Hindawi
Journal of Optimization
Volume 2017, Article ID 9710719, 11 pages
https://doi.org/10.1155/2017/9710719

Research Article

Robust Circle Detection Using Harmony Search

Jaco Fourie

Lincoln Agritech Ltd., Lincoln, New Zealand

Correspondence should be addressed to Jaco Fourie; jacofourie@gmail.com

Received 10 January 2017; Accepted 24 May 2017; Published 2 August 2017

Academic Editor: Ferrante Neri

Automatic circle detection is an important element of many image processing algorithms. Traditionally the Hough transform has been used to find circular objects in images but more modern approaches that make use of heuristic optimisation techniques have been developed. These are often used in large complex images where the presence of noise or limited computational resources make the Hough transform impractical. Previous research on the use of the Harmony Search (HS) in circle detection showed that HS is an attractive alternative to many of the modern circle detectors based on heuristic optimisers like genetic algorithms and simulated annealing. We propose improvements to this work that enables our algorithm to robustly find multiple circles in larger data sets and still work on realistic images that are heavily corrupted by noisy edges.

1. Introduction

Circle detection is a key element in the larger field of automatic extraction and identification of geometric shapes from digital images. The ability to identify and extract shapes from images has many applications in industry and agriculture. This is because simple geometric shapes are common in man-made environments and are often used in symbols that only have meaning when properly identified.

Circle and ellipse are particularly common shapes to identify due to their application in biological cell tracking, automated mechanical parts inspection, and biometrics (iris detection). Circle detection is traditionally done using the circle Hough transform (CHT) [1, 2]. The CHT algorithm has been used for circle detection for over 30 years and much research has been done to improve the original algorithm. Most of the research focused on the main limitations of CHT, namely, high computational and storage requirements. Proposed improvements include probabilistic, randomised, and fuzzy adaptations of the original algorithm [3–5]. These improvements all aim to decrease the computational complexity of CHT and some also address the accuracy of the algorithm in the presence of noise [6].

With the dramatic increase in computational power, more researchers have begun investigating metaheuristic optimisation algorithms as possible approaches to more robust and accurate circle detection. These algorithms are often biologically inspired like the genetic algorithm and the particle swarm optimisation algorithm. They are usually computationally expensive but have the advantage of not making any assumptions about the objective function or the amount and type of noise that may be present. This usually implies more robust and accurate performance on noisy or ambiguous data.

Both genetic algorithms and particle swarm optimisation algorithms have been successfully used in circle detection along with other metaheuristics like simulated annealing and bacterial foraging optimisation algorithm [7–10]. We propose the use of another biologically inspired metaheuristic optimiser, namely, the Harmony Search (HS) algorithm [11].

Harmony Search is a musical improvisation-inspired metaheuristic optimisation method originally developed for the design of pipeline network systems and published in 2001 [12]. Since its publication, it has been used in many engineering and scientific fields, including computer science, as a robust optimisation technique [13].

2. Theory

In this section, we introduce circle detection in digital images and give an overview of the circle Hough transform (CHT).

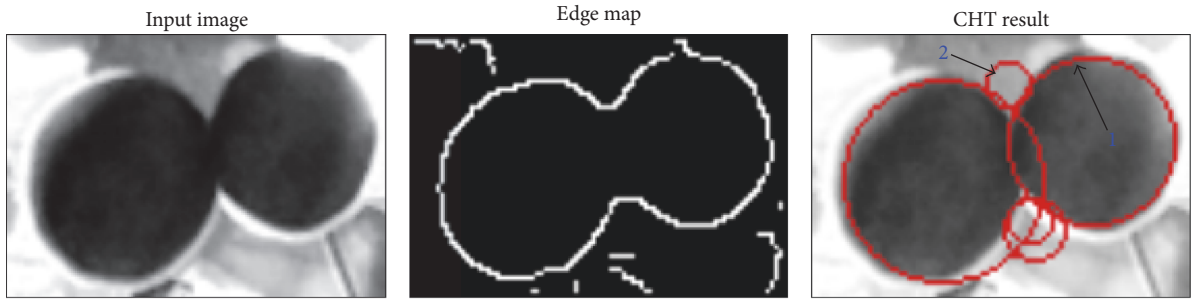


Figure 1: An example of the CHT detecting multiple circles in an image. The largest and second-largest maxima in the accumulator matrix is labelled as 1 and 2, respectively.

Being the most popular method of circle detection, the CHT gives us a starting point to investigate circle detection methods with specific focus on their limitations and accuracy under challenging conditions.

We pay specific attention to circle detection methods that interpret the problem as a fitness function that has to be optimised and that make use of metaheuristics to solve the optimisation problem. We then introduce Harmony Search (HS) as a heuristic optimiser and mention some of the variants of the original algorithm that attempt to address the limitations of traditional HS.

2.1. Classical Hough Transform Based Circle Detection. As previously mentioned, the classical way to do circle detection is using the circular Hough transform. This method assumes that the edge pixels of the image have already been identified using one of the many edge detection methods, for example, the Canny edge detector [14]. The collection of edge pixels, called the edge map, is then processed to identify which of the pixels is part of a circle by transforming the edge pixels into a 3D parameter matrix called the Hough parameter space. The maximum point in the parameter space corresponds to the parameters of a circle that has the most edge pixels on its perimeter.

Consider a circle in 2D space represented by the equation

$$(x-a)^2 + (y-b)^2 = r^2, \tag{1}$$

where (a, b) represents the centre of the circle and r is the radius. From this equation, we can map each pixel (x, y) to a conic surface in the 3D parameters space (a, b, r) that represents all possible circles that the pixel can form a part of. By discretising and limiting the parameter space based on the original image dimensions, a 3D accumulator matrix is formed. The location of the maxima in this matrix represents the circles detected by the CHT.

