel} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

Summarizing, we can see the extracted edge map $\boldsymbol{E}$ from the input image $\boldsymbol{A}$ as a point cloud. The main objective is to find all circular shapes within this point cloud by fitting as many points as possible to each circle found. Notice that proper detection of circles in a CDP strongly depends on the quality of the edge map, which, in our case, is obtained by

applying a blur filter. The main characteristics of previous works related to CDP are described in the following section.

## 2.1 Relevant work

Circle Detection methods can be broadly classified as deterministic and stochastic. Regarding the deterministic category, Zhou and He [37] presented a method based on the Hough transform, with the main goal of decreasing the candidate circles' storage space by using a spatial decomposition approach with a vector quantization representation. Chauris et al. [5] presented an extension of the 1D wavelet transform and used Fourier analysis to detect the circles. In the same category, Akinlar and Topal [3] presented a circle-ellipse detector based on an arc detection and join procedure along with a validation stage from the Helmholtz principle; this method is named edge drawing circles (EDCircles). On the other hand, an example of a stochastic method is the work of Lopez and Cuevas [20], where a metaheuristic (teaching-learning based optimization or TLBO) method is implemented. The works in [8, 9, 11, 32] exploit the advantage of such methods. Table 1 shows a comparison among different approaches to this particular problem. This table's structure (column-wise) is as follows: main feature, proposed method, image pre-processing method, sub-pixel validation, and accuracy percentage reported (Not present = NP). The last column indicates the references. Note that the column "image pre-processing" shows that some methods [5, 8, 35] do not present a noise-reduction stage and handle the noise within the method. Also, note that one method [5] does not use the edge-map as input for the candidate circle search. Additionally, the "Sub-pixel validation" column shows that works such as those in [11, 20, 32, 33, 36] present a sub-pixel validation process.

GAs have been applied to a wide variety of image processing problems [1, 13, 32], e.g., Gudmundsson et al. [13] used a GA for edge detection, comparing its results with those obtained from using a simulated annealing approach

**Table 1** Characteristics of some works on circle detection methods

| Feature | Method | Image pre-processing | Sub-pixel validation | Accuracy | Year | Reference |
|---|---|---|---|---|---|---|
| Deterministic | | | | | | |
| Spatial decomposition into subimages using a Vector Quantization Algorithm. | Hough Transform. | Canny edge detection. | × | NP | 2016 | [37] |
| Power histogram, circle power theorem. | Cross-validation of N power histogram, computing a circle parameters. | Canny edge detector. | × | NP | 2015 | [35] |
| Avoid false detection and novel edge segmentation. | A least- square fitting approach. | Gaussian smoothing, propose a edge detection method | × | 80–100 % | 2014 | [19] |
| Circle and arc fitting | Gradient-direction edge clustering and direct least-square fitting. | Gaussian filter, Canny edge detection. | ✓ | 93–100 % | 2013 | [33] |
| Circle and ellipse detection | Arc join and detection | Edge drawing parameter free [2] | × | NP | 2012 | [3] |
| Characteristics finite frequency of circles. | Extension of 1D wavelets to 2D: circlet transform. | Fast Fourier Transform. | × | NP | 2011 | [5] |
| **Stochastic** | | | | | | |
| Use contours instead of edge positions. | Use a teaching- learning based optimization algorithm. | Canny edge detector. | ✓ | 48–100 % | 2018 | [20] |
| Use gradient orientation and contours. | Use an isosceles triangles identification whitin a circle shape algotihm. | Canny edge detector. | ✓ | NP | 2016 | [36] |
| Based on 5% of edge pixels stored, and 2 random methods. | LA as individual selection for GA process. | Canny edge detector. | × | 97–100 % | 2012 | [9] |
| Combine PSO,GA and chaotic dynamics. | Evolutionary method that involves positions updates and use of GA operators. | Unspecified | ✓ | 100% | 2012 | [11] |
| Eight adjacent points, compute the gradient pixel. | Lines gradient intersection. | Gaussian template, Canny edge detector. | × | NP | 2011 | [18] |
| Artificial Bee Colony. | Circumference equation for fitness evaluation. | Canny edge detector. | × | 98–100% | 2011 | [8] |
| Genetic Algorithm. | Circumference equation for fitness evaluation. | Sobel filter | ✓ | 92–100% | 2006 | [32] |

and applying this approach to magnetic resonance images. Abdel-Khalek et al. [1] attempted to achieve a threshold segmentation based on Tsallis and Renyi entropies [26, 30] and implemented a GA to find a segmentation that maximizes such entropies.

The approach proposed here is close to the one used by Ayala-Ramirez's [32], differing in the following aspects that we added:

- A noise-reduction pre-processing step.
- A comparison of two methods: MCA [8, 9, 20, 31] and TP [11, 32] to generate test points that validate the identified circle's quality.
- A Canny edge detector implementation from the OpenCV library is introduced instead of the Sobel edge detector implemented in Matlab Ayala-Ramirez et al. [32] used.
- An in-house version of a simple GA with a detailed explanation of each of its components is introduced.
- Even though Ayala-Ramirez's approach is a multiple circle detector, the number of circles to be found is an input parameter. Our method does not need this parameter to detect multiple circles properly.
- Data availability: We make all images publicly available so that other researchers can compare their results with the ones presented here. The set of benchmark images is available at https://github.com/ricglez89/circles_dataset.

The following section provides the details of the proposed method.

## 3 Circle detection by a simple genetic algorithm (CDSGA)

We propose a method based on a Simple Genetic Algorithm (SGA) for circle detection problem that works as follows:

1. We acquired an edge image $\boldsymbol{E}$ (using the OpenCV implementation of the Canny edge detector with both input parameters set to 50) taken from an input RGB image $\boldsymbol{A}$. This input image is turned into gray-scale space using Eq. 4 and applying a smoothing filter using the implementation of medianBlur() from OpenCV, with a kernel size $N_k = 3$ since large size kernels strongly affect the quality of the resulting image.
2. An array was created to store positions of every edge pixel in the form of $(x, y)$. Let $V$ be such array with $N_e$ coordinates containing every edge pixel ($\boldsymbol{E}(i,j)$) found in image $\boldsymbol{A}$, *i.e.* $V = \{(x_0, y_0), (x_1, y_1), ..., (x_{N_e-1}, y_{N_e-1})\}$.
3. A population of candidate circles was created. Each $S_i$ (a candidate circle) is encoded as a concatenation of three points randomly taken from $V$, and by computing the center $(x, y)$ and radius $r$ with Eqs. 1, 2 and 3, the individual is stored for the evolution process.
4. The created population undergoes a standard evolution process for a Genetic Algorithm, and it stops after the termination criterion expires.
5. We selected the fittest circle $S^* = [(x^*, y^*), r^*]$ found in the last generation of the SGA's evolving process.
6. We erased from $V$ every coordinate passing through $S^*$ an repeated the process from step 3 until the stopping criterion was satisfied or $V$ was empty.

We propose a GA to solve the circle detection problem because this population based paradigm exploits a combination operator that merges two individuals (circles) to generate a new one. As shown in erlier work for geometric primitives extraction [27], this can help obtain the correct circle from two approximations (see appendix B from supplementary material).

### 3.1 Two approaches for circumference test points generation

In order to match a candidate circle $S$ generated by the algorithm with the image's edge pixels, we created a set of test points from a candidate circle $S = [(x, y), r]$. Let $T(S) = \{t_0(S), t_1(S), ..., t_{nte-1}(S)\}$ be this set, where $nte$ is the number of test points to create, and $t_i(S) = (x_i, y_i)$ a coordinate point generated by Eqs. 6 and 7, for every $i \in \{0, 1, ..., nte - 1\}$. This approach to generating test points was also used among the metaheuristics in [11, 32]. We call this test points generation approach TP.

$$x_i = x_0 + r \cos \frac{2\pi i}{nte} \tag{6}$$

$$y_i = y_0 + r \sin \frac{2\pi i}{nte} \tag{7}$$

Another method used to create test points is the midpoint circle algorithm (MCA), which defines the required amount of points to draw a circle whiting a digital image. The MCA needs the following input parameters: a central coordinate $(x_0, y_0)$ and a radius value $r$. This algorithm computes discrete positions only over the first eighth of the circumference and mirrors to the rest of the eighths. That is to say, if an $(x, y)$ point is computed, the points $\{(y, x), (-y, x), (-x, y), (-x, -y), (-y, -x), (y, -x), (x, -y)\}$ are also computed. The MCA starts at the $(x_0 + r, 0)$ point and continues anticlockwise, ending at $(x, y)$ when $y > x$, showing that first eighth of the circumference has been visited. Once the first point $t_i$ is computed, then the next point is defined as follows [20]:

$$t_{i+1} = \begin{cases} (x_i, y_i + 1), & \text{If } (x_i - 0.5 + x_0)^2 + (y_i + 1 - y_0)^2 - r^2 \leq 0 \\ (x_i - 1, y_i + 1), & \text{Otherwise} \end{cases} \tag{8}$$

The MCA aims to determine the number of points that minimizes the distances between the discrete point and the continuous candidate circle. This method was used in [8, 9, 20].

Given a candidate circle $S = (x_0, y_0, r)$, its quality $ev(S)$ can be computed as follows:

$$ev(S) = \sum_{j=0}^{nte-1} P[t_j(S)] \tag{9}$$

where:

$$P[t_j(S)] = \begin{cases} 1, & \text{if the euclidean distance between } t_j \text{ and } V \text{ is smaller than } e \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

Remember that $V$ is an array where every edge pixel position is stored, and $e$ is a threshold that defines the maximum allowed distance for a pixel to be considered a neighbor to another. As mentioned in [21], to consider the computed test point $t_i(S)$ as correct, the $e$ value is fixed at $\sqrt{2}$. This value is the maximum distance between a test point in $T(S)$ and an edge pixel in $V$ to be considered neighbor and, therefore, a match.

### 3.2 Genetic algorithm

Genetic Algorithms (GAs) are general optimization techniques inspired by natural selection and the survival of the fittest. This metaheuristic was introduced by Holland in the 1960s [14]. The typical algorithm consists of the following main components [7]: (1) A population of individuals, where each individual represents a solution to the problem. (2) A way of computing how good the individual is (the fitness function). (3) A method for mixing fragments of good quality solutions in order to form new solutions enriched with those qualities (the recombination operator). (4) A mutation operator that helps to keep diversity in the population.

Using the three point description of a circumference, we can create a population of solutions (three points), using an SGA to evolve that population, and guided by the objective function, find a circle in the image.

We will now explain the SGA process in detail:

– *Encoding*

Encode each individual as a binary concatenation of three point indexes $(i, j, k)$ in a $V$ array (see Fig. 2) where each index corresponds with a point in the position-array $V$. The example provided in Fig. 2 shows that, $i = 6, j = 12,$ and $k = 3$. Meaning that we will select three points from $V$ indexed by $(i, j, k)$ to define a candidate circle $S$. Representing a circle based on three points was also used in [8, 9, 11, 20, 32]. Notice that this representation has a redundancy of 6 because every permutation of the three points will lead to the same edge pixels. However, this redundancy does not scale with the number of edge pixels and can be easily compensated with an appropriate population size.

