Pattern Recognition 172 (2026) 112407

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

# Leveraging synthetic data for zero–shot and few–shot circle detection in real–world domains

Paolo Andreini [a,*], Marco Tanfoni [a,b], Simone Bonechi [a], Monica Bianchini [a]

[a] Department of Information Engineering and Mathematics, University of Siena, Siena, Italy
[b] Department of Agriculture, Food, Environment and Forestry, University of Florence, Florence, Italy

ARTICLE INFO

*Keywords:*
Computer vision
Circle detection
Object detection
Style transfer
Zero-shot learning
Few-shot learning

ABSTRACT

Circle detection plays a pivotal role in computer vision, underpinning applications from industrial inspection and bioinformatics to autonomous driving. Traditional methods, however, often struggle with real–world complexities, as they demand extensive parameter tuning and adaptation across different domains. In this paper, we present the Synthetic Circle Dataset (SynCircle), a large synthetic image dataset designed to train a YOLO v10 network for circle detection. The YOLO v10 network, pre–trained solely on synthetic data, demonstrates remarkable off–the–shelf performance that surpasses conventional methods in various practical scenarios. Furthermore, we show that incorporating just a few labeled real images for fine–tuning can significantly boost performance, reducing the need for large annotated datasets. To promote reproducibility and streamline adoption, we publicly release both the trained YOLO v10 weights and the full SynCircle dataset.

## 1. Introduction

Object detection and localization are fundamental tasks in computer vision, underlying applications such as industrial inspection [1], bioinformatics [2], and autonomous driving [3]. Within this field, the detection of circular objects holds unique significance, a fact underscored by the inclusion of dedicated circle–finding functions in major libraries like OpenCV and MATLAB's Image Processing Toolbox. In industrial applications, circular rotating elements are of great use, while in natural sciences, many life forms have circular shapes, since they optimize the area/perimeter ratio [4]. Finally, in biomedical applications, circular objects appear in many problems, ranging from iris [5] and glomeruli detection [6,7] to white blood cell segmentation [5,6,8,9].

In this paper, we propose a novel circle detection framework that leverages a heterogeneous synthetic dataset to train a state–of–the–art localization network. This method effectively bridges the domain gap to real–world images, enabling accurate out–of–the–box detection and rapid fine–tuning for new datasets when needed.

Despite the apparent simplicity of circles as geometric entities, robust detection in real–world images is challenging. Traditional methods often require intensive parameter tuning and domain–specific adjustments, which limit their adaptability across different contexts.

Recent advances in deep learning could address these challenges by leveraging huge datasets to train large models; however, collecting and annotating massive amounts of real–world data remains prohibitively difficult and expensive in many cases. To overcome this limitation, we collected the Synthetic Circle (SynCircle) dataset, a new large collection of labeled images that can be used to train networks for circle detection. Although this study is centered on circle detection, the underlying framework is inherently generalizable to a broader range of geometric shapes. In particular, the synthetic image generation pipeline is structured to support the creation of additional shapes, including squares and triangles.

The SynCircle dataset is constructed in a three–phase process to simulate a wide range of conditions (see Fig. 1). During geometric image generation, empty images with varying background colors are populated with a diverse set of geometric shapes, including circles, ellipses, rectangles, squares and triangles. These shapes are rendered with random colors, can slightly overlap, and may be concentric, thereby simulating many of the complexities found in natural scenes. Some examples of images generated during this phase are depicted in Fig. 2 (Top). Such images are then processed through a style transfer network. By transferring textures and color patterns from a dataset of reference style images, we endow the synthetic images with a high degree of variability in appearance (see Fig. 2 (Bottom)). This step is crucial for bridging the gap between artificially generated images and the rich, complex textures found in real–world scenes. Although the images were created synthetically and the exact positions of the shapes were established a priori, the style transfer process may slightly alter the edges of the object. To address this issue, a refinement network is employed to adjust and

* Corresponding author.
*E-mail addresses:* paolo.andreini@unisi.it (P. Andreini), simone.bonechi@unisi.it (S. Bonechi).

https://doi.org/10.1016/j.patcog.2025.112407
Received 30 April 2025; Received in revised form 14 July 2025; Accepted 2 September 2025
Available online 4 September 2025

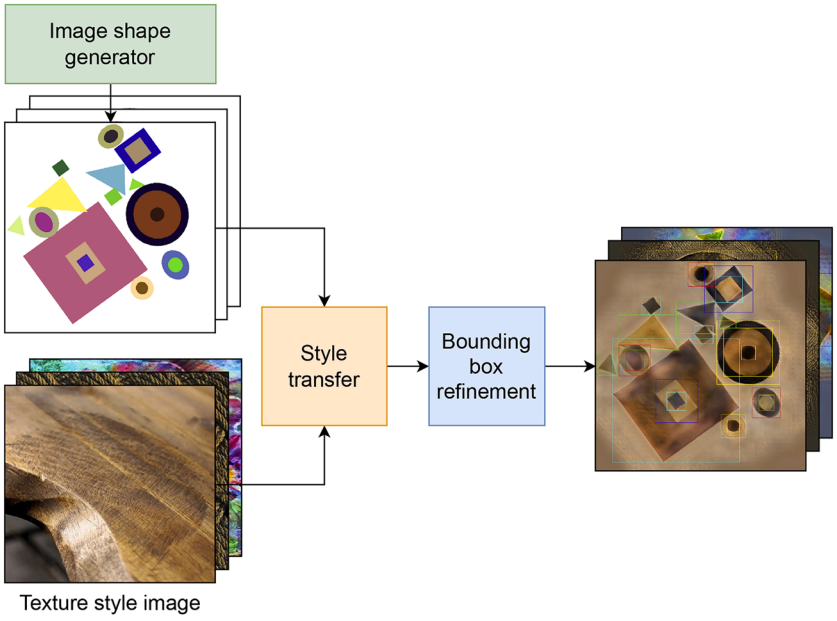


**Fig. 1.** Synthetic image generation procedure: The image shape generator creates images containing random shapes placed on a uniform background. A style transfer step is then applied to introduce diverse textures and colors. Finally, a bounding box refinement stage adjusts the target annotations to account for any alterations introduced during the style transfer process.