Though this simple approach is somewhat robust to noise and occlusion, the main limitation is in the computational and space requirements. As the 3D accumulator matrix grows cubically with the size of the image, the memory required to store this matrix and the time spent creating it quickly become prohibitive. This is especially true when circle location accuracy demands that the parameter space be quantised into accumulator cells that are as small as possible or when the space needs to be enlarged to beyond the size of the image to allow for partial circle detection that may have centres that fall outside the image bounds. In Figure 1, we show a common result of the CHT applied to a noisy image. In this example, we did not apply a set threshold to determine which maxima in the accumulator should be selected as valid circles but instead picked the top 5 largest values. In this way, we highlight the false circles that are often detected in real image examples. We label the strongest circle in the accumulator as 1 and the second strongest as 2 to show that simply picking the strongest circles in the accumulator will not solve the false circle problem even when the exact number of circles to be detected is previously known. If we only picked the strongest two circles in this example, the large circle on the left would not be detected while the small circle in the middle is chosen instead.

2.2. Circle Detection Using Metaheuristics. In order to address some of the limitations of the CHT, researchers have extended and adapted the CHT in various ways to improve the accuracy but mainly to improve the performance and improve the large memory requirements that classical CHT has. Some examples of CHT extensions include the randomised Hough transform [5] and the fuzzy Hough transform [16]. Both these approaches attempt to improve performance by either decreasing the memory and computational load, as with the randomised Hough transform, or by generalising the algorithm to improve robustness as with the fuzzy Hough transform.

More recently researchers have also begun to apply evolutionary algorithms to circle detection. Examples include genetic algorithm based approaches [8], differential evolution based approaches [17], and bacterial foraging algorithm based approaches [10]. These methods all have the common feature that they turn the circle detection problem into a mathematical optimisation problem. It is important to note however that no assumptions are being made regarding the amount or type of noise in the search space of solutions for this optimisation problem. The shape of the search space is also not assumed and can be discrete and highly multimodal and can contain discontinuities that would make it difficult for traditional gradient-based optimisation techniques to

input: A fitness function f for scoring the solutions along with an upper bound (UB) and lower bound (LB) for each dimension of the M-dimensional solution. A function, $U(l, u)$, for sampling random numbers between l and u from a uniform distribution is also required.
output: The best solution $x_{\text{best}} = [x_0, \ldots, x_M]$

(1) Initialise the HM with random entries
(2) **foreach** $n \in [1, \text{NI}]$ **do**
(3) **foreach** $i \in [1, M]$ **do**
(4) **if** $U(0, 1) > \text{HMCR}$ **then**
(5) $x_i' \longleftarrow \text{LB}_i + r \times (\text{UB}_i - \text{LB}_i)$
(6) **else**
(7) $x_i' \longleftarrow x_i^j$
(8) **if** $U(0, 1) \leq \text{PAR}$ **then**
(9) $x_i' \longleftarrow x_i' + r \times \text{FW}$
(10) **end**
(11) **end**
(12) $x_{\text{new}}[i] \longleftarrow x_i'$
(13) **end**
(14) select the lowest scoring solution x_{worst} from the HM
(15) **if** $f(x_{\text{new}}) > f(x_{\text{worst}})$ **then**
(16) remove x_{worst} from HM and replace with x_{new}
(17) **end**
(18) select the highest scoring solution x_{best} from the HM and return it

Algorithm 1: The Harmony Search algorithm.

successfully traverse and find a solution. This is why heuristic optimisers, like the family of evolutionary algorithms, are used since they typically make no assumptions about the fitness function that describe the search space.

2.3. Harmony Search Optimisation. The (HS) algorithm was inspired by the way musicians collectively and cooperatively improve harmonies in certain genres of music, especially Jazz [11, 12]. Like evolutionary algorithms, it generates a population of candidate solutions and then iteratively improves on the candidate population by adding and removing individual candidates. Unlike most evolutionary algorithms, it does not update the entire solution population at every iteration but only changes one individual at a time. This maintains critical diversity in the population for more iterations making the early convergence to local optima much less likely.

Consider the analogy of a Jazz trio during an improvisation session. Each member of the trio plays a note on their instrument selected from a limited range of notes which may be different for each member. Together the three notes form a harmony that may or may not be pleasing to the audience (or the trio itself). For the sake of this analogy, the aim of the trio is to find the 3-note harmony that is most pleasing to the audience.

In our analogy, the 3-note harmony is the solution to an optimisation problem of finding the most pleasing harmony as judged by the audience. The search space of possible solutions is 3-dimensional in this case and is bounded by the notes playable by each musician. For example, if each musician could only choose from 5 different notes, the search of all possible solutions would only contain $5 \times 5 \times 5 = 125$ candidates. The fitness function that evaluates each candidate harmony is represented by the audience.

Improvisation of new harmonies is an iterative process involving each member of the trio choosing a note to play based on what resulted in pleasing harmonies during previous iterations. In this way, the improvisation process uses a memory of good solutions that is constantly updated and improved with new entries as better harmonies are discovered. The improvisation process continues until either a harmony of sufficiently high quality is produced or the maximum number of iterations has been reached.

This HS analogy is summarised as follows. The performance of HS is controlled by four parameters: the harmony memory size (HMS), the harmony memory consideration rate (HMCR), the fret width (FW), and the maximum iterations (NI). The HMS is the number of *good solutions* that are stored at any one time in the harmony memory (HM) and used for future iterations. The HMCR determines how likely the memory is considered as opposed to randomly selecting a value during improvisation and the FW determines the amount a value from the memory may be adjusted during improvisation. The NI puts a limit on how long the process may continue by limiting the number of improvisation attempts. HS can be described in pseudo code as shown in Algorithm 1.