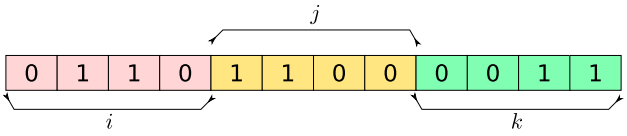



**Fig. 2** An example chromosome for representing a circumference. The three indexes: $i = 6, j = 12,$ and $k = 3$

– *Initialization of population*

We had an input image $\boldsymbol{A}$ with $N$ rows and $M$ columns and assumed that the circles to be detected were entirely contained within the image; thus, no circle with a diameter greater than the smallest value of either $M$ or $N$ would be detected. We fixed the maximum diameter $d_{\max}$ as follows:

$$d_{\max} = \begin{cases} N, & \text{if } N \leq M \\ M, & \text{otherwise} \end{cases} \tag{11}$$

Population initialization process:

1. Generate a random number $i$ from 0 to $N_e - 1$ (number of edge pixels).
2. Generate a random number $j$ from 0 to $N_e - 1$, until the euclidean distance between $V_i$ and $V_j$ is less than or equal to $d_{\max}$.
3. Generate a random number $k$ from 0 to $N_e - 1$, until the euclidean distance between $V_k$ and $V_j$ is less than or equal to $d_{\max}$, and the distance between $V_k$ and $V_i$ is also less than or equal to $d_{\max}$.
4. $i \neq j \neq k$ /* Do not select repeated points */.
5. Concatenate in binary code for $i$, $j$, and $k$.

We did this to generate every individual in the population.

– *Fitness function*

For the fitness function evaluation, we used $i, j,$ and $k$, taking the values from $V$ in $V_i, V_j, V_k$ and computing a candidate circle using Eqs. 1, 2, and 3 to obtain the circle $S$ and create the test points $T$ (see the beginning of Sect. 3.1), the fitness function is computed as follows:

$$F(S_i) = \frac{\sum_{j=0}^{\text{nte}-1} P[t_j(S_i)]}{\text{nte}} \tag{12}$$

Thus, of all individual (circles) $S_i$, the fittest one ($S^* = [(x^*, y^*), r^*] = \text{argmax}F(S_i)$) at the end of the algorithm's run is a detected circle. The fitness function given by (12) was also used in [8, 9, 11, 20, 32]. Equation (12) evaluates how well the generated circle fits into the edge map **E**. Note that $F$ is in the range [0, 1].

– *Selection and variation operators*

The roulette-wheel selection and generational replacement strategies were used. The standard 1-point crossover along with a bit-wise mutation operator was applied. These operators were also used in [32].

### 3.3 Deleting coordinates

We took the fittest individual $S^*$ in this evolving process and computed the euclidean distance between the circle's center $(x^*, y^*)$ and every point of the array of edge pixels $V$. If the distance is smaller than $r^* + \varepsilon$ (a bias value) and greater than $r^* - \varepsilon$, that point is erased from $V$. This process reduces the size of $V$ and therefore the computation time in the next search process. We applied this entire process iteratively in order to find multiple circles. Although not clearly stated, this procedure was probably also used in [20, 32]. Table 2 shows the stopping criteria. The first criterion is a fitness threshold of the fittest individual in the last generation, and if the fitness of this individual is less than or equal to the threshold, it will imply that the remaining set of points are not enough to form a circle. The second criterion establishes that the maximum number of circles that ensure the algorithm will not run forever if the number of circles to be found is too large. Finally, the algorithm also stops if there are no more edge points in $V$, which happens when all circles are detected.

**Table 2** Input parameters for SGA taken from Ayala-Ramirez et al. [32]

| Parameter | Value |
|---|---|
| Population size | 70 |
| Crossover rate | 0.4 |
| Mutation rate | 0.01 |
| Selection method | Roulette wheel |
| Crossover method | 1-point crossover |
| *Stopping criteria* | |
| Fitness threshold | 0.5 |
| Maximum number of circles | 1000 |
| Size of $V$ | 0 |

## 4 Proposed benchmark images

One of the motivations of this work is to overcome the lack of a benchmark of images. To this end, we make publicly available a set of images that capture the most relevant features of the images used in previous methods for circle detection. They fall into three main categories, synthetic, hand-drawn, and real. The details of each category will be explained in the following sections.

### 4.1 Synthetic images

In order to evaluate **CDSGA**'s performance, we generated a set of ten synthetic images. The computation time was measured by running the implemented algorithm on images containing one to ten circles of equal radius (75 px); the size of the images used is 512×512 px and 512×1024 px. A second set of 19 images containing ten circles with added Salt and Pepper noise (from 1% to 19% of the image) was created (see Fig. 18) to assess the algorithm's robustness to noise. The imnoise() function from GNU Octave was used to add noise. The code is available at https://github.com/ricglez89/circles_dataset. The third set of 10 synthetic images created contains challenging features such as concentric circles, overlapping circles, embedded circles, low-contrast circles, circles of different sizes, and multiple shapes. The synthetic images sets were created to emulate extreme feature extraction conditions to evaluate the robustness of **CDSGA**. All synthetic images were generated with Inkscape [17].

### 4.2 Hand-drawn images

A set of two hand-drawn images (both with a size of 508×508 px) containing irregular circular and non-circular shapes was also included in order to assess the robustness of the proposed method, see Fig. 10.

### 4.3 Real images

A set of 5 real images captured with a Nikon D5100 Digital Single Lens Reflex (DSLR) camera, 4 of them in 512×512 px in size, and one size 1016×508 px containing both circular and non-circular shapes was also included. The set of real images also included non-uniform contrast. See Figs. 11 and 13.

### 4.4 External images dataset

The authors in [36] kindly provided a dataset containing 6 images: coins (273×274 px), plates (335×316 px), eye (350×

343 px), speaker (287×291 px), balls (303×299 px) and a watch (302×347 px). Some of these images were also used in other works [3, 20, 36].

## 5 Experimental setup and results

We implemented the proposed approach in C++ (version 7.4.0), using OpenCV (version 3.4.4) for image processing. The code was run on a workstation computer with the following specifications: Intel Xeon CPU W-2104 CPU 3.20GHz × 4, 16 Gb RAM. Table 2 shows the parameters established to evolve the population. Population size, selection method and type of crossover and mutation held the same value as those used in the work of Ayala-Ramirez et al. [32]. The parameters were tuned through an exhaustive grid search for crossover values in the range of [0.3, 1.0] in steps of 0.1 and in the [0.01, 0.1] range with 0.01 steps for mutation, i.e. 80 different combinations were tested. Pairwise Wilcoxon statistical tests were performed to compare the combination with the best performance against the performances of any other combination in the grid. The algorithm runs for 50 generations for each detected circle instead of the 500 generations used in [32]. This number was selected after running the algorithm on three different types of image (synthetic, hand-drawn and real) 30 times, 100 generations each. In every case, the average fitness converges at generation 25. We doubled the number to 50 generations as a safety margin. No significant improvement with further generations was observed (see appendix B in the supplementary material for details). In addition to these input parameters, we proposed the stopping criteria shown in Table 2. They are: a large number of circles to be found, the size of $V = 0$, or when the fitness of the fittest individual reaches a given threshold. The fitness threshold was set to 0.5, this value can recover circles with 55% of missing perimeter points (pixels). Details for this threshold and the tuning process are presented in a supplementary material in the repository. Circle Detection by a Simple Genetic Algorithm (**CDSGA**) takes an RGB image as input without depending on any other input parameter. Several images are used in order to evaluate the performance of **CDSGA**. The details are explained below. All images used in this research are available at https://github.com/ricglez89/circles_dataset.

We also performed a sub-pixel validation [21] as a false positive control stage. This validation involved computing the test points $T$ of the detected circles $S_i$ once the **CDSGA** process was done. The *nte* is the cardinality of $T$, and since the radius $r$ is in pixels, it suffices to use the formula $nte = \text{round}(2\pi r)$. Let $D(t_j)$ be the minimum euclidean distance from $t_j$ to $V$. The mean $\mu_d$ and standard deviation $\sigma_d$ of $D(t_j)$ of all $j$'s can be computed. If $(\mu_d + \sigma_d) > e$ the detected circle is not considered as a perfect circle as described in [21], and then it is rejected. The user can adjust this threshold $e$ to include circles with different levels of distorted shape or discontinuity. In our case, the values of $e$ were fixed after a trial and error process. The specific values for $e$ are provided in each experiment.

### 5.1 Computing circumference test points

The detection process depends on the circle generation stage. For this reason, we performed a comparison between the two methods of computing circumference test points (see Sect. 3.1) in order to assess which was the most suitable for our proposal. The first method evaluated was the one explained in Sect. 3.1, called test points (TP). The TP method requires three input parameters: a central coordinate $(x_0, y_0)$, a radius value $r$, and the number of points (*nte*) to be computed. TP creates a fixed size set of equally spaced points, which establishes the number of points based on the radius length. The second method, known as MCA, was also explained in Sect. 3.1.

#### 5.1.1 Comparison between TP and MCA

Figure 3 shows an example of how TP and MCA methods sample a generated circle. Notice that $\mu_d$ for MCA is more significant than for TP. This difference is because, in MCA, the distances between a created point and a real one is either 1 or 0. Whereas in TP, this distance varies because of the continuous position of the generated coordinates. See Eqs. (6) and (7).

The *nte* value (number of test points to be created) was validated to obtain fully formed circles. We created a set of 10 synthetic images with increasing radii ($r$) circles ($r = \{10, 20, 30, ..., 250\}$) in a 512×512 size image; the center position is the same for all circles ($x_0 = 255.5, y_0 = 255.5$). The goal is to compute the mean Euclidean distance between a test point and an edge pixel, with different values of *nte* by assuming that $nte = \text{round}(\alpha d\pi)$, where $\alpha$ is the fraction of the circumference points we are looking for and $d$ the diameter. Then *nte* will be the number of sampling points

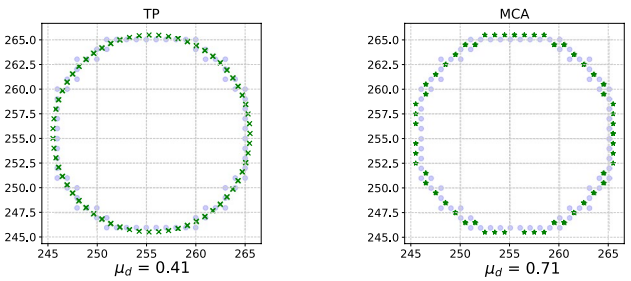



**Fig. 3** Comparison between a synthetic circle (o) and both test points generation methods; TP (x) and MCA (*)

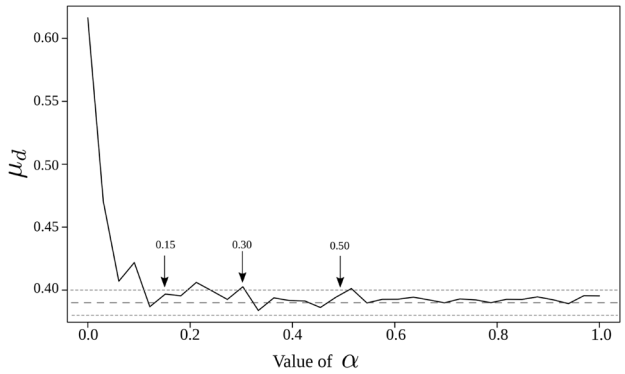



**Fig. 4** Mean distance $\mu_d$ as a function of $\alpha$.

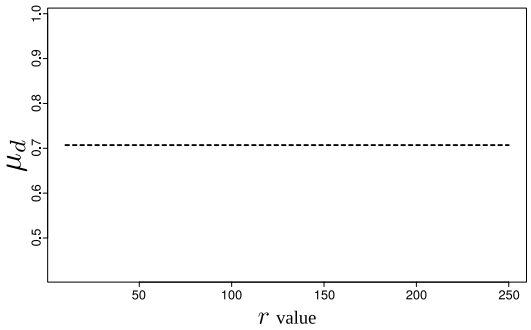



**Fig. 5** Mean distance $\mu_d$ as a function of $r$.

of the perimeter for a specific diameter. We computed the mean distance $\mu_d$ from a test point $t_i$ to the closest edge point in $V$, measured in pixels, by using the following equation:

$$\mu_d = \frac{\sum_{i=0}^{nte-1} \min(\text{euclideanDistance}(V, t_i))}{nte} \tag{13}$$