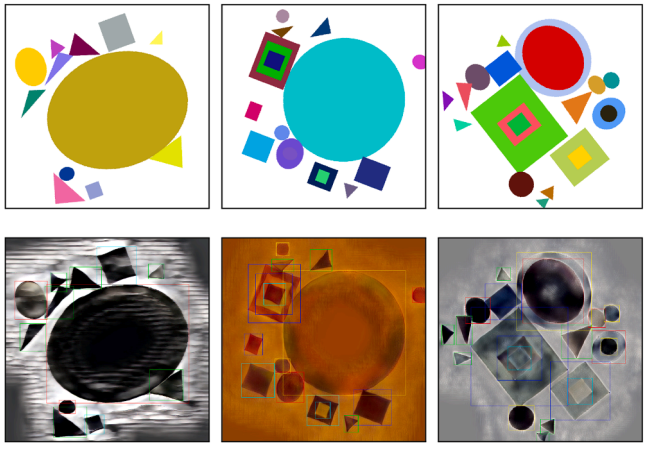

**Fig. 2.** (Top) Images containing different shapes and colors, generated through the image shape generator. (Bottom) Images obtained after the style transfer application. Bounding boxes are drawn in different colors for different shapes: yellow for circles, red for ellipses, green for triangles, light blue for squares, and blue for rectangles. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

accurately restore the positions of the bounding boxes, ensuring that the training targets remain precise. Some examples of generated images are shown in Fig. 2.

A key advantage of the synthetic dataset collection is that the exact location of each object is readily available, which simplifies the process of generating accurate training targets. Moreover, our experiments reveal that fine–tuning the pre–trained network, with varying amounts of real labeled data, significantly increases performance. Interestingly, even a small amount of real data is sufficient to adapt the network effectively to the target domain, underscoring the efficiency of our method. Further, the proposed approach demonstrates significant advantages in scalability and adaptability, two crucial aspects for real–world deployment. Specifically, the network trained on synthetic data can be zero–shot applied on different domains, whereas few images allow to improve the performance on the target data.

To promote reproducibility and encourage future research, we have made our YOLO training and validation code, pre–trained weights, and the generated SynCircle dataset publicly available[1]. Indeed, a versatile and easily fine–tunable circle–detection network could underpin a host of future applications. Such a model could verify bottle–cap alignment on production lines, help autonomous robots identify manhole covers, delineate optic discs in retinal scans, or count bacterial colonies in Petri dishes. Furthermore, the same network could support sports analytics by tracking fast–moving balls, locate craters in planetary imagery, and anchor augmented–reality content to round fiducial markers - all with only minimal, domain–specific fine– tuning.
In summary, the contributions of this paper are as follows:

- We construct a large dataset of synthetic images (SynCircle) containing a variety of textures, colors, and geometric and spatial distributions;
- We train an object detection network that, when pre–trained on SynCircle, outperforms traditional circle detection methods on various datasets;
- We experimentally demonstrate that, on many different real–world datasets, the pre–trained network can be easily adapted via fine–tuning with minimal annotated data;
- We have publicly released the SynCircle dataset and its corresponding YOLO v10 model to encourage adoption of our method and promote full reproducibility.

The remainder of this paper is organized as follows: Section 2 reviews related work and frames our contribution within the existing literature; Section 3 provides a detailed overview of the datasets and models used in this study; Section 4 presents the pipeline for generating synthetic circle images, while Section 5 describes the experimental protocol and reports the performance of the circle–detection model trained on synthetic data and evaluated on real images in zero–shot and few–shot scenarios. Finally, Section 6 summarizes our contributions and highlights directions for future research.

[1] https://github.com/pandrein/SynCircle

## 2. Related works

Circle detection is a fundamental task in computer vision, with applications in industrial inspection, medical imaging, and autonom-ous systems. Over the years, a wide range of techniques have been proposed, broadly categorized into traditional computer vision methods and more recent deep learning–based approaches. Each strategy presents unique trade–offs in terms of accuracy, computational complexity, and adaptability to challenging environments.

The classical Hough Transform (HT) remains a foundational method for circle detection [10], prized for its robustness but limited by its high computational and memory demands. Several optimized versions aim to alleviate these constraints. The Randomized Hough Transform (RHT) introduces random sampling to reduce computational cost while maintaining detection quality [11]. Refinements like circular power theory and mobile–optimized variants, such as real–time smartphone adaptations [12], further improve practicality in real–time applications. Yet, these methods continue to face limitations under high noise and complex background scenarios. Building on RHT, Wang [13] combines randomized sampling with a fitting procedure to suppress bias and invalid detections, achieving improved precision and noise resilience. Similarly, Yao and Yi [14] proposes the Curvature Aided Hough Transform (CACD), which estimates circle radii based on curvature, offering significant speed–ups - up to eightfold over standard HT in cluttered environments. In pursuit of scalability, parallel implementations such as MapReduce–based HT [15] and multithreaded Circular Hough Transform [16] allow the technique to function efficiently in high–throughput or real–time systems, albeit requiring advanced computing infrastructures.