Algorithm 1 describes the simplest form of HS and while this form has been used in many practical applications, much research has gone into improving HS to be more robust, accurate, and computationally efficient. Over the last

Figure 2: The Canny edge detector is used to extract the edge map.

decade, many new variations of HS have been developed. The majority aim to address a limitation of HS arising from the choice of the HMS, HMCR, PAR, and FW parameters. The optimal choice of these parameters depends on the shape of the search space which in most optimisation problems is unknown. Sensible default values for each of these parameters can be found in the first published results [12] and will result in good performance in most cases but can usually be improved considerably. Most of the modern HS variants remove one or more of these parameters from the algorithm completely or replace them with other parameters that are less sensitive to the shape of the search space or are easier to choose. These improved HS variants have been used in many diverse applications and the interested reader is referred to [18–23], for examples, many HS variants that are used in computer vision, vehicle routing, music composition, solving Sudoku, and various engineering disciplines.

3. Circle Detection as an Optimisation Problem

The main contribution of this work is based on applying a variation of HS to the circle detection problem. This approach was pioneered by Cuevas et al. and their work forms the basis of our proposed algorithm [24]. This section starts with a review of their work followed by a description of our contribution and how it relates to the original pioneering work.

In their work, Cuevas et al. describe circles in an image by the well-known second-degree equation introduced in Section 2.1 as (1). Images are preprocessed to extract object outlines or edges in the image by using the Canny edge detector [14]. This identifies the pixels in the image that should be tested as potentially being part of circles in the image. In Figure 2, an image pair representing the original image and its edges is shown to illustrate the way the Canny edge detector is used to identify circular edge candidates.

Detection of circles in the image is interpreted as an optimisation problem with each edge pixel in the edge map considered to be part of the search space of possible solutions. A candidate solution, that is, a circle in the image, is represented as three distinct edge points since 3 points on a plane are required to define a unique circle. Candidate solutions are scored using a fitness function defined as the number of pixels in the edge map that support the circle described by the 3 points in the candidate solution. The more edge pixels that lie on the circle described by the candidate solution, the higher its fitness.

3.1. Circle Detection Using Harmony Search. Once the circle detection problem is defined as optimisation of a finite search space and an appropriate fitness function is provided to compare candidate solutions, HS can be used to find the best solution in the search space. Since the search space is simply a vector of all the edge pixels, candidate solutions can be coded in the HM as three indices into the vector of edge points. Let $P = \{p_1, p_2, p_3, \ldots, p_E\}$ be the set of E edge points, that is, the search space. Then a candidate solution in the HM is a 3-dimensional vector $x_c = [p_i, p_j, p_k]$, where $0 \leq i, j, k < E$. This means that we now have a standard 3-dimensional optimisation problem with a search space limited in all three dimensions by the number of edge pixels.

All that remains to be defined is the fitness function. Let $p_i = [x_i, y_i]$, $p_j = [x_j, y_j]$, and $p_k = [x_k, y_k]$ be 3 edge pixels that together define a circle and a solution vector. The radius, r, and the centre, x_0, y_0 of this circle can be calculated as follows. Let

$$X_{ji} = x_j - x_i,$$

$$X_{ki} = x_k - x_i,$$

$$Y_{ji} = y_j - y_i,$$

$$Y_{ki} = y_k - y_i,$$

$$A_0 = x_j^2 + y_j^2 - x_i^2 - y_i^2,$$

$$A_1 = x_k^2 + y_k^2 - x_i^2 - y_i^2,$$

$$A = \begin{bmatrix} A_0 & 2Y_{ji} \\ A_1 & 2Y_{ki} \end{bmatrix},$$

$$B = \begin{bmatrix} 2X_{ji} & A_0 \\ 2X_{ki} & A_1 \end{bmatrix},$$
$$D = 4\left(X_{ji}Y_{ki} - X_{ki}Y_{ji}\right); \tag{2}$$

then,

$$x_0 = \frac{\det(A)}{D},$$
$$y_0 = \frac{\det(B)}{D}, \tag{3}$$
$$r = \sqrt{(x_0 - x_i)^2 + (y_0 - y_i)^2},$$

where det denotes the matrix determinant.

With the circle centre and radius calculated, we can generate a list of pixels that lie on the perimeter of that circle $S = \{s_0, s_1, \ldots, s_C\}$ by using the midpoint circle algorithm (MCA) [25]. The fitness function f is then defined as follows. Let

$$E(P, s) = \begin{cases} 1 & \text{if } s \text{ is one of the edge points in } P \\ 0 & \text{if } s \text{ is not an edge point in } P \end{cases} \tag{4}$$

$$f(S) = \left[\frac{\sum_{i=0}^{C} E(P, s_i)}{C}\right]. \tag{5}$$

In other words, $f(S)$ simply counts the number of points on the perimeter of S that are also edge pixels from the set of all edge pixels P.

In this implementation, the detector will only find one circle but Cuevas et al. expand this idea to allow for multiple circle detection by removing any circles found from the edge map and then iteratively rerunning the HS optimiser until no more circles can be found of sufficient quality or until some threshold has been reached. For further details as well as experimental results from this circle detector, the reader is referred to the original publication by Cuevas et al. [24].