First, we computed the mean distance $\mu_d$ for each method, as Fig. 4 shows for TP and Fig. 5 for MCA. Fig. 4 presents the mean distance $\mu_d$ as a function of $\alpha$, the result of varying $\alpha$ in the interval [0.1, 1], and averaging the distance over $r$ in [10, 250] for each value of $\alpha$ in the set $\{0.1, 0.2, ..., 1.\}$. We can see that an $\alpha$ value of 0.5 or greater yields an acceptable mean distance. From this experiment we select $\alpha = 0.5$ if $r < 100px$ and to avoid large values of $nte$ we select $\alpha = 0.3$ if $r \geq 100px$. On the other hand, Fig. 5 shows the mean distance $\mu_d$ between MCA and $r$ (also in [10, 250]). Note that for MCA the mean distance $\mu_d$ is the same for each value of $r$.

Computation time is measured as a function of population size (from 10 to 100 individuals) to extend the comparison of both methods and revise the fitness computation behavior. The experiment consists of generating a population and calculating fitness for each individual without involving any evolution process. Figure 6-right shows the multiple box-plot of 30 measures for each population size. We can observe, that as expected from the results shown in Fig. 6-left, the TP method is faster than the MCA. Note in Fig. 6-left for MCA (blue "o") it is not possible to control the amount of generated points, as it is done in the TP method (green "*": $alpha = 1$, green "x": $alpha = 0.5$) by decreasing the value of $\alpha$, and as Fig. 4 shows, $alpha = 0.5$ presents a $\mu_d = 0.4$, approximately. For this reason, we selected the TP approach for further experiments.

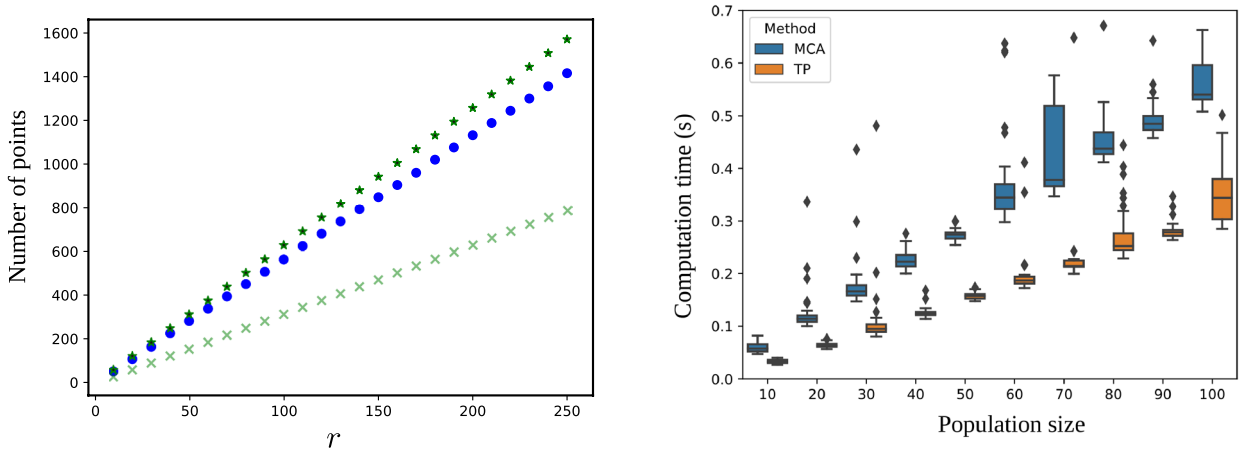



**Fig. 6** Left: Number of evaluation points as a function of $r$, TP with $\alpha = 1$ (*), MCA (o) and, TP with $\alpha = 0.5$ (x). Right: Computation time for fitness evaluation of each method as a function of population size. MCA (blue) TP (orange)

## 5.2 Results and discussion

### 5.2.1 Assessment of CDSGA

We use the first set of 10 synthetic images with equal radius circles by running the algorithm 30 times on each image. As expected, the average computation time increases as the number of circles contained in the images increases. Figure 7-left shows the average computation time as a function of the number of circles found. Figure 7-right also shows average computation time, which is a function of the number of edge pixels found. A second-order polynomial can model this plot. The primary purpose of using the multiple box-plots (Fig. 7-left) is to show the scatter of computation time values (on each run) as a function of the number of circles within the image, as it can be seen in each box-plot the scatter is kept close to the mean value. The time behavior curve from Fig. 7 shows a strong dependence on the number of edge pixels within the edge map $\boldsymbol{E}$ for the computation time of the circles to be found. In all cases, all circles were detected.

As mentioned in Sect. 5, we use a sub-pixel validation for reducing the number of false positives. This process is controlled by threshold $e$. Figure 8 shows an example of this process. The validated circles are shown as solid circles (green), whereas the excluded are shown as dashed circles (cyan).

A recovered circle is considered valid if $(\mu_d + \sigma_d) \leq e$, which will help avoid false positive circles. For instance, Fig. 8-1 column c) depicts a synthetic image in which the small dashed circle with $(\mu_d + \sigma_d) \leq e = \sqrt{2}$ is discarded. On the other hand, when it comes to a real image (Fig. 8-2), an $e = 4$ is used as a threshold to discard false positive circles, illustrated with dashed lines in column c).

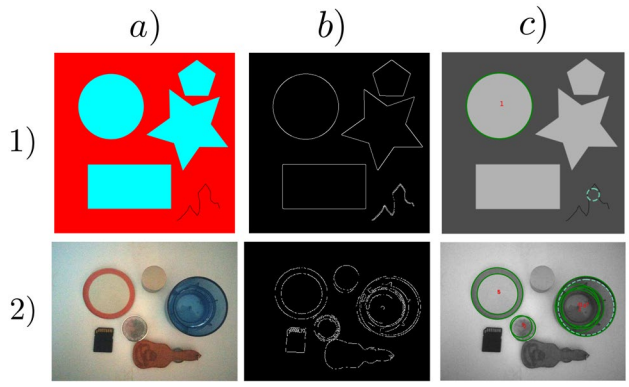



**Fig. 8** Example of sub-pixel validation; **a** Input image, **b** edge map, and **c** resulting image, green solid circles are the detected circles, cyan dashed circles are false positives eliminated with the sub-pixel validation. Row (**1**) A synthetic image, and row (**2**) a real image

### 5.2.2 Comparison with state-of-the-art methods

We created a ground truth set of images where the circles were manually annotated on every image from the proposed benchmark. Then we compared the results from the CDSGA with those of CHT and EDCircles methods. On the one hand, for the CHT method, we used the OpenCV HoughCircles() implementation. HoughCircles() needs nine input parameters, see [23] for details. As mentioned in [3], this method's availability and wide use in literature facilitate its comparison with other methods. However, CHT needs a proper setup for several input parameters. This work uses a trial-and-error process for each compared image. The input image used for CHT is the same noise-reduced image used in the CDSGA method. Input parameters used for every image are shown in supplementary

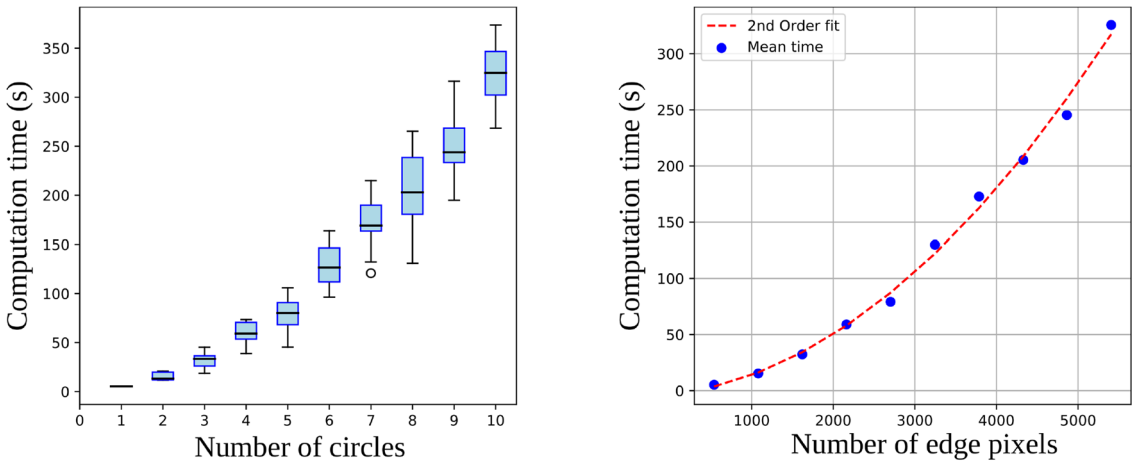



**Fig. 7** Left: Multiple box plot for the computation time as a function of the number of circles. Right: Computation time as a function of the number of edge pixels

material in Table D.1, and the rest of the parameters are left to their default configurations.

On the other hand, for the EDCircles method, we used the public implementation taken from the following repository: https://github.com/CihanTopal/ED_Lib; this method uses its own process to detect edges [2], and does not require any input parameters. We compared our approach with EDCircles not only because of its availability but also because of its outstanding reported performance. For every image compared, and due to its random component, we ran the CDSGA method 30 times and selected as output the circles with the lowest $\mu_d + \sigma_d$ in each image (see details in appendix C from the supplementary material).

To assess the relative performance of the methods regarding their accuracy to precisely recover the center and the radius for every circle, we propose to use the following metrics:

$$E_{\text{pos}} = \frac{|x_p - x_a| + |y_p - y_a|}{x_a + y_a} \tag{14}$$

$$E_{\text{rad}} = \frac{|r_p - r_a|}{r_a} \tag{15}$$

where $E_{\text{pos}}$ and $E_{\text{rad}}$ are normalized errors for the position and the radius, respectively; $(x_a, y_a, r_a)$ are the center and radius of the manually annotated circle, and $(x_p, y_p, r_p)$ are the center and radius of the predicted circle from the detector. Notice that, each metric is a normalized Manhattan distance between the detected circle and the ground truth circle. Additionally, we manually matched the detected circles to each method's output and compared the detected circle with the ground truth circle. We analyze these metrics for the synthetic, hand-drawn and real images.