Voting and geometric strategies offer alternative means to address the circle detection challenge. Methods such as Probabilistic Pairwise Voting [5] and 1D edge–based voting [17] aim to reduce computational complexity and enhance resilience to noise, although they may falter in scenes with multiple or overlapping circles. Several techniques target specific challenges: Ou et al. [18] addresses noise with specialized pre–processing, while [19] compensates for perspective distortions. Rapid detection of small structures is tackled by [20], and subpixel accuracy is achieved by blurred edge models in [21], though at higher computational cost.

In addition, a distinct method leveraging random sampling without relying on HT was introduced by [22]. The Randomized Circle Detection (RCD) approach uses randomly selected edge pixels to hypothesize circle candidates, offering lower memory usage and faster execution under moderate noise conditions. Heuristic and metaheuristic algorithms provide yet another avenue. The Harmony Search algorithm [23], Clonal Selection Algorithm [24], and the Adaptive Artificial Fish Swarm Algorithm [25] demonstrate strong performance in terms of robustness and speed, although their dependence on parameter tuning and limited scalability can hinder general applicability. Additionally, Lu et al. [26] proposes a robust method that uses arc–support line segments, polarity analysis, and a dual–stage fitting process, achieving high detection accuracy through geometric consistency checks and spatial constraints. Several approaches focus on contour organization and region segmentation using graph–based models. In [27], ellipse candidates are derived by extracting arc segments through depth–first search and grouping them via geometric constraints. Clustering validates candidates on both synthetic (occluded, overlapping, noisy, concentric, and concurrent ellipses) and real–world datasets. A method grounded in information compression, converting edge maps into a compact set of “information points”, which encodes geometric parameters and verification data, is introduced in [28]. An occlusion–aware detection by grouping arcs based on inscribed triangles and applying a Theil–Sen estimator for robust parameter fitting, followed by a strict geometric verification stage, is presented in [29]. Instead, in a follow–up work, Zhao et al. [30] presents a technique based on coherent chord computation and projective cross–ratio verification. This method enhances robustness to occlusion and noise by grouping arcs using intersecting chord geometry, avoiding differential calculations.

Finally, Deep Learning has emerged as a powerful alternative for robust circle and ellipse detection in unconstrained environments. Convolutional Neural Networks (CNNs), in particular, have demonstrated remarkable adaptability and precision, especially under noisy, cluttered, or non–ideal conditions. For example, Ercan et al. [31] and [32] apply CNN–based architectures to underwater imagery, effectively handling distortions and partial occlusions. These approaches excel in scenarios where traditional methods fail but require large annotated datasets and high computational resources for both training and inference. For a comprehensive overview of deep learning techniques tailored to low–data contexts — including their challenges, solutions, and real–world applications — see [33] and [34].

Traditional heuristic–based approaches can perform well in narrowly defined domains, but they lack flexibility and require painstaking parameter tuning. By contrast, deep–learning methods can be retargeted to new domains, though they generally need large, annotated datasets for effective fine–tuning.

Our work aims to improve upon previous methods by pre–training a deep model on synthetic data, so that it immediately works in different scenarios without having to resort to large annotated sets. Moreover, with only a handful of real samples, it can be rapidly adapted to new domains via few–shot learning.

## 3. Materials

This section introduces the datasets (Section 3.1) and the deep learning methods (Section 3.2) used for generating the SynCircle dataset and for training the circle detection model.

### 3.1. Datasets

#### 3.1.1. Describable textures dataset

The Describable Textures Dataset (DTD) is a publicly available image collection specifically created to advance the analysis and understanding of textures through attributes inspired by human perception. It contains 5640 images, organized into 47 texture categories such as “polka dotted”, “line–like”, “irregular”, and “noisy”. Each category consists of 120 representative images, primarily sourced from Google and Flickr and curated to ensure that at least 90 % of each image surface corresponds to the designated textural attribute.

The images range in size from $300 \times 300$ to $640 \times 640$ and were collected and annotated using Amazon Mechanical Turk [35], where each image is associated with both key and joint perceptual attributes. The dataset includes predefined splits for training, validation, and testing, each containing 40 images per category, with 10 total splits available for reproducible evaluations. In the context of our work, the textures contained in the dataset are transferred by the style transfer algorithm to produce synthetic images. Since our goal is to detect circular patterns, to avoid that circular objects were unintentionally produced by the style transfer algorithm, we manually removed images that feature prominent circular shapes.

#### 3.1.2. Datasets used for evaluation and fine–tuning

*PCB.* The PCB image dataset [26] consists of 100 industrial PCB images, each containing at least one manually annotated circular feature. The images present challenges such as noise, blur, occlusions and overlighting, providing a realistic benchmark for evaluating circle detection methods.

*GH.* A challenging real–world dataset [29] comprising 258 grayscale images captured from diverse scenarios. Blurred boundaries, occlusions from various objects, and significant variations in circle radii collectively increase the difficulty of the detection task.