Before describing the proposed algorithm that forms the main contribution of this work, it is valuable to investigate the size and shape of the search space that HS optimises over. HS was designed to be robust enough to optimise over almost any kind of search space even those that are highly discontinuous, multimodal, and not differentiable. However, like most heuristic optimisers, HS relies to some extent on the existence of a *basin of attraction* around local optima in the search space, even if it is small compared to the size of the search space [26]. This means that solutions close to each other (based on some distance measure) in the search space are expected to have similar fitness values. In situations where this is not the case, HS may still be able to find a good solution but its performance will suffer greatly and could even degrade into a randomised search. As an example, refer to Figure 3 that shows a 3-dimensional representation of the Ackley function [27] which is often used to benchmark optimisation algorithms. Notice that it is highly multimodal

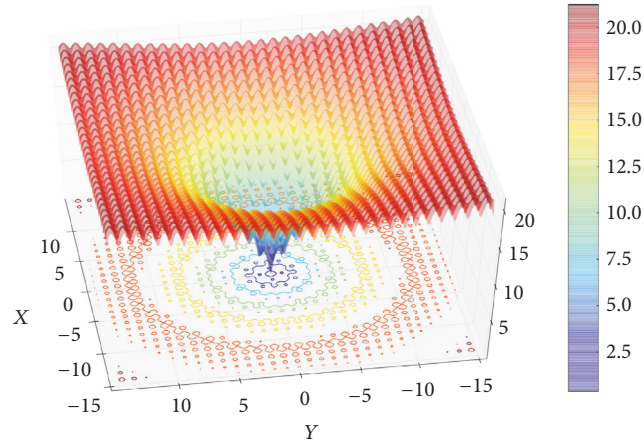


Figure 3: Illustrating local minima and basins of attraction in the Ackley function.

and has many local minima but only one global minimum. Also notice that even though they may be small, each local minimum is surrounded by a small neighbourhood of points that have similar function values and represents the basin of attraction around that local minimum.

Even on a small 1-megapixel image, the number of edge points can easily be in in the order of 10,000. Since any three of these points can potentially form a circle, the size of the search space that HS optimises over is $10{,}000^3 = 10^{12}$ potential solutions. This large search space is typically highly multimodal since a dense edge map will have many ways in which pixels from different areas can be combined into circular or approximately circular shapes. These characteristics on their own already make for a challenging optimisation problem but a greater problem lies in the shape of the search space.

Consider the way that a solution vector is defined from the list of edge pixels. Instead of storing the pixel location in, for example, Cartesian coordinates, a solution vector is three indices into the list of edge pixels which inherently discards the relative distance information between pixels. This decision was made for practical reasons since a more complicated encoding of the solution would mean that the HS improvisation step would need to be adapted so that improvisation based on previous good solutions would still likely result in better ones. However, one result of this encoding is that edge pixels that are far from each other (large difference between index values) in the list of edge pixels may be close together on the image plane in a Euclidean sense. This means that edge pixels that are neighbours and therefore likely belong to the same circle are not neighbours in the search space. Consider the example of Figure 4.

Part of the edge list from an example image is displayed as an edge map in Figure 4. One can easily identify two separate circular objects where each contains multiple edge pixels which could be considered as belonging to a circle. However, if one encodes this edge map by starting from the top row of the image and recording the edge pixel locations row-by-row, then edge pixels that are in the same row will be neighbours in the edge list as well as those which are

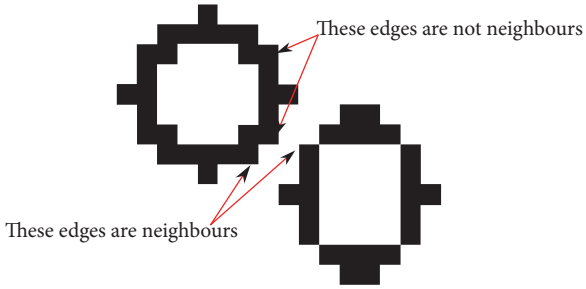


Figure 4: Edge pixels can be close together on the image plane and belong to the same circle but may be far apart in the list of edge pixels that define the search space.

separated by rows containing only nonedge pixels. However, pixels on different rows are not neighbours in the search space even when they are close together on the image plane. Figure 4 highlights edge pixels that are close together on the image plane and belong to the same circle but will not be in the same neighbourhood in the search space. Similarly, it also shows two pixels that would be right next to each other in the search space but belong to completely different objects. Note that as long as there are no nonedge pixels between those two pixels, they will always be next to each other in the search space no matter how large the distance is between them on the image plane.

This is a serious problem for the HS algorithm as it means that a basin of attraction in the search space will not necessarily contain the edge pixels required to find the circle that it may represent. Also, small changes to a solution in the HM (pitch adjustment in HS) may cause edge pixels from a completely different part of the image to be included in a new improvisation that should be based on an already discovered circle.

In small images (few edge pixels), this limitation in the encoding of the HM may simply cause slow convergence while still resulting in correct identified circle. However, in our experiments, this approach fails on large complex images that contain many thousands of edge pixels and potentially dozens of circles that we aim to identify. The edge map in Figure 5 is an example of a 1.4 megapixel image that contains 68,094 edge pixels. This means the search space of possible solutions contains 3.15×10^{14} distinct candidates. This large search space and the previously mentioned limitations are the two primary reasons why HS failed in our experiments with larger, more complex images.

3.2. The Proposed Algorithm. We propose two main changes to the approach discussed in the previous section. Firstly, the size of the search space needs to be limited and it needs to be organised so that basins of attraction surround most local optima and that neighbouring solutions are likely to belong to the same circle. Secondly, instead of using the standard HS algorithm, we use one of the modern variants (see Section 2.3) that is designed to be more robust in discontinuous, highly multimodal search spaces.

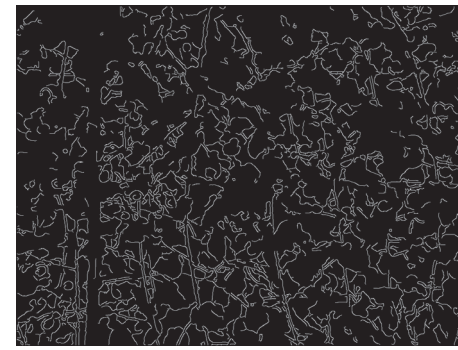


Figure 5: This complex edge map was constructed from a 1386 × 1039 image and contains 68,094 edge pixels.