*Synthetic images*

First, we analyzed the results for the synthetic images (see Sect. 4.1). Low-contrast and overlapped circles within the image present difficult detection conditions for the feature extraction method; however, our CDSGA could detect all the circles within every tested synthetic image. Figure 9 shows the results and the comparison among the three methods: EDCircles, CHT and CDSGA. In Fig. 9, the *Ground truth* column shows the input image, with an overlapped manually annotated circle and the edge map used for CHT and CDSGA methods. The columns *EDCircles, CHT*, and *CDSGA* present the results for each method. The green circles match with the ground truth circle, while the orange circles are the false positives.

Additionally, we present a bar chart for each metric, showing the results for every circle detected in each image. Figure 9-1 shows three concentric, low-contrast circles and one large circle; observing in detail its edge map, we can notice that the low-contrast circles are incomplete. For this image, the CDSGA was run with an $e = 8$, whereas for the others in this set, $e = \sqrt{2}$. It can be seen the three methods can locate the four manually annotated circles. Analyzing the *Position* bar chart in Fig. 9-1, we notice that the CDSGA method yields better results for circle number 2. However, when assessing the metric *Radius*, circle 4 is the one with the lowest error. Figure 9-2 shows that the CHT method detects only one circle of the 3 manually annotated, whereas EDCircles and CDSGA detect all of them. Each bar chart provides a direct comparison of each method for every circle. Note that if the circle is not displayed on each method's output image, the bar corresponding to this circle is not displayed either. Regardless, this does not mean that the error is equal to zero.

*Hand-drawn Images*

Regarding the hand-drawn images set (see Sect. 4.2), all methods could recover circular shapes. In Fig. 10-1, the CDSGA method yields the best results. However, EDCircles detected two ellipses, shown in solid red. For this reason and comparison purposes, we used the ellipse center as the center of the circumference and the best major- or minor-axis as the circumference radius. For Fig. 10-2, we observe the same situation, CDSGA yields the best results, but EDCircles detected one circle and two ellipses.

*Real images* Figures 11 and 13 shows the results yielded by the analysis of real images. These images shows concentric, low-contrast, and different size circles. Figure 11-1 presents 11 manually annotated circles. Seven, four, and seven of these circles were detected by EDCircles, CHT and CDSGA, respectively. A detailed examination of Fig. 11-1 reveals the reason for the false positive generation of CDSGA (see Fig. 12). This figure shows that the selected edge map points (rightmost figure) depict the circles in the center of the image. Also, circle 9 (Fig. 11-1) was detected only by EDCircles, whereas circle 11 was detected by the CDSGA only. None of the three methods detected the smallest, manually annotated circles (6–10) in Fig. 11-2, and the edge map shows that some of them are occluded. Figure 13-3 shows that EDCircles for circle 1 generated an ellipse as output. The center of the ellipse and its best axis were used for the comparison, as we did with the hand-drawn images. Figure 13-4 shows that the three methods detected all manually annotated circles, whereas Fig. 13-5 shows that EDCircles detected the largest number, with 8 out of the 14 manually annotated. In contrast, CHT and CDSGA detected circles 1 and 2, respectively. Figure 13-5 presents a complex case, as its corresponding edge map shows. The bars corresponding to not detected circles were not displayed in their bar charts (see Figs. 11-2 and 13-5).

*External images*

Figures 14 and 15 display the external image data set (see Sect. 4.4). The three methods detected all seven manually annotated circles (Fig. 14-1). Circle 4 was detected

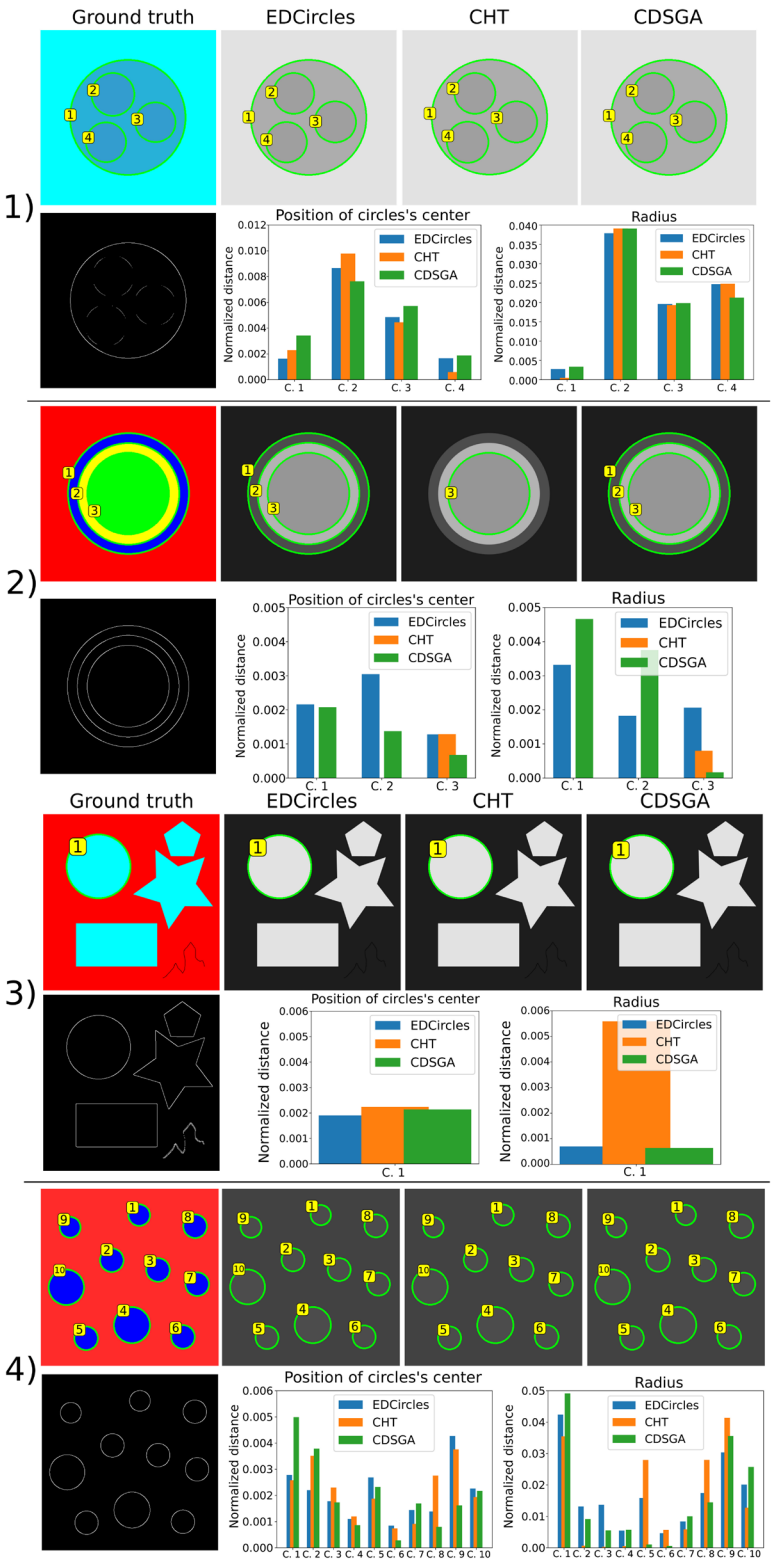



**Fig. 9** Results in synthetic images with the following features: concentric, overlapped, and multiple shapes; **Ground truth**: input image with manually annotated circles and its edge map. **EDCircles, CHT, CDSGA**: results from each method (for CDSGA $e = \sqrt{2}$ all images, except row 1 with $e = 8$). On each row a bar chart for each metric is presented

by CDSGA with the lowest error, whereas circle 6 was better detected by CHT. Figure 14-2 shows that CHT only detects four circles out of the eight manually annotated. In contrast, Fig. 14-3 shows that EDCircles present the lowest error for manually annotated circles, whereas CDSGA in circle 2 detects a circle with a slightly greater radius. In Fig. 15-4, we manually annotated 22 circles that yielded another complex image. The arcs formed by the bottom

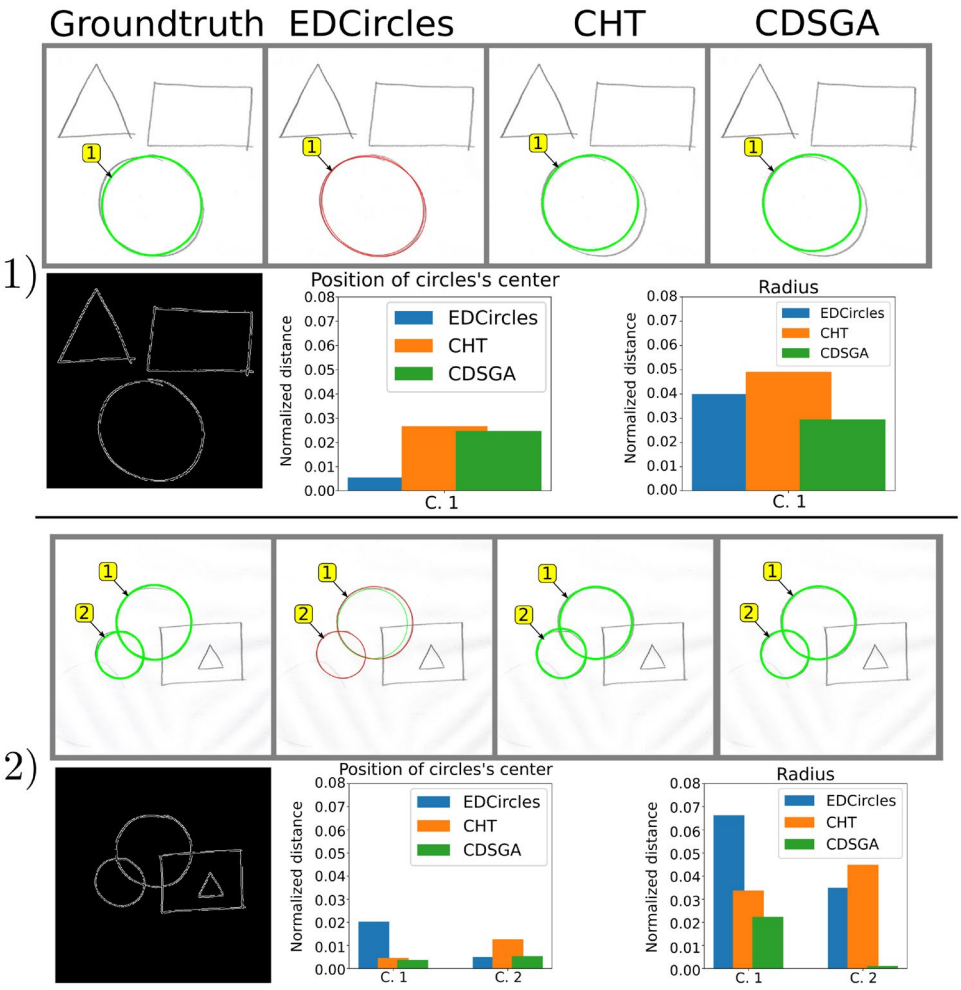



**Fig. 10** Hand-drawn images with overlapped circles and multiple shapes within. **Ground truth**: input image with overlapped manually annotated circles and their edge maps. **EDCircles, CHT, CDSGA**: results from each method (for CDSGA image 1) with $e = 14$ and image 2) with $e = 8$). On each row a bar chart for each metric is presented

speakers were not included. 13, 2 and 6 circles were detected by EDCircles, CHT and CDSGA, respectively; however, not detected circles were not included in these bar charts. In Fig. 15-5, circle 4, EDCircles presents the lowest error. Finally, for Fig. 15-6, circle 1, CHT provides the best results, whereas CDSGA for circle 3 is the one with the lowest error; note that CHT only detects one out of the three manually annotated circles.