*MY.* A set of 111 real–world images [29] collected using smartphones across various scenarios, including indoor and outdoor environments, both during the day and at night. All images in the MY dataset are in color. Compared to the GH dataset, MY contains more circles per image and features greater perspective distances. Due to non–perpendicular camera angles, near–elliptic circles may also appear.

*Prasad.* The dataset [36] comprises 400 real–world images drawn from 48 object categories in the Caltech–256 [37] repository. These images pose significant challenges — contours of elliptical shapes are often degraded by complex backgrounds, varying illumination, partial occlusions, noise, shadows and specular reflections. For each image, 20 independent volunteers marked all recurring ellipses; any ellipse cluster receiving more than 10 votes was accepted as part of the global ground truth. Across the 400 images, the validated ellipse count per image varied between 1 and 60.

### 3.2. Network models

Sections 3.2.1 and 3.2.2 describe the two deep learning models involved in the collection of the SynCircle dataset, while Section 3.2.3 details the object detection network used to train the circle–detection model.

#### 3.2.1. Whitening and coloring transform (WCT)

The Whitening and Coloring Transform (WCT) method is utilized to achieve universal style transfer by directly aligning the statistical properties of deep features extracted from content and style images. In this approach, a pre–trained VGG–19 network is used as the encoder to extract content features, which are then centered and decorrelated through a whitening transform that removes the original style information while preserving the global structure. Subsequently, the whitened features are reintroduced to style information via a coloring transform that adjusts their covariance to match that of the style image, thereby transferring salient stylistic patterns in a feedforward manner, without the need for additional style–specific training. This transformation, which hinges on eigenvalue decompositions of the respective covariance matrices, effectively captures a wide variety of styles and easily adapts to novel ones. In this work, style transfer via WCT is applied to transfer textures on synthetic images.

#### 3.2.2. CascadePSP segmentation refinement

CascadePSP is a segmentation refinement framework that progressively improves coarse segmentation outputs through a cascade of global–to–local refinement stages. It leverages pyramid pooling to integrate multi–scale contextual information and recovers fine boundary details, enabling the correction of segmentation errors without requiring high–resolution training data. By sequentially refining the segmentation mask, CascadePSP is able to capture both the overall structure and precise edge information, which enhances the accuracy of the predicted segmentation. In our pipeline, the refined segmentation outputs produced by CascadePSP are subsequently used to improve the labeling accuracy.

#### 3.2.3. YOLO v10

YOLO v10 [38] is a state–of–the–art model designed for real–time end–to–end object detection, significantly advancing the popular YOLO (You Only Look Once) series. It addresses key limitations present in earlier YOLO models — particularly computational redundancy and the dependency on non–maximum suppression (NMS) — by introducing a novel consistent dual–assignment training strategy that eliminates the need for NMS during inference. YOLO v10 further incorporates a holistic efficiency–accuracy–driven design, optimizing multiple architectural components such as light-weight classification heads, spatial–channel decoupled downsampling, compact inverted blocks, and partial self–attention modules. This careful architecture optimization substantially enhances computational efficiency without sacrificing detection accuracy. In this work, YOLO v10 was selected primarily due to its balance between top–tier detection performance and relatively modest computational demand. YOLO v10 comes in different variants (n, s, m, b, l, x), pre–trained on the Microsoft COCO object detection dataset. In this work, we used the x variant (29.5 millions of parameters), which is the largest and top–performing.

## 4. SynCircle dataset generation

The SynCircle dataset is generated following the procedure depicted in Fig. 1 where the main phases of the generation are the following.

*Geometric Image Generation.* Empty white images are created and populated with a diverse set of geometric shapes including circles, ellipses, rectangles, squares and triangles with different colors. The shapes are colored with random colors and randomly positioned in the image; moreover, they may partially overlap or be arranged concentrically. This variability mimics natural scenarios where objects can occlude one another or appear in close proximity. In particular, a random number between 1 and $n$ shapes are rendered on the same image; the shapes can be rotated (with an angle from 0 to 360°), can be concentric (three at most), and can overlap for at least $p$ percent. In this study, we empirically set the parameters to $n = 20$ and $p = 5$ based on visual inspection of some selected samples.The pseudocode of the generation pipeline is described in Algorithm 1.

**Algorithm 1** Image shape generator.

```
Num_shapes ← n                                  ▷ Max number of shapes to render
for i = 1 to Num_shapes do
    S ← {ellipse, rectangle, triangle}
    N_c ← random integer in {0, 1, 2}           ▷ Num. concentric shapes
    for j = 0 to N_c do
        insertable ← False
        while not insertable do
            pos ← random shape position
            C ← random color
            if shape S is insertable in pos then
                insertable ← True
            end if
        end while
    end for
    Render S (N_c concentric shapes, color C, position pos)
end for
α ← random angle
Rotate shapes relative targets with angle α
Save the final image and targets (bounding boxes, masks)
```

*Style Transfer.* The generated images are processed through a style transfer network (Section 3.2.1). By utilizing a separate dataset of reference style images (Section 3.1.1), this network transfers different textures and color patterns onto the images, resulting in a high variability of appearance. This step is essential for reducing the gap between synthetic and real–world imagery.