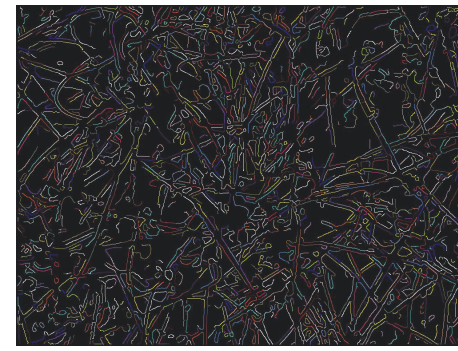


Figure 6: The edge map shown here has been organised into connected neighbourhoods called ribbons.

In order to both limit the size of the search space and improve the shape, we sort the list of edge pixels into connected components and focus our search around a single neighbourhood of edge pixels that are 8-connected neighbours on the image plane. Pixels are 8-connected when they touch horizontally, vertically, or diagonally on the image plane. We call these connected collections of edge pixels *ribbons*. They are combined to form the ribbon list that replaces the edge list to describe the search space. In Figure 6, the edge list is colour coded to illustrate how it may be sorted into multiple ribbons that can be individually considered.

Because we use the Canny edge detector to build our edge map, the edges that are connected are already grouped during the hysteresis thresholding step (see [14]). Therefore, we do not have to separately sort the edge list into separate ribbons. Additionally, the search space is further reduced by removing all the ribbons that are considered too small to form a significant part of any circle. Note that even though these ribbons are removed from the search space for the improvisation of new solutions, the edge pixels inside them are still considered when scoring other solutions. All edge pixels, even those from other ribbons, affect the fitness score of any potential solution. This allows for circles that span multiple ribbons to be included as viable candidates.

We also split long ribbons into smaller ones so that multiple circles may be found on long connected edges. Given that our search space now only consists of short simple ribbons, we can simplify the search space further by defining the circle as only two points and implying the location of the third. Given two points on a single ribbon, we make the assumption that these points represent the start and end points of a circle arc that forms part of the ribbon. The third point required to identify the circle is implied as being half way between the start and end point on the ribbon. This significantly reduces the size of the search space and simplifies the optimisation problem by changing the 3-dimensional problem into a 2-dimensional one.

In addition to the previous modification of the search space, we optionally normalise the fitness function slightly to favour larger circles. Normally the fitness function is normalised by dividing the number of edge pixels that fall on the candidate circle by the circle circumference as defined in (5). This tends to favour smaller circles since fewer pixels are required to define a given ratio of the total circle circumference. In some images, this results in larger circles misidentified as small ones. In these cases, it is better to normalise with a constant factor based on the circumference of the largest circle one expects to find in the image.

The minimum and maximum circles sizes are user parameters that allow the algorithm to focus only on a specific range of circle sizes and help to filter out circles that could be considered as false positives. In our alternate definition of the fitness function, we normalise by the circumference of the maximum circle size so that $f(S)$ is defined as

$$f(S) = \left[\frac{\sum_{i=0}^{C} E(P, s_i)}{C_{\max}} \right], \tag{6}$$

where $C_{\max}$ is circumference of the maximum circle size expected. This is an optional modification of the fitness function and is only used when the application would benefit from a detector that favours larger circles.

In addition to simplifying the search space, we also propose the use of a modern variant of HS that is specifically designed to be more robust when the search space is discrete and discontinuous, for example, labelling problems and integer optimisation. The CHS algorithm is based on HS and was designed to solve the labelling problem associated with the blind deconvolution problem of binary images [28]. Our problem is similar in that our search space is also discrete and discontinuous but we do not need the special improvisation step used in CHS called the *cadenza* operation as this is specific to image deconvolution. Instead we use the standard HS improvisation method. However, we also split the HM into separate islands with special border members like CHS in order to maintain as much diversity in the HM as possible. For more details on CHS and optimising discrete and discontinuous search spaces, see the original CHS paper [28].

Our proposed detection method can be summarised with the following steps:

(1) Use the Canny edge detector to extract the edge map from the source image and sort the edges into connected components called ribbons.

(2) Filter out ribbons that are too short and split ribbons that are too long to both limit the search space and ensure that multiple circles on long continuous edges are not missed.

(3) Use the adapted CHS algorithm to find the best candidate circle on each ribbon in the filtered list.

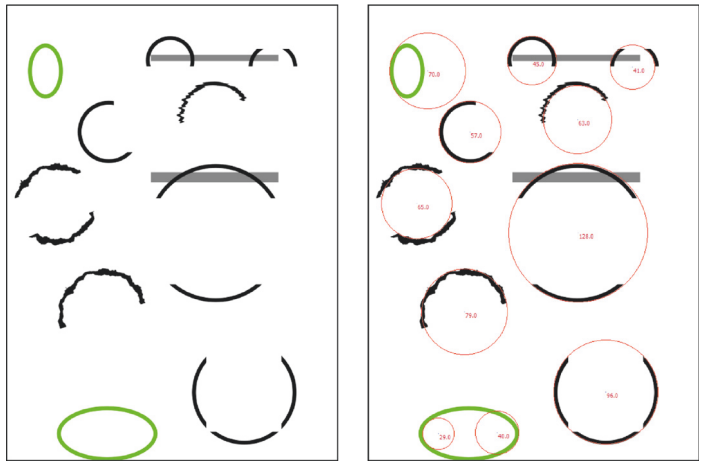
4. Experimental Results

We tested our method using both synthetic images and images captured using normal consumer grade cameras of real objects. Using synthetic images, we can generate scenes with an arbitrary amount of noise and can partially occlude any matches by an arbitrary amount. This allows us to test the robustness and accuracy of the algorithm in a more quantitative way rather than using images captured from real scenes. Conversely, the nonsynthetic images are more realistic and allow us to show potential results in practical applications.