Figure 16 presents a multiple box-plot of the performance metrics of all tested images. To verify whether the observed differences are statistically significant we followed a procedure proposed elsewhere [12]. First, we performed a normality test by applying the Shapiro-Wilk test, where the null hypothesis establishes that the results follow a normal or Gaussian distribution; Table 3 shows the results. We used a 99% confidence interval for the tests. The results in bold with * are the ones where the null hypothesis is rejected, i.e. where the results do not follow a normal distribution. Finally, a Wilcoxon test is performed on those results. In this test, the null hypothesis defines that the medians of the metrics are the same. Table 4 shows the test results.

The difference between the normalized distances of the position and radius of the detected circles is not statistically significant. This implies that the accuracy of the methods, when they recover a circle, is comparable.

Although the closest reference to our approach is the work proposed by Ayala-Ramirez et al. [32], we could not perform a direct quantitative assessment, due to the unavailability of the set of images they used and the lack of implementation details of their algorithm. Fortunately, some of the images used in [20] were shared with us by the authors in [36]; this allowed us to compare the number of recovered circles, which Table 5 shows. The first column shows the corresponding image number in Figs. 14 and 15. The second column presents the number of circles in the image, and the third column shows the number of circles identified by the TLBO method, as reported in [36], the fourth, fifth and sixth columns report the number of circles detected by EDCircles, CHT and CDSGA, respectively. These columns have two numbers; the first one represents the number of original circles detected, while the number in parenthesis shows the total number of circles

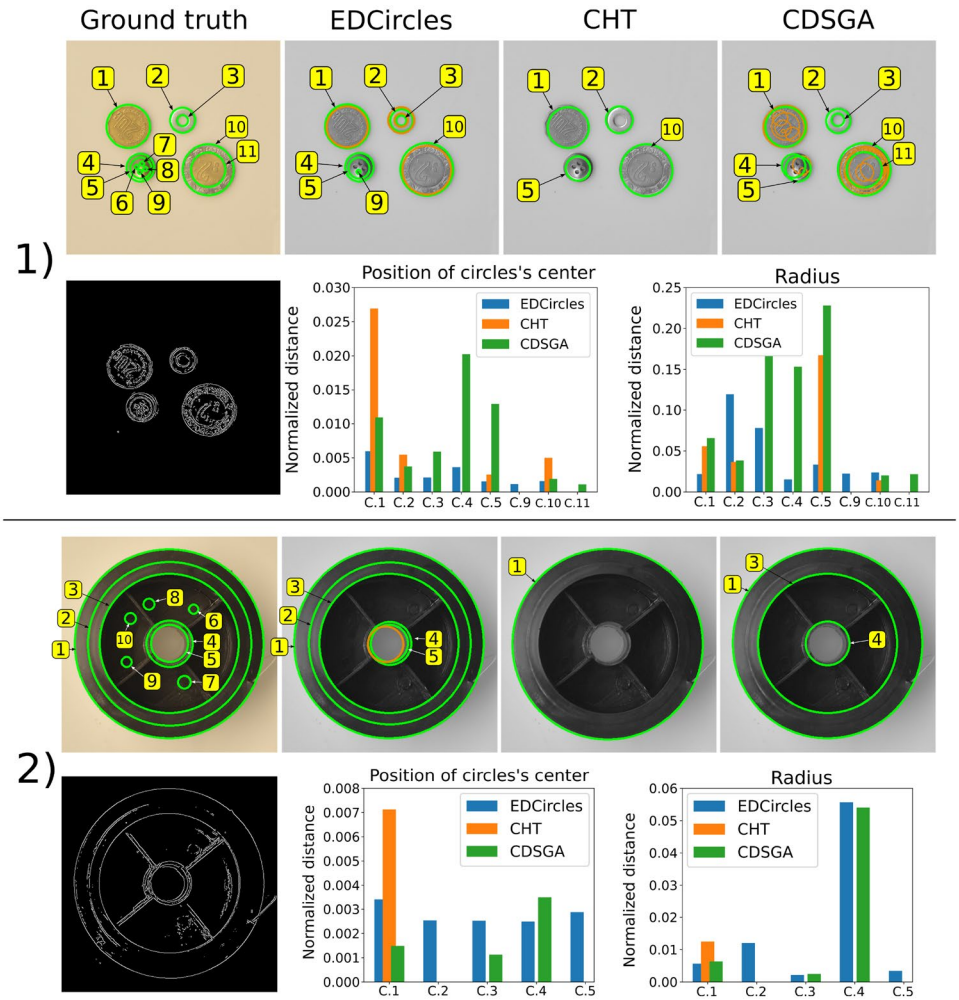



**Fig. 11** Real images with the following features: concentric and multiple sizes. **Ground truth**: input image with manually annotated circles and its edge map. **EDCircles, CHT, CDSGA**: results from each method (for CDSGA $e = 3$). On each row a bar chart from each metric

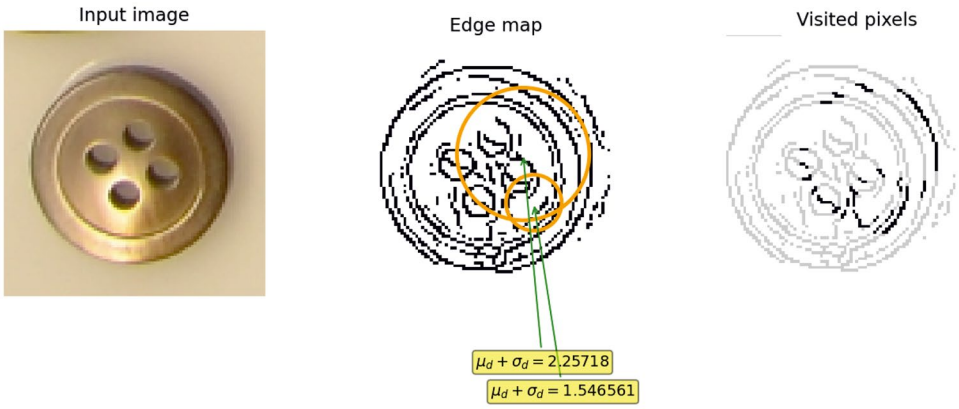



**Fig. 12** Subsample of Fig. 11-1, with its corresponding edge map and two false positives detected by CDSGA, and the visited pixels by each false positive

identified by the algorithm. Here, we can observe that CDSGA detects fewer circles than TLBO and EDCirles in image 4, one more circle than TLBO in image 6, and more circles than CHT in most images. Please notice that none of the methods can recover all circles from image 4 and that when CDSGA outputs more circles than the Ground truth, it is because of false positives. However, most of these false positives are shifted versions of the same original circle.

*Robustness to salt and pepper noise*

A final experiment is intended to assess the robustness to salt and pepper noise of the compared methods. The methods are evaluated in computation time and efficacy. Figure 17 shows the computation time of the three

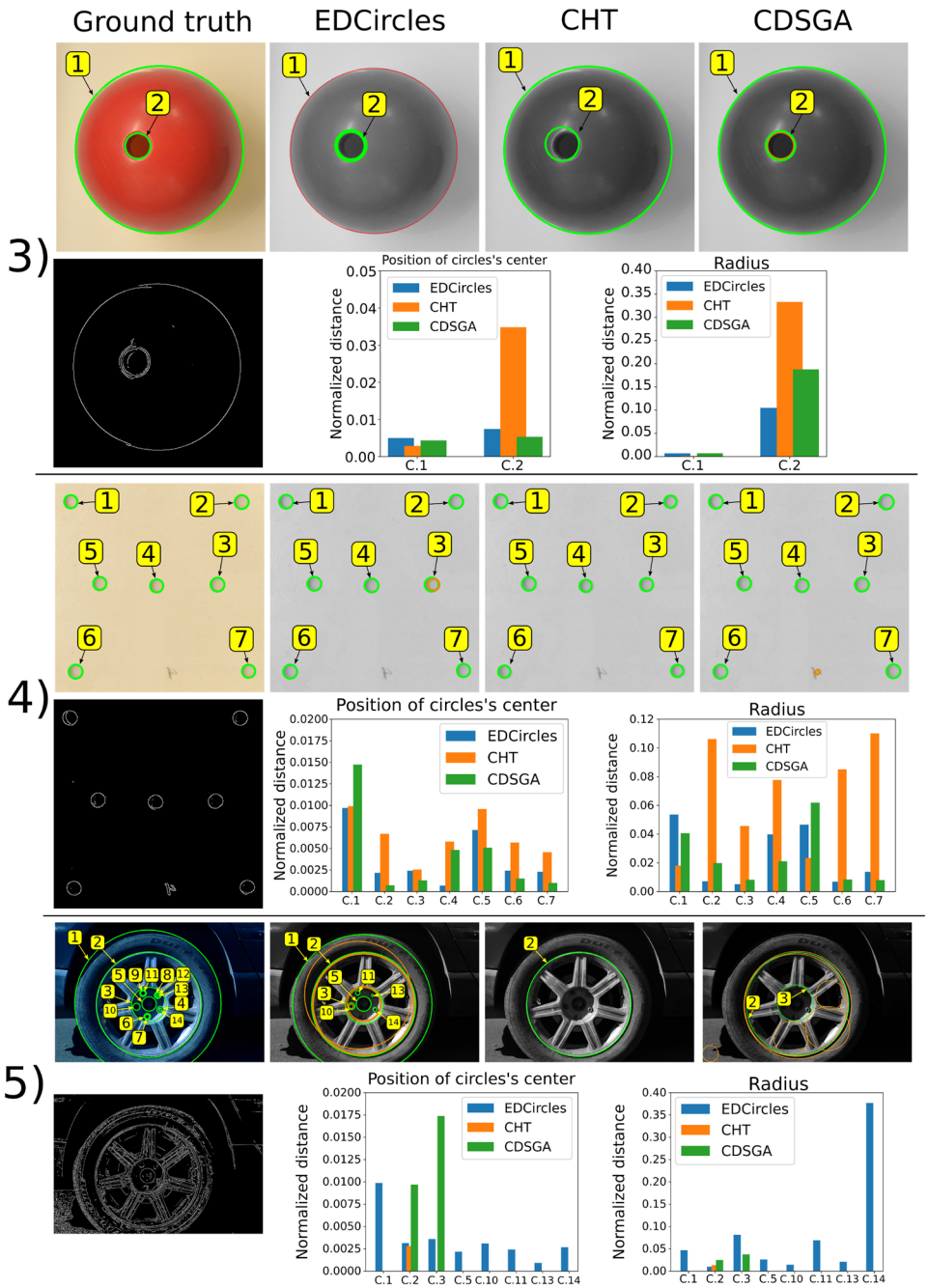



**Fig. 13** Real images with the following features: concentric and multiple sizes. **Ground truth**: input image with manually annotated circle and its edge map. **EDCircles, CHT, CDSGA**: results from each method (for CDSGA $e = 3$). On each row a bar chart from each metric is presented

methods compared in this work. Figure 18 shows the input image used. As depicted, EDCircles method is one order of magnitude faster than CDSGA and two orders faster than the CHT. However, when we compare the output results, as shown in Fig. 18, we can see a decrease in the detection performance concerning the number of detected circles than the CDSGA. Even though the **CHT** approach is a robust circle detection method, in our experiments, the level of noise not only increases its computation time but decreases the efficiency of this approach in circular shapes detection.