*Bounding Box Refinement.* Although the geometric image generation step provides exact object locations, the subsequent style transfer can slightly alter object boundaries. To correct these distortions, a refinement network [39] is applied to adjust the object bounding boxes. This ensures the accuracy of the synthetic annotations is maintained after the stylistic modifications. In particular, the refinement network takes in input, for each shape generated by the Geometric Image Generator, the stylized image and the shape segmentation mask, and produces a

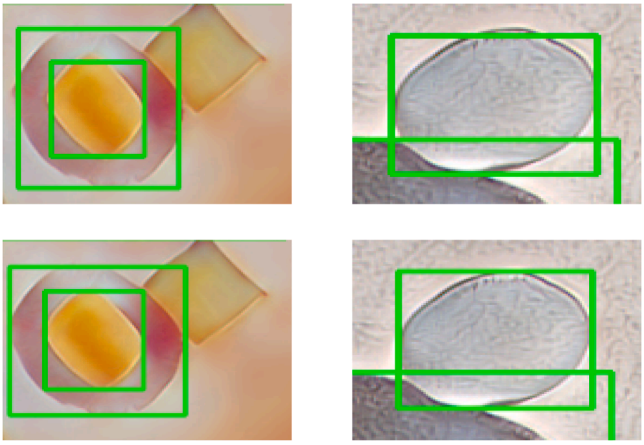

**Fig. 3.** Example of the effect of the CascadePSP refinement: original images and bounding boxes (top row) versus refined bounding boxes (bottom row).

refined mask. The min enclosing bounding box of the refined mask is computed and used in place of the original. Some qualitative examples of the refinement procedure are given in Fig. 3.

The above approach is used to create the SynCircle dataset that comprises a large set of synthetic images targeted to recognise circular shapes. The generated images have a resolution of $720 \times 720$ pixels and contain a variable number of possibly overlapping and concentric geometric shapes with different textures and colors. Each circular shape is labeled with its bounding box coordinates that are provided as targets. The dataset collects 54,147 images that are divided into training, validation, and test sets, containing 52,644, 546, and 957 images, respectively. The goal of the dataset is to provide a ready–to–use set of images to train, and possibly evaluate, machine learning systems for circle detection.

## 5. Experiments and results

In this section, the SynCircle dataset is used to train a circle detection network based on YOLO v10. The network is first trained on synthetic data and evaluated on real–world data in a zero–shot setting. Subsequently, the pre–trained model is fine–tuned using a small subset of real images to assess its performance in a few–shot scenario.

All experiments were conducted on a workstation equipped with an Intel® Core™ i9–13900K CPU, 128 GB of RAM, and a single NVIDIA GeForce RTX 4090 GPU.[2]

The experiments pursue two main objectives: (1) to evaluate how well the model trained on SynCircle generalizes to real datasets without additional training (Section 5.1); and (2) to assess the model performance when fine–tuned on a limited number of real images (Section 5.2).

The performance of the models is assessed with the mean Average Precision at IoU 0.5 ($mAP_{50}$) and the $F_1$–score.

$mAP_{50}$ is a common metric for evaluating the accuracy of object detectors. Detections $\hat{b}_i$ are ranked by confidence and greedily matched to ground–truth boxes $b_j$ in the same image. A detection is a true positive (TP) if $\mathrm{IoU}(\hat{b}_i, b_j) \geq 0.5$; otherwise it is a false positive (FP). Accumulating TP and FP yields a precision-recall curve $P_c(r)$ for each class $c$ ($r \in [0, 1]$):

$$AP_{50}(c) = \int_0^1 P_c(r)\, dr, \qquad mAP_{50} = \frac{1}{K} \sum_{c=1}^{K} AP_{50}(c).$$

In our setup $K = 1$ (only the class *circle*), so $mAP_{50} = AP_{50}$.

[2] The model, using our computer, requires an average inference latency of 23 ms on the GPU, and 358 ms on the CPU.

**Table 1**
Comparison of F–measure across different methods and datasets.

| Method | GH | MY | PCB | Prasad |
|---|---|---|---|---|
| Chen T.-C. et al. [22] | 0.12 | 0.12 | – | – |
| Yao Z. et al. [14] | 0.54 | 0.54 | – | – |
| Lu C. et al. [26] | – | – | **0.97** | – |
| Wang G. et al. [13] | 0.29 | 0.25 | – | – |
| Shen Z. et al. [27] | – | – | 0.79 | 0.46 |
| Zhao M. [29] | **0.75** | 0.69 | 0.95 | – |
| Ou Y. et al. [28] | 0.70 | – | 0.93 | – |
| Ou Y. et al. [18] | 0.64 | 0.59 | – | – |
| Zhao M. [30] | – | – | 0.88 | 0.51 |
| **Ours (zero–shot)** | 0.69 | **0.77** | 0.92 | **0.68** |

The $F_1$–score is a metric that combines precision and recall into a single score to evaluate the performance of a classification model, particularly in scenarios with imbalanced datasets. Sweeping the confidence thresholds, the same matching rule provides TP, FP and false negatives (FN):

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \qquad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

$$F_1 = \frac{2\,\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

The maximum $F_1$–score across thresholds is reported to evaluate our model.

### 5.1. Zero–shot learning on real data

YOLO v10x is trained on the SynCircle dataset for a maximum of 100 epochs, using early–stopping on the validation set. Following the training hyperparameters specified in the original YOLO v10 paper [38], we used the SGD optimizer with a momentum of 0.9 and a weight decay of $5 \times 10^{-4}$. The learning rate was initialized at 0.01, linearly warmed up over the first three epochs, and then gradually decayed to 1 % of its initial value by the final epoch. The model is evaluated across a range of real–world datasets. Since these datasets are not involved in the training process, all their images are used exclusively for testing. The results, summarized in Table 1, are compared against state–of–the–art methods tailored to each specific dataset.