4.1. Synthetic Images. In Figure 7, we show a synthetic image designed to test the robustness of the circle detector. The image contains circle segments of various sizes representing various ratios of a complete circle. Some of the segments are corrupted with rectangular overlays that obscure some edges in the edge map with conflicting edges that do not belong to any circle. Other edges are corrupted with a random pixel offset in both x and y directions causing a wavy arc that has few pixels actually landing on the circle solution. This is made clear in Figure 8 which shows the edge map and highlights how the corruption of the circle arcs results in edges that offer little evidence that they are part of a circle.

In Figure 9, we add Gaussian noise to the test image and redetect the circles using the same algorithm and the same parameter settings. As more noise is added, the edge map becomes more corrupted with edges that do not belong to any circle and overlaps with edges that are part of a circle. Furthermore, edges that were easily identified as part of a circle often get broken into smaller edges which cannot easily be identified by the addition of the noise. We see this effect in Figure 9 particularly in the 20% and 40% noise images.

It is worth noting that, even with 40% Gaussian noise added, most of the circles are still identified correctly. Part of the reason for this robustness is the relatively low quality threshold which is indicative of found circles (see (5)). However, a low quality threshold poses a risk that some edges may be incorrectly identified as circles. We see this happening several times in the noisy images of Figure 9 when small circles are detected on the edges of what should be identified as a much larger circle. This situation seems to indicate that (6) may be a better alternative to the original fitness function since it will suppress the small circles, but this was not the case. In images with greater variation in the expected size of the circles, (5) fails to accurately describe what sufficient



(a) (b)

Figure 7: We use this synthetic test image to test the robustness of the proposed circle detector under various conditions. The red circles in (b) indicate the solutions found by the detector.

Figure 8: This is the edge map of the synthetic image of Figure 7.

evidence for a circle should be. This is especially true when there is a high degree of noise in the image.

4.2. Real Images. We used a selection of images from the Caltech Computer Vision Group archive [15] to test our detector on real images that contain circles or objects that are roughly circular or elliptical. In Figure 10, we show four images that each highlight different features of the algorithm.

Figure 10(a) is a good example of using the detector to find only circles of a specific size. We restricted the detector to circles with a minimum radius of 240 and a maximum of 250 pixels. The detector found all the coins with the smallest radius at 244 and the largest at 247. Figure 10(b) is an example of using the detector to match elliptical objects. Due to the perspective of the image, the bowl is a large ellipse in the edge map but with an appropriate quality threshold the circle detector can approximate the bowl's shape with a circle. In Figure 10(c), we used a low quality threshold and a large range of circle sizes. This allowed at least one noncircular edge to be identified as a circle. In a similar example on Figure 10(d), we used the same circle sizes but a much stricter quality threshold resulting in fewer circles detected but also no false positives. Note that in all the results from these images, we filtered out the detected circles that overlap with other circles that have a higher fitness score.

Our original intention with this work was to address some limitations of the original work done by Cuevas et al. [24] and show that a HS based circle detector can be used in practical applications to detect multiple circles in complex images. One such practical example is automated fruit counting as shown in Figures 11 and 12. In both these examples, the majority of the fruit was correctly identified making it useful as part of a yield estimation system.

In another practical example, the aim is to count cells or other microscopic particles in an image captured from a microscope. In this example (Figure 13), the detector was able to find 3140 circular particles in one image. This is also a good example of where the approach used by Cuevas et al. [24] will likely give poor results since most of the circles share edges since almost all of them touch. In their original work, as mentioned in Section 3.1, Cuevas et al. can find multiple circles by removing the edges of a circle already detected from the edge map before searching for the next one. In situations like those seen in Figure 13, this will likely cause edges from neighbouring circles to be removed as well as making it likely that many circles will not be successfully detected since most of their edges were already removed from the edge map in previous iterations.

5. Conclusion

Based on the original research by Cuevas et al. [24], we propose an alternative approach to using the Harmon Search algorithm for circle detection. Our approach differs in the way that the optimisation problem is structured so that we

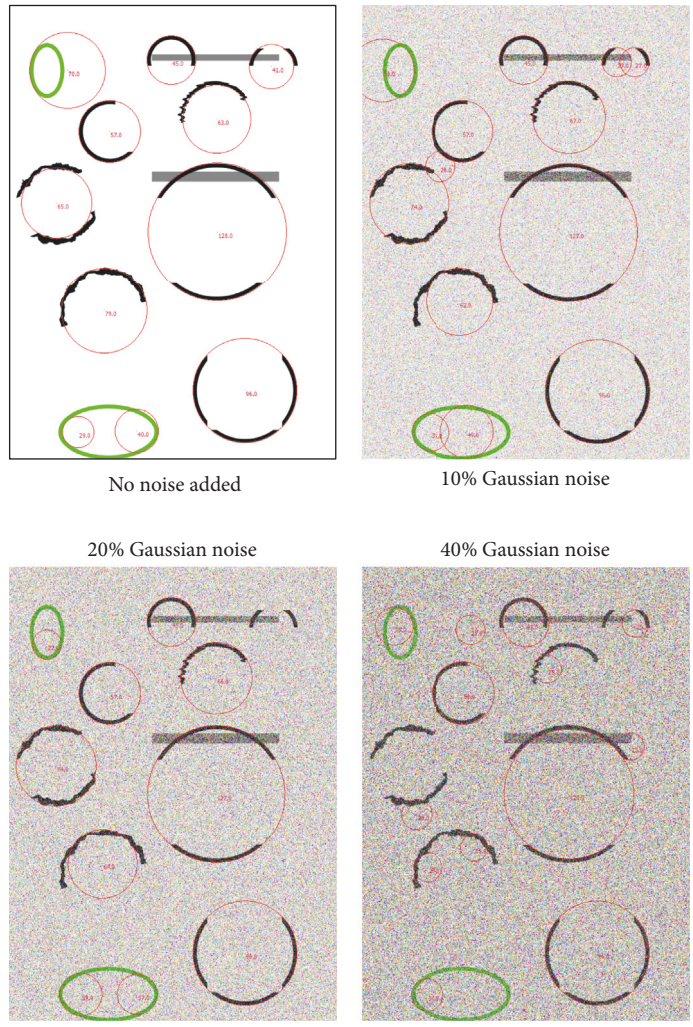


Figure 9: Gaussian noise is added to the synthetic test images.