A set of synthetic images with gradual Salt and Pepper noise (from 1 to 20%) was used to evaluate every method's relative robustness. We achieved this by counting the number of recovered circles by each method at every noise level. Figure 19 depicts the results. We can observe that CDSGA recovers all circles at every noise level; CHT starts missing some circles at 7% noise, whereas EDCircles misses circles as noise increases. The figure also shows that no circle can be recovered from 16% noise and above.

Notice that images with another type of noise, e.g. Gaussian, will generate a different edge map (see Fig. 20). In this

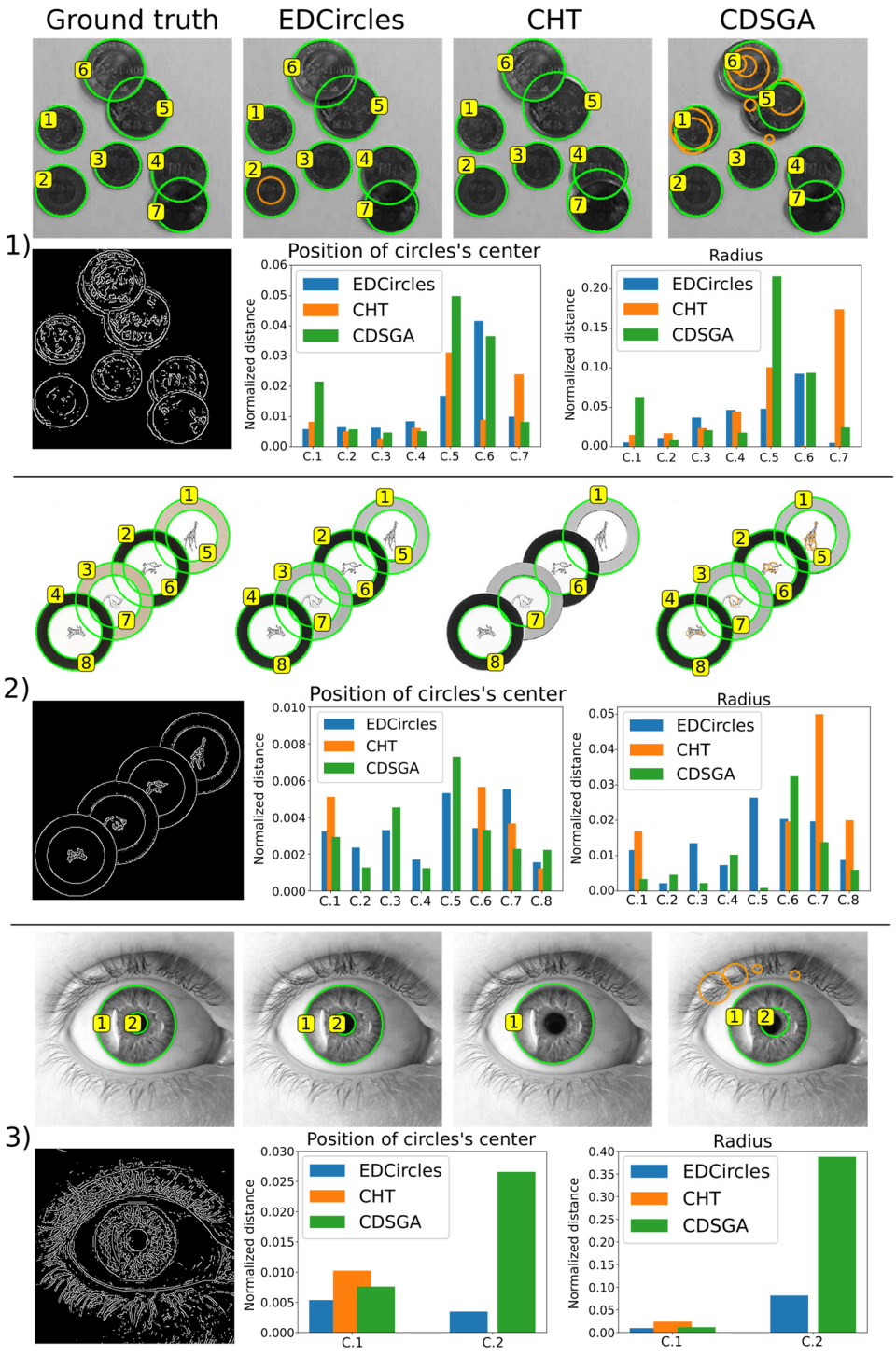



**Fig. 14** Real images from external data set, and multiple shapes; **Ground truth**: input image with overlapped manually annotated circles and its edge map. **EDCircles, CHT, CDSGA**: results from each method (for CDSGA $e = 8$ in all images). On each row a bar chart from each metric is presented

case a different filter method should be used, for instance, for Gaussian noise, Zhang et al. [36] proposed an effective Gaussian blur filter that can be incorporated to our method.

CDSGA presents better results in the following cases: in Fig. 10-2, CDSGA presents the lowest error for position and radius for both circles in the image, whereas EDCircles detects one circle and one ellipse that corresponds to the same hand-drawn circle (circle number 1); in Fig. 14-1, for circles number 1 and 3, CDSGA presents the lowest error for position and radius; and in Fig. 13-5 CDSGA and CHT do not present false positives. In contrast, EDCircles outputs three false positives. Performing an objective assessment of the detection results is problematic since it depends on a manually annotated image (ground truth). To better quantify the obtained results, we analyzed the three methods through metrics that account

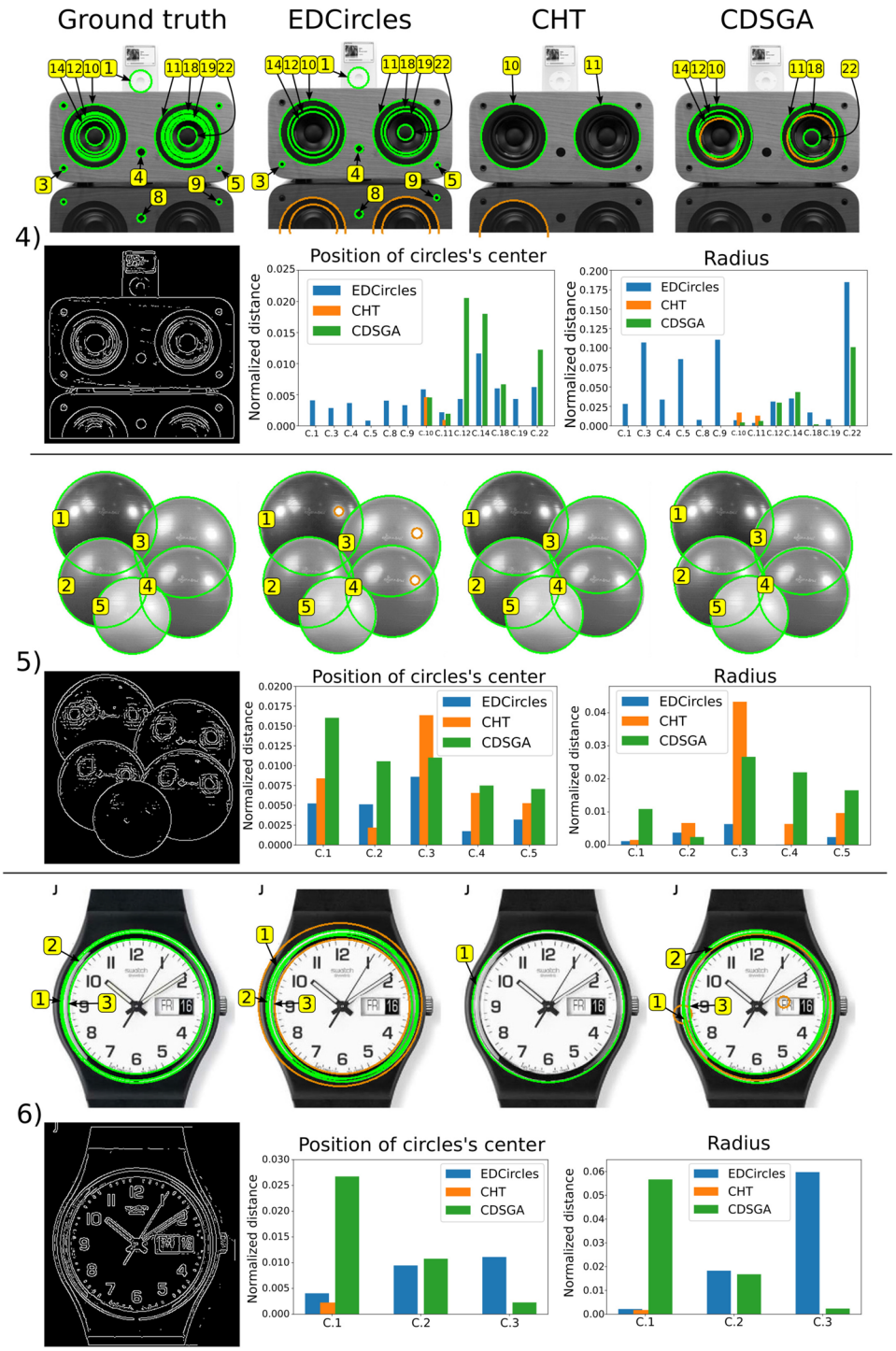



**Fig. 15** Real images from external data set, and multiple shapes; **Ground truth**: input image with overlapped manually annotated circles and their edge maps. **EDCircles, CHT, CDSGA**: results from each method ( $e = 8$ all images). On each row a bar chart for each metric is presented

for false positives. Figure 21 shows a summary multiple box-plot with the results of the 17 images included in this work (from Fig. 9 to 13) and Tables 6 and 7 show the details for each image; we also included the three images presented with noise added in Fig. 18. The metrics used are accuracy, precision, recall, F1 score and false negatives rate (FNR); see supplementary material (appendix E) for each metric's definition. These metrics are often used for binary classification problems, where four quantities are involved: true positive (TP), true negative (TN), false positive (FP) and false negative (FN). In our circle detection problem, TP is the number of correct detected circles, FP is the number of detected circles that are not in the image, and FN is the number of not detected circles. Notice that TN is the number of detected non-circular shapes. However, none of the algorithms is aimed to perform this kind