Remarkably, without any fine–tuning, the model pre–trained solely on the SynCircle dataset achieves performance that is comparable to or even surpasses that of existing methods across multiple domains. This demonstrates the strong generalization capability of the proposed approach. It is important to highlight that competing methods typically rely on carefully tuned hyperparameters, which are specifically optimized for each target application. In contrast, our model — trained once on the SynCircle dataset with no domain–specific adaptations — performs robustly across diverse real–world settings. This underlines the effectiveness of the SynCircle dataset in capturing domain–invariant features relevant to circle detection.

### 5.2. Few–shot learning on real data

Our experimental design reflects a scenario in which no more than thirty real target–domain images are available for adaptation. To emulate this constraint, we first reserve thirty images from each dataset to form our pool of training and validation examples, while all the other images are used as a fixed test set. Specifically, we compare two fine–tuning strategies, described below.

- **Fixed epochs:** training is performed for a predetermined number of epochs using a limited set of images. In our experiments, we evaluated performance using 5, 10, 15, 20, 25, and 30 images.

**Table 2**
Few–shot learning with a fixed number of epochs using different dimensions for the training set.

| # Training Images | Dataset | Synthetic | | Default | |
|---|---|---|---|---|---|
| | | F1 | mAP@50 | F1 | mAP@50 |
| 0 | GH | 0.7385 | 0.7611 | – | – |
| | MY | 0.7812 | 0.8224 | – | – |
| | PCB | 0.9107 | 0.9402 | – | – |
| | Prasad | 0.6809 | 0.7303 | – | – |
| 5 | GH | **0.5297** | **0.3898** | 0.20945 | 0.0776 |
| | MY | **0.7441** | **0.7594** | 0.2324 | 0.1905 |
| | PCB | **0.9444** | **0.9744** | 0.5081 | 0.6596 |
| | Prasad | **0.5682** | **0.5236** | 0.2010 | 0.1148 |
| 10 | GH | **0.6121** | **0.5785** | 0.4305 | 0.3609 |
| | MY | **0.7345** | **0.7450** | 0.3725 | 0.3422 |
| | PCB | **0.9230** | **0.9595** | 0.8303 | 0.8722 |
| | Prasad | **0.6560** | **0.6420** | 0.4278 | 0.3795 |
| 15 | GH | **0.9307** | **0.9666** | 0.5555 | 0.5134 |
| | MY | **0.7560** | **0.7879** | 0.5762 | 0.5746 |
| | PCB | **0.9307** | **0.9666** | 0.9060 | 0.9356 |
| | Prasad | **0.6711** | **0.6736** | 0.5430 | 0.5264 |
| 20 | GH | **0.7508** | **0.7671** | 0.5991 | 0.5777 |
| | MY | **0.7918** | **0.8257** | 0.6639 | 0.6864 |
| | PCB | **0.9598** | 0.9596 | 0.9135 | **0.9612** |
| | Prasad | **0.6745** | **0.6817** | 0.5670 | 0.5823 |
| 25 | GH | **0.7350** | **0.7543** | 0.5763 | 0.5497 |
| | MY | **0.7652** | **0.7911** | 0.6963 | 0.7237 |
| | PCB | **0.9374** | 0.9626 | 0.9200 | **0.9667** |
| | Prasad | **0.6988** | **0.6973** | 0.6366 | 0.6590 |
| 30 | GH | **0.7489** | **0.7801** | 0.6109 | 0.5645 |
| | MY | **0.7820** | **0.7955** | 0.7021 | 0.7253 |
| | PCB | **0.9628** | **0.9875** | 0.9326 | 0.9512 |
| | Prasad | **0.7128** | **0.7112** | 0.6513 | 0.6784 |

- **Early stopping:** terminate training using the validation set. In this setup, 60 % of the data is randomly chosen for training and the remaining 40 % for validation. Specifically, we evaluate the following splits: 12 training/8 validation, 15 training/10 validation, and 18 training/12 validation images.

When the dataset contains less than 20 images, creating a reliable validation split becomes impractical; therefore, only the fixed–epochs strategy is applied. In both setups, we use a fixed learning rate of $1 \times 10^{-4}$. Table 2 reports the results obtained on the test set using a different number of images to fine–tune the model. For comparison, we also include the results of the zero–shot approach evaluated on the same test set.

In every dataset and for every evaluation metric, models fine–tuned from the pre–trained checkpoint consistently outperform those initialized with the default YOLO weights. Pre–training on synthetic data, therefore, provides a decisive advantage when real data are limited and should be preferred for circle detection tasks.

The results obtained using different training/validation splits to early stop the network training are reported in Table 3.

Comparing the results with and without early stopping, we observe that, overall, early stopping offers limited benefits when only 20 images are available. In particular, the performance achieved without early stopping is comparable to that obtained using a 12/8 training/validation split, and in some cases, even superior. This is likely because, with such a small dataset, it is more effective to utilize all available images for training rather than setting aside a portion for validation to guide the stopping criterion. However, when a larger number of images is available (e.g., 25 or 30), early stopping tends to provide a clearer performance advantage. In these cases, the model has access to enough data to allow for an effective validation split, without severely compromising the amount of training data.