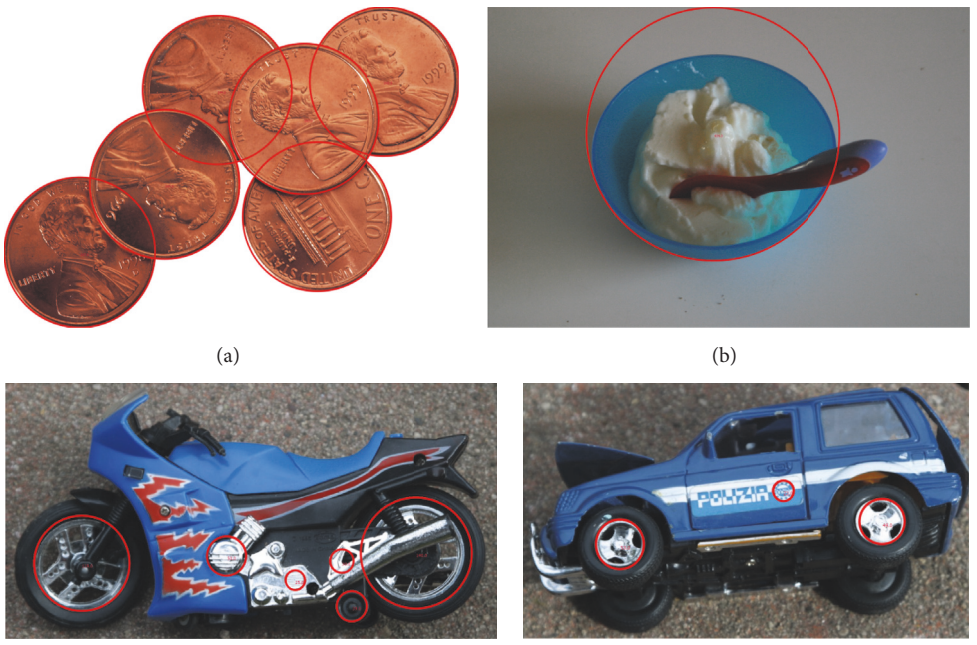
sort the edges into connected components called ribbons which becomes input to the detector in isolation of the other edges in the edge map. This both improves the shape of the search space making it easier to optimise and allows one to accurately find multiple circles in a single image. Instead of using the original Harmony Search optimiser we use a variant based on one of the modern improvements to Harmony Search that is designed for search spaces similar to what we find in the circle detection problem.

We showed through various examples that our approach leads to accurate results that are robust to noise as well as incomplete or imperfect circles. We also highlighted some practical applications that demonstrate our detector's ability to find thousands of circles in images that will be challenging to find multiple circles using the approach used by Cuevas et al. [24]. However, it should be noted that our approach does not improve on the results of Cuevas et al. [24] in all examples. Instead it should be thought of as an alternative that could be more effective in specific applications.

5.1. Future Work. The main aim of this paper is to build on the research done by Cuevas et al. [24] and give an alternate approach to circle detection using Harmony Search. However, there are several ways to potentially improve what we started. One example is to improve the robustness to imperfect circles by expanding the algorithm to detect ellipses instead of circles. The same set of connected and sorted edges can be used but instead of 3 points, 5 points are now needed to define a unique ellipse. This would turn the optimisation problem into a 5-dimensional problem that can be optimised using the same Harmony Search variant used in this paper.

Another less complex improvement involves regularising the fitness function of (5) by adding various penalty terms. For example, the presence of extraneous edges inside a circle candidate or edges that cross the circle's perimeter could be made into a penalty term that discourages the detector to find circles in noisy areas of the image making it less likely to be confused by a dense edge map.

The fitness function could also be improved by taking into account the gradient of the edges when deciding which edges



(a) (b)

(c) (d)

Figure 10: Various test images that highlight different aspects of the circle detector. Figures (b), (c), and (d) come from the image archive of the Computer Vision Group at Caltech [15] and (a) comes from the free clip art repository at http://pngimg.com.

Figure 11: In this example, the circle detector is used to count fruit in order to estimate yield.

Figure 12: Another example of our circle detector used to count fruit.

would be considered evidence for a particular candidate circle. One would expect that an edge that forms part of a real circle would have a similar gradient as the candidate circle at the point of overlap and if the gradient differs by too much the edge can safely be rejected as evidence of a circle.

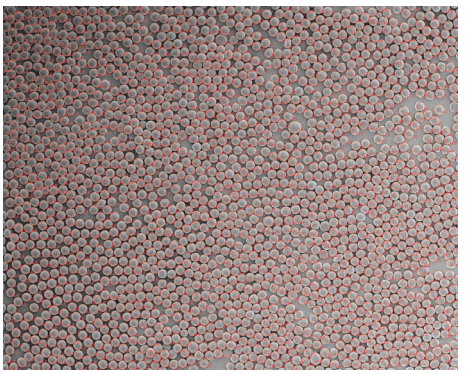


Figure 13: In this example, the circle detector is used to count cells in an image captured from a microscope.

Conflicts of Interest

The author declares that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was partially funded through a grant from the New Zealand Ministry of Business Innovation and Employment under the research grant, Optimum N Nitrogen Sensing and Management (Contract no. CONT29854BITR_LVL).

References

[1] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[2] J. Illingworth and J. Kittler, "A survey of the hough transform," *Computer Vision, Graphics and Image Processing*, vol. 44, no. 1, pp. 87–116, 1988.