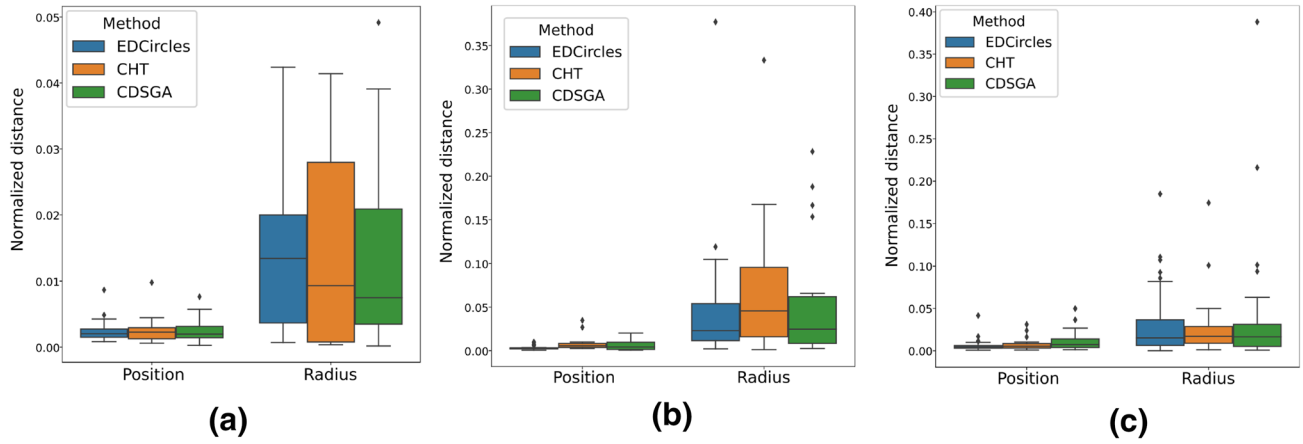



**Fig. 16** Box-plot of the results from the performance metric of EDCircles, CHT, and CDSGA. **a** Synthetic images from Fig. 9; **b** Real images from Figs. 11 and 13; **c** Real images from Figs. 14 and 15 (external dataset)

**Table 3** Shapiro-Wilk normality test, *p*-value results

| Method | Synthetic | | Real | | Real external | |
|---|---|---|---|---|---|---|
| | Position | Radius | Position | Radius | Position | Radius |
| EDCircle | ***0.00** | 0.05 | ***0.00** | ***0.00** | ***0.00** | ***0.00** |
| CHT | ***0.00** | 0.01 | ***0.00** | ***0.00** | ***0.00** | ***0.00** |
| CDSGA | 0.01 | ***0.00** | ***0.00** | ***0.00** | ***0.00** | ***0.00** |

**Table 4** Wilcoxon test, *p*-value results

| CDSGA vs | Synthetic | | Real | | Real external | |
|---|---|---|---|---|---|---|
| | Position | Radius | Position | Radius | Position | Radius |
| EDCircle | 0.3353 | 0.4325 | 0.9385 | 0.8847 | 0.9839 | 0.7224 |
| CHT | 0.51 | 0.5699 | 0.039 | 0.1338 | 0.8529 | 0.4492 |

**Table 5** Number of circles detected on the external dataset by the TLBO [20], EDCircles, CHT method and our proposed CDSGA. The images are from Figs. 14 and 15

| Images in Figs. 14 and 15 | Ground truth | TLBO [20] | EDCircles | CHT | CDSGA |
|---|---|---|---|---|---|
| (1) | 7 | – | 7(8) | 7(7) | 7(15) |
| (2) | 8 | 8 | 8(8) | 4(4) | 8(8) |
| (3) | 2 | – | 2(2) | 1(1) | 2(6) |
| (4) | 22 | 8 | 13(17) | 2(3) | 6(8) |
| (5) | 5 | 5 | 5(8) | 5(5) | 5(5) |
| (6) | 3 | 2 | 3(5) | 1(1) | 3(5) |

of detection. Because of this, specificity and false positive rate (FPR) were not included in the analysis.

Overall, and regarding computation time, EDCircles is the fastest among the three methods. Regarding robustness to salt and pepper noise, CDSGA is the clear winner. When we consider accuracy for the center and radius of the detected circles, Fig. 16 shows that the observed differences are not statistically significant. Table 5 shows that EDCircles can recover more circles from real images than the other methods.

## 6 Conclusions and future work

This work proposes an evolutionary approach to detect multiple fully-contained circles within images automatically. The proposed approach used a Simple Genetic Algorithm to input an edge map and detect circles within this map following an optimization approach. A pre-processing stage, aimed at reducing salt and pepper noise from the image, is included. The approach was tested on synthetic (noise added and clean) and hand-drawn images, with 100% of detection accuracy in both cases. On real images, where circles are affected by shape deformations, the algorithm could not achieve a 100% detection accuracy. These results show comparable or superior performance when

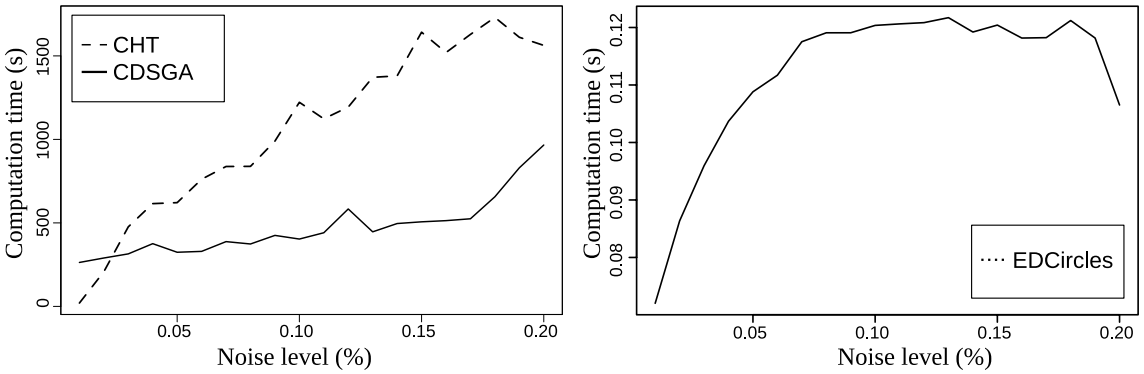
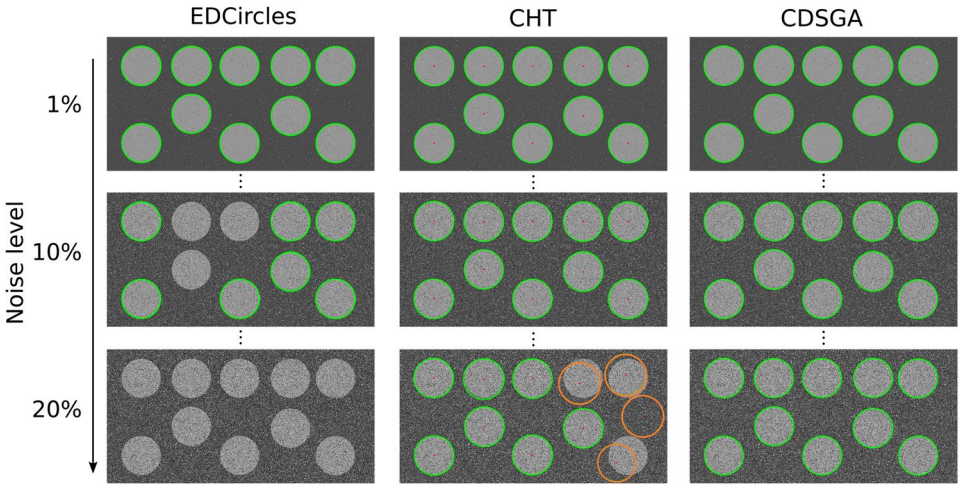



**Fig. 17** Elapsed processing time for circle detection. The dashed line describes **CHT**, the continuous line describes **CDSGA**, the and dotted line belongs to **EDCircles**. Tested on images from Fig. 18

**Fig. 18** Detected circles through EDCircles, CHT and CDSGA methods.



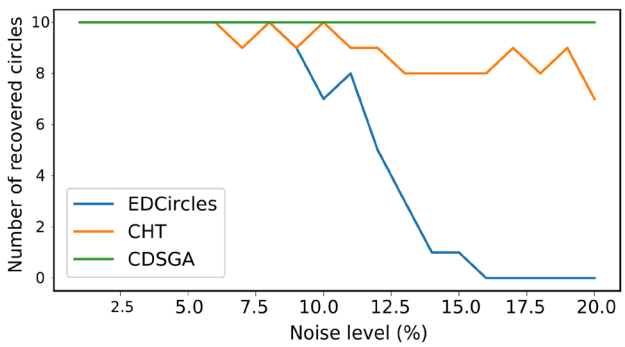
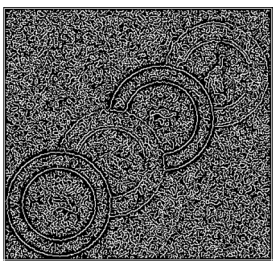




**Fig. 19** Number of recovered circles through EDCircles (blue line), CHT (orange) and CDSGA (green) methods, as a function of the noise level (color figure online)

**Fig. 20** Edge map of Fig. 14-2 with added Gaussian noise, $\sigma = 0.1$



competing with the CHT and the TLBO methods, and improve the EDCircles method regarding noisy images. However, the EDCircles method outperforms all analyzed methods regarding computation time. Furthermore, to compensate for the lack of benchmark images for circle detection, all used images were made publicly available for downloading so that comparison with existing or newly proposed approaches can be performed on the same baseline. Future work aims to: (1) include algorithms that recognize circles with internal geometrical deformations, (2) analyze other filters and edge detectors, (3) include color analysis in order to facilitate the detection of low contrast overlapping circles, (4) a method to define the sub-pixel