A comparison between the few–shot and zero–shot approaches yields additional insights. Specifically, when only 10 images are available for fine–tuning, the results indicate that this limited amount of data is insufficient to effectively adapt the model. In such a low–data regime, especially without a dedicated validation set, it becomes challenging to prevent overfitting, which negatively impacts generalization. These findings further highlight the effectiveness of the SynCircle dataset: despite being synthetic, it provides training data that enables strong generalization to real–world scenarios when used to pre–train a detection model. Moreover, they demonstrate the practical value of the YOLO v10 model pre–trained on SynCircle (which will be publicly released). This model offers a reliable off–the–shelf solution for circle detection in natural images across diverse domains, even in the absence of additional fine–tuning. Nonetheless, our approach still faces some challenges. First, the performance of the pre–trained networks hinges on the quality of the synthetic data they are fed. Synthetic generation can yield vast datasets that replicate many real–world complexities, yet some domain shift remains almost unavoidable. Narrowing this gap will likely require new data–generation techniques. Second, because the pipeline is deep–learning–based, even modest improvements on a new target domain demand at least a small, supervised fine–tuning set - hand–crafted parameter tweaking, as in earlier heuristic methods, is no longer an option.

**Table 3**
Few–shot learning using different dimensions for both training and validation sets.

| # Training/Val. Images | Dataset | Synthetic | | Default | |
|---|---|---|---|---|---|
| | | F1 | mAP@50 | F1 | mAP@50 |
| 12/8 | GH | **0.7137** | **0.7573** | 0.6731 | 0.7097 |
| | MY | **0.7706** | **0.8061** | 0.7232 | 0.7638 |
| | PCB | **0.9544** | **0.9707** | 0.9472 | 0.9653 |
| | Prasad | 0.6829 | **0.7242** | **0.6916** | 0.7193 |
| 15/10 | GH | **0.7216** | **0.7653** | 0.6194 | 0.6244 |
| | MY | 0.7510 | 0.7715 | **0.7518** | **0.7950** |
| | PCB | **0.9654** | **0.9700** | 0.9503 | 0.9628 |
| | Prasad | **0.6931** | **0.7333** | 0.6539 | 0.6978 |
| 18/12 | GH | **0.7596** | **0.7887** | 0.5992 | 0.5853 |
| | MY | **0.8144** | **0.8444** | 0.7854 | 0.8067 |
| | PCB | 0.9444 | **0.9723** | **0.9537** | 0.9643 |
| | Prasad | **0.6857** | **0.7303** | 0.6781 | 0.7184 |

## 6. Conclusion and future work

In this work, we introduced SynCircle, a richly diverse synthetic dataset designed to understand different shapes and textures, and leveraged it to train a YOLO v10 network for circle detection. Remarkably, when evaluated on real–world images without any additional tuning, our model — trained solely on SynCircle — rivals and often surpasses specialized classical detectors in a zero–shot setting. By incorporating just a handful of annotated real images, we can further fine–tune the network via few–shot learning, rapidly adapting it to new domains and boosting its accuracy. This stands in sharp contrast to traditional circle detection methods, which typically demand painstaking manual hyperparameter adjustments to handle novel environments. Our fully data–driven approach not only streamlines deployment but also delivers greater flexibility.

Throughout the experiments, we demonstrated two key strengths of our method. First, YOLO v10, trained via SynCircle, can be immediately deployed to any target domain, outperforming most conventional algorithms right out of the box. Second, when even a small set of labeled examples ( > 10 images) is available, our model can be quickly refined to achieve substantial performance gains. Looking forward, there remain new research avenues to explore: more sophisticated few–shot strategies might lower the number of real images needed for effective fine–tuning, and broadening the synthetic generator to include a richer array of shapes, backgrounds, and imaging artifacts could further narrow the gap between synthetic and real data.

## CRediT authorship contribution statement

**Paolo Andreini:** Writing – review & editing, Writing – original draft, Supervision, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Marco Tanfoni:** Software, Investigation; **Simone Bonechi:** Writing – review & editing, Software, Investigation, Data curation, Formal analysis, Methodology; **Monica Bianchini:** Writing – review & editing, Validation, Supervision.

## Data availability

The data will be made publicly available

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

During the preparation of this work, the authors used ChatGPT in order to improve the quality of English text. After using this tool/service, the authors reviewed and edited the content as needed and took full responsibility for the content of the publication.

## Funding

This study was co–funded by the European Union – Next Generation EU, in the context of The National Recovery and Resilience Plan – Investment 1.5 Ecosystems of Innovation, Project Tuscany Health Ecosystem (THE), Spoke 3 – Advanced technologies, methods and materials for human health and well–being. ECS00000017, CUP: B63C22000680007.

## References

[1] I. Yousif, L. Burns, F. El Kalach, R. Harik, Leveraging computer vision towards high-efficiency autonomous industrial facilities, J. Intell. Manuf. 36 (2024) 2983–3008.
[2] Y. Megel, A. Kutsenko, I. Blagov, S. Kovalenko, S. Kovalenko, M. Malko, A. Rybalka, Information system for automating processes of biological objects detection, recognition, and measurement, in: 2021 XXXI International Scientific Symposium Metrology and Metrology Assurance (MMA), IEEE, 2021, pp. 1–6.
[3] D. Feng, A. Harakeh, S.L. Waslander, K. Dietmayer, A review and comparative study on probabilistic object detection in autonomous driving, IEEE Trans. Intell. Transp. Syst. 23 (8) (2021) 9961–9980.
[4] S. Chen, R. Xia, J. Zhao, Y. Chen, M. Hu, A hybrid method for ellipse detection in industrial images, Pattern Recognit. 68 (2017) 82–98.
[5] L. Pan, W.-S. Chu, J.M. Saragih, F. De la Torre, M. Xie, Fast and robust circular object detection with probabilistic pairwise voting, IEEE Signal Process. Lett. 18 (11) (2011) 639–642.
[6] E.H. Nguyen, H. Yang, R. Deng, Y. Lu, Z. Zhu, J.T. Roland, L. Lu, B.A. Landman, A.B. Fogo, Y. Huo, Circle representation for medical object detection, IEEE Trans. Med. Imaging 41 (3) (2023) 746–754.
[7] P. Andreini, S. Bonechi, G.M. Dimitri, Enhancing glomeruli segmentation through cross-species pre-training, Neurocomputing 563 (2024) 126947.
[8] H. Sazak, M. Kotan, Automated blood cell detection and classification in microscopic images using YOLOv11 and optimized weights, Diagnostics 15 (1) (2024) 22.
[9] O.K. Bagui, J.T. Zoueu, Red blood cells counting by circular hough transform using multispectral images, J. Appl. Sci. 14 (24) (2014) 3591–3594.
[10] R.O. Duda, P.E. Hart, Use of the hough transformation to detect lines and curves in pictures, Commun. ACM 15 (1) (1972) 11–15.
[11] Q. Zhou, K. Zhang, Z. Zhang, H. Yu, Improved randomized hough transform based on circular power theory, in: 2023 International Conference on Image Processing, Computer Vision and Machine Learning (ICICML), 2023, pp. 288–293. https://doi.org/10.1109/ICICML60161.2023.10424820
[12] V.J. Schneider, Real time circle detection by simplified hough transform on smartphones, in: Real-Time Image Processing and Deep Learning 2021, 11736, SPIE, 2021, pp. 89–102.
[13] G. Wang, A sub-pixel circle detection algorithm combined with improved RHT and fitting, Multimed. Tools Appl. 79 (39) (2020) 29825–29843.
[14] Z. Yao, W. Yi, Curvature aided hough transform for circle detection, Expert Syst. Appl. 51 (2016) 26–33.
[15] M.M.A. Coelho, D.N. Sugimoto, G.A. Melo, V.V. Curtis, J. de Melo Bezerra, A MapReduce based approach for circle detection, in: ICSOFT, 2019, pp. 454–459.
[16] V.K. Yadav, M.C. Trivedi, S.S. Rajput, S. Batham, Approach to accurate circle detection: multithreaded implementation of modified circular hough transform, in: Proceedings of International Conference on ICT for Sustainable Development: ICT4SD 2015 Volume 1, Springer, 2016, pp. 25–34.
[17] K. Chen, J. Wu, One-dimensional voting scheme for circle and arc detection, J. Opt. Soc. Am. A 31 (12) (2014) 2593–2602.
[18] Y. Ou, H. Deng, Y. Liu, Z. Zhang, X. Lan, An anti-noise fast circle detection method using five-quadrant segmentation, Sensors 23 (5) (2023) 2732.
[19] S. Guo, S. Yang, P. Zhang, A circle detection algorithm based on ellipse removal, J. Image Process Theory Appl. 4 (1) (2021) 42–50.
[20] L. Jiang, H. Yuan, C. Li, Circular hole detection algorithm based on image block, Multimed. Tools Appl. 78 (2019) 29659–29679.
[21] W. Liu, X. Yang, H. Sun, X. Yang, X. Yu, H. Gao, A novel subpixel circle detection method based on the blurred edge model, IEEE Trans. Instrum. Meas. 71 (2021) 1–11.
[22] T.-C. Chen, K.-L. Chung, An efficient randomized algorithm for detecting circles, Comput. Vision Image Understanding 83 (2) (2001) 172–191.
[23] J. Fourie, Robust circle detection using harmony search, J. Optim. 2017 (1) (2017) 9710719.
[24] E. Cuevas, V. Osuna-Enciso, F. Wario, D. Zaldívar, M. Pérez-Cisneros, Automatic multiple circle detection based on artificial immune systems, Expert Syst. Appl. 39 (1) (2012) 713–722.
[25] W. Wang, K. Lu, R. Hong, P. Chen, J. Zhang, B. Wang, A machine vision method for automatic circular parts detection based on optimization algorithm, in: Intelligent Computing Theories and Application: 13th International Conference, ICIC 2017, Liverpool, UK, August 7–10, 2017, Proceedings, Part I 13, Springer, 2017, pp. 600–611.
[26] C. Lu, S. Xia, W. Huang, M. Shao, Y. Fu, Circle detection by arc-support line segments, in: 2017 IEEE International Conference on Image Processing (ICIP), IEEE, 2017, pp. 76–80. https://doi.org/10.1109/ICIP.2017.8296254
[27] Z. Shen, M. Zhao, X. Jia, Y. Liang, L. Fan, D.-M. Yan, Combining convex hull and directed graph for fast and accurate ellipse detection, Graph. Models 116 (2021) 101110. https://doi.org/10.1016/j.gmod.2021.101110
[28] Y. Ou, H. Deng, Y. Liu, Z. Zhang, X. Ruan, Q. Xu, C. Peng, A fast circle detection algorithm based on information compression, Sensors 22 (19) (2022) 7267. Published online: 2022-09-25. https://doi.