[3] J. Han, L. Koczy, and T. Poston, "Fuzzy hough transform," in *Proceedings of the 2nd International Conference on Fuzzy Systems*, pp. 803–808, IEEE, San Francisco, Calif, USA.

[4] D. Shaked, O. Yaron, and N. Kiryati, "Deriving stopping rules for the probabilistic hough transform by sequential analysis," *Computer Vision and Image Understanding*, vol. 63, no. 3, pp. 512–526, 1996.

[5] L. Xu, E. Oja, and P. Kultanen, "A new curve detection method: randomized Hough transform (RHT)," *Pattern Recognition Letters*, vol. 11, no. 5, pp. 331–338, 1990.

[6] J. Iivarinen, M. Peura, J. Srel, and A. Visa, "Comparison of combined shape descriptors for irregular objects," in *Proceedings of the 8th British Machine Vision Conference*, pp. 430–439, University of Essex, London, UK, 1997.

[7] I. Habibie, A. Bowolaksono, R. Rahmatullah et al., "Automatic detection of embryo using particle swarm optimization based hough transform," in *Proceedings of the International Symposium on Micro-NanoMechatronics and Human Science, MHS '13*, IEEE, Nagoya, Japan, November 2013.

[8] V. Ayala-Ramirez, C. H. Garcia-Capulin, A. Perez-Garcia, and R. E. Sanchez-Yanez, "Circle detection on images using genetic algorithms," *Pattern Recognition Letters*, vol. 27, no. 6, pp. 652–657, 2006.

[9] K.-Y. Huang and Y.-L. Chou, "Simulated annealing for hierarchical pattern detection and seismic applications," in *Proceedings of the International Joint Conference on Neural Networks, IJCNN '08*, pp. 1257–1264, IEEE, Hong Kong, China, June 2008.

[10] S. Dasgupta, A. Biswas, S. Das, and A. Abraham, "Automatic circle detection on images with an adaptive bacterial foraging algorithm," in *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, GECCO '08*, p. 1695, ACM, Atlanta, Ga, USA, July 2008.

[11] Z. W. Geem, *Music-Inspired Harmony Search Algorithm: Theory and Applications*, vol. 191, Springer Publishing Company, Berlin, Heidelberg, 1st edition, 2009.

[12] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[13] X. Z. Gao, V. Govindasamy, H. Xu, X. Wang, and K. Zenger, "Harmony search method: theory and applications," *Computational Intelligence and Neuroscience*, vol. 2015, Article ID 258491, 10 pages, 2015.

[14] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[15] Caltech Computer Vision Group Image Archive, "Electronic image archive," 2005, http://www.vision.caltech.edu/html-files/archive.html.

[16] K. P. Philip, E. L. Dove, K. B. Chandran, D. D. McPherson, N. L. Gotteiner, and W. Stanford, "The fuzzy hough transform-feature extraction in medical images," *IEEE Transactions on Medical Imaging*, vol. 13, no. 2, pp. 235–240, 1994.

[17] E. Cuevas, D. Zaldivar, M. P, and M. Ram, "Circle detection using discrete differential evolution optimization," *Pattern Analysis and Applications*, vol. 14, no. 1, pp. 93–107, 2011.

[18] J. Fourie, S. Mills, and R. Green, "Harmony filter: a robust visual tracking system using the improved harmony search algorithm," *Image and Vision Computing*, vol. 28, no. 12, pp. 1702–1716, 2010.

[19] Z. W. Geem, Ed., *Recent Advances In Harmony Search Algorithm*, vol. 270 of *Studies in Computational Intelligence*, Springer, Berlin, Germany, 1st edition, 2010.

[20] O. M. Alia, R. Mandava, D. Ramachandram, and M. E. Aziz, "Dynamic fuzzy clustering using harmony search with application to image segmentation," in *Proceedings of the 9th IEEE International Symposium on Signal Processing and Information Technology, ISSPIT '09*, pp. 538–543, IEEE, Ajman, UAE, December 2009.

[21] Z. Geem, "Harmony search algorithm for solving sudoku," in *Knowledge-Based Intelligent Information and Engineering Systems*, B. Apolloni, R. J. Howlett, and L. Jain, Eds., vol. 4692, pp. 371–378, Springer, Berlin, Germany, 2007.

[22] Z. W. Geem, K. S. Lee, and Y. Park, "Application of harmony search to vehicle routing," *American Journal of Applied Sciences*, vol. 2, no. 12, pp. 1552–1557, 2005.

[23] Z. W. Geem and J.-Y. Choi, "Music composition using harmony search algorithm," in *Proceedings of the 2007 Evo Workshops*, pp. 593–600, Springer-Verlag, Berlin, Heidelberg, Germany, 2007.

[24] E. Cuevas, N. Ortega-S, D. Zaldivar, and M. Prez-Cisneros, "Circle detection by harmony search optimization," *Journal of Intelligent & Robotic Systems*, vol. 66, no. 3, pp. 359–376, 2012.

[25] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems Journal*, vol. 4, no. 1, pp. 25–30, 1965.

[26] V. Tirronen and F. Neri, "Differential evolution with fitness diversity self-adaption, studies in computational intelligence," in *Nature-Inspired Algorithms for Optimisation*, vol. 193, pp. 199–234, Springer, Berlin, Heidelberg, Germany, 2009.

[27] D. H. Ackley, *A Connectionist Machine for Genetic Hillclimbing*, Kluwer Academic Publishers, Norwell, Mass, USA, 1987.

[28] J. Fourie, R. Green, and S. Mills, "Counterpoint harmony search: an accurate algorithm for the blind deconvolution of binary images," in *Proceedings of the International Conference on Audio, Language and Image Processing, ICALIP '10*, pp. 1117–1122, IEEE, Shanghai, China, November 2010.