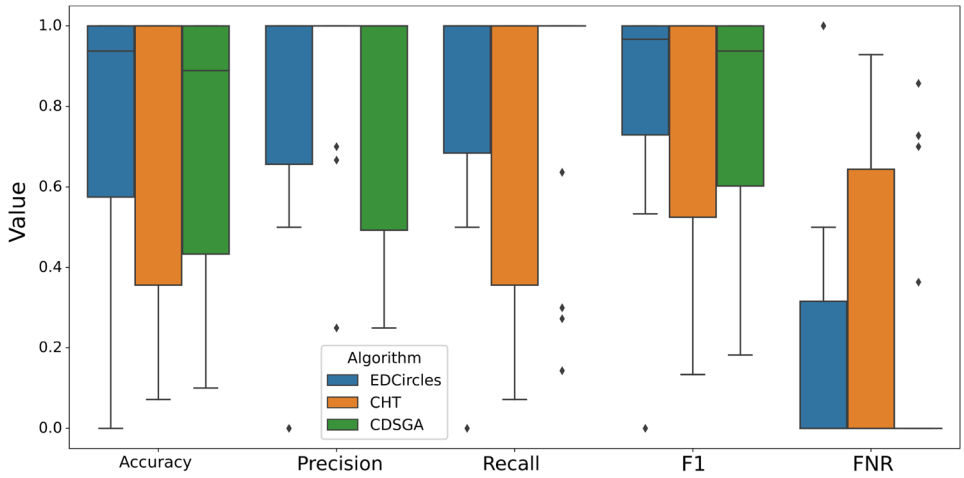



**Fig. 21** Multiple box-plot with the metrics results on each method (EDCircles, CHT and CDSGA)

**Table 6** Metrics results from Figs. 9 to 13 and 18 (accuracy, precision and recall)

| Image | Accuracy | | | Precision | | | Recall | | |
|---|---|---|---|---|---|---|---|---|---|
| | EDcircles | CHT | CDSGA | EDcircles | CHT | CDSGA | EDcircles | CHT | CDSGA |
| Fig. 9-1 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| Fig. 9-2 | 1.000 | 0.167 | 1.000 | 1.000 | 0.250 | 1.000 | 1.000 | 0.333 | 1.000 |
| Fig. 9-3 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| Fig. 9-4 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| Fig. 10-1 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| Fig. 10-2 | 0.667 | 1.000 | 1.000 | 0.667 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| Fig. 11-1 | 0.438 | 0.364 | 0.269 | 0.583 | 1.000 | 0.318 | 0.636 | 0.364 | 0.636 |
| Fig. 11-2 | 0.455 | 0.100 | 0.300 | 0.833 | 1.000 | 1.000 | 0.500 | 0.100 | 0.300 |
| Fig. 13-1 | 1.000 | 1.000 | 0.667 | 1.000 | 1.000 | 0.667 | 1.000 | 1.000 | 1.000 |
| Fig. 13-2 | 1.000 | 1.000 | 0.778 | 1.000 | 1.000 | 0.778 | 1.000 | 1.000 | 1.000 |
| Fig. 13-3 | 0.364 | 0.071 | 0.100 | 0.500 | 1.000 | 0.250 | 0.571 | 0.071 | 0.143 |
| Fig. 14-1 | 0.875 | 1.000 | 0.467 | 0.875 | 1.000 | 0.467 | 1.000 | 1.000 | 1.000 |
| Fig. 14-2 | 1.000 | 0.500 | 0.471 | 1.000 | 1.000 | 0.471 | 1.000 | 0.500 | 1.000 |
| Fig. 14-3 | 1.000 | 0.500 | 0.333 | 1.000 | 1.000 | 0.333 | 1.000 | 0.500 | 1.000 |
| Fig. 15-1 | 0.500 | 0.087 | 0.250 | 0.765 | 0.667 | 0.750 | 0.591 | 0.091 | 0.273 |
| Fig. 15-2 | 0.625 | 1.000 | 1.000 | 0.625 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| Fig. 15-3 | 0.600 | 0.333 | 0.500 | 0.600 | 1.000 | 0.500 | 1.000 | 0.333 | 1.000 |
| Fig. 18-1% | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| Fig. 18-10% | 0.700 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 0.700 | 1.000 | 1.000 |
| Fig. 18-20% | 0.000 | 0.538 | 1.000 | 0.000 | 0.700 | 1.000 | 0.000 | 0.700 | 1.000 |

**Table 7** Metrics results from Figs. 9 to 13 and 18 (F1 score and FNR)

| Image | F1 score | | | FNR | | |
|---|---|---|---|---|---|---|
| | EDcircles | CHT | CDSGA | EDcircles | CHT | CDSGA |
| Fig. 9-1 | 1.000 | 1.000 | 1.000 | 0.000 | 0.000 | 0.000 |
| Fig. 9-2 | 1.000 | 0.286 | 1.000 | 0.000 | 0.667 | 0.000 |
| Fig. 9-3 | 1.000 | 1.000 | 1.000 | 0.000 | 0.000 | 0.000 |
| Fig. 9-4 | 1.000 | 1.000 | 1.000 | 0.000 | 0.000 | 0.000 |
| Fig. 10-1 | 1.000 | 1.000 | 1.000 | 0.000 | 0.000 | 0.000 |
| Fig. 10-2 | 0.800 | 1.000 | 1.000 | 0.000 | 0.000 | 0.000 |
| Fig. 11-1 | 0.609 | 0.533 | 0.424 | 0.364 | 0.636 | 0.364 |
| Fig. 11-2 | 0.625 | 0.182 | 0.462 | 0.500 | 0.900 | 0.700 |
| Fig. 13-1 | 1.000 | 1.000 | 0.800 | 0.000 | 0.000 | 0.000 |
| Fig. 13-2 | 1.000 | 1.000 | 0.875 | 0.000 | 0.000 | 0.000 |
| Fig. 13-3 | 0.533 | 0.133 | 0.182 | 0.429 | 0.929 | 0.857 |
| Fig. 14-1 | 0.933 | 1.000 | 0.636 | 0.000 | 0.000 | 0.000 |
| Fig. 14-2 | 1.000 | 0.667 | 0.640 | 0.000 | 0.500 | 0.000 |
| Fig. 14-3 | 1.000 | 0.667 | 0.500 | 0.000 | 0.500 | 0.000 |
| Fig. 15-1 | 0.667 | 0.160 | 0.400 | 0.409 | 0.909 | 0.727 |
| Fig. 15-2 | 0.769 | 1.000 | 1.000 | 0.000 | 0.000 | 0.000 |
| Fig. 15-3 | 0.750 | 0.500 | 0.667 | 0.000 | 0.667 | 0.000 |
| Fig. 18-1% | 1.000 | 1.000 | 1.000 | 0.000 | 0.000 | 0.000 |
| Fig. 18-10% | 0.823 | 1.000 | 1.000 | 0.300 | 0.000 | 0.000 |
| Fig. 18-20% | 0.000 | 0.700 | 1.000 | 1.000 | 0.300 | 0.000 |

validation threshold as a function of the input image and (5) an exhaustive false positive detection stage.

**Supplementary Information** The online version contains supplementary material available at https://doi.org/10.1007/s10044-021-01007-6.

**Acknowledgements** We thank the anonymous reviewers for their suggestions that helped improve the paper a great deal. MRGM acknowledges the National Council for Science and Technology (CONACyT) support through scholarship No. 762659. MRGM and MEMR thank the Universidad Autónoma de Baja California (UABC), Programa para el Fortalecimiento y Calidad Educativa (PFCE) and to the Programa para el Desarrollo Profesional Docente (PRODEP) for providing the conditions to develop this research.

## References

1. Abdel-Khalek S, Ishak AB, Omer OA, Obada AS (2017) A two-dimensional image segmentation method based on genetic algorithm and entropy. Optik-Int J Light Electron Opt 131:414–422
2. Akinlar C, Topal C (2012) Edpf: a real-time parameter-free edge segment detector with a false detection control. Int J Pattern Recognit Artif Intell 26(01):1255002
3. Akinlar C, Topal C (2013) Edcircles: a real-time circle detector with a false detection control. Pattern Recognit 46(3):725–740. https://doi.org/10.1016/j.patcog.2012.09.020
4. Canny J (1986) A computational approach to edge detection. IEEE Trans Pattern Anal Mach Intell 8(6):679–698
5. Chauris H, Karoui I, Garreau P, Wackernagel H, Craneguy P, Bertino L (2011) The circlet transform: a robust tool for detecting features with circular shapes. Comput Geosci 37(3):331–342
6. Chen TC, Chung KL (2001) An efficient randomized algorithm for detecting circles. Comput Vis Image Underst 83(2):172–191
7. Coley DA (1997) An introduction to genetic algorithms for scientists and engineers, har/dskt edn. World Scientific, Singapore
8. Cuevas E, Sención-Echauri F, Zaldivar D, Pérez-Cisneros M (2012) Multi-circle detection on images using artificial bee colony (abc) optimization. Soft Comput 16(2):281–296
9. Cuevas E, Wario F, Zaldivar D, Pérez-Cisneros M (2013) Circle detection on images using learning automata. Artificial intelligence, evolutionary computing and metaheuristics. Springer, Berlin, pp 545–570
10. Dawson-Howe K (2014) A practical introduction to computer vision with opencv. John Wiley & Sons, Hoboken
11. Dong N, Wu CH, Ip WH, Chen ZQ, Chan CY, Yung KL (2012) An opposition-based chaotic ga/pso hybrid algorithm and its application in circle detection. Comput Math Appl 64(6):1886–1902
12. García S, Molina D, Lozano M, Herrera F (2009) A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the cec'2005 special session on real parameter optimization. J Heuristics 15:617–644
13. Gudmundsson M, El-Kwae EA, Kabuka MR (1998) Edge detection in medical images using a genetic algorithm. IEEE Trans Med Imaging 17(3):469–474
14. Holland JH (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press, Cambridge
15. Hollitt C, Johnston-Hollitt M (2012) Feature detection in radio astronomy using the circle hough transform. Publ Astron Soc Aust 29(3):309–317
16. Illingworth J, Kittler J (1988) A survey of the hough transform. Comput Vis Graph Image Process 44(1):87–116
17. Inkscape Project: Inkscape (2011). https://inkscape.org
18. Jia LQ, Peng CZ, Liu HM, Wang ZH (2011) A fast randomized circle detection algorithm. In: Image and Signal Processing

(CISP), 2011 4th International Congress on, vol. 2, pp. 820–823. IEEE
19. Liu D, Wang Y, Tang Z, Lu X (2014) A robust circle detection algorithm based on top-down least-square fitting analysis. Comput Elect Eng 40(4):1415–1428
20. López A, Cuevas FJ (2018) Automatic multi-circle detection on images using the teaching learning based optimisation algorithm. IET Comput Vis 12(8):1188–1199
21. Mairesse F, Sliwa T, Binczak S, Voisin Y (2008) Subpixel determination of imperfect circles characteristics. Pattern Recognit 41(1):250–271
22. Murillo-Bracamontes EA, Martinez-Rosas ME, Miranda-Velasco MM, Martinez-Reyes HL, Martinez-Sandoval JR, Cervantes-de Avila H (2012) Implementation of hough transform for fruit image segmentation. Procedia Eng 35:230–239
23. OpenCV: opencv/houghcircle_demo.cpp at master . opencv/opencv . github. https://github.com/opencv/opencv/blob/master/samples/cpp/tutorial_code/ImgTrans/HoughCircle_Demo.cpp (2011)
24. Peckinpaugh SH, Holyer RJ (1994) Circle detection for extracting eddy size and position from satellite imagery of the ocean. IEEE Trans Geosci Remote Sens 32(2):267–273
25. Pérez-Zavala R, Torres-Torriti M, Cheein FA, Troni G (2018) A pattern recognition strategy for visual grape bunch detection in vineyards. Comput Electron Agric 151:136–149
26. Rényi A (1961) On measures of entropy and information. Tech. rep, HUNGARIAN ACADEMY OF SCIENCES, Budapest, Hungary
27. Roth G, Levine MD (1994) Geometric primitive extraction using a genetic algorithm. IEEE Trans Pattern Anal Mach Intell 16(9):901–905. https://doi.org/10.1109/34.310686
28. Schwarz L, Gamba HR, Pacheco FC, Ramos RB, Sovierzoski MA (2012) Pupil and iris detection in dynamic pupillometry using the opencv library. In: 2012 5th International Congress on Image and Signal Processing, pp. 211–215. IEEE
29. Tan L, Jiang J (2007) Fundamentals of analog and digital signal processing. AuthorHouse
30. Tsallis C (1993) Generalized entropy-based criterion for consistent nonparametric testing. preprint
31. Van Aken JR (1984) An efficient ellipse-drawing algorithm. IEEE Comput Graph Appl 4(9):24–35
32. Victor AR, Carlos H, Arturo PG, Raul E (2006) Circle detection on images using genetic algorithms. Pattern Recognit Lett 27(6):652–657
33. Wu J, Chen K, Gao X (2013) Fast and accurate circle detection using gradient-direction-based segmentation. JOSA A 30(6):1184–1192
34. Wu WY et al (2017) A hybrid method for circle detection. Sour J Multidiscip Eng Sci Stud (JMESS) 3:1975–1981
35. Yuan B, Liu M (2015) Power histogram for circle detection on images. Pattern Recognit 48(10):3268–3280
36. Zhang H, Wiklund K, Andersson M (2016) A fast and robust circle detection method using isosceles triangles sampling. Pattern Recognit 54:218–228
37. Zhou B, He Y (2017) Fast circle detection using spatial decomposition of hough transform. Int J Pattern Recognit Artif Intell 31(03):1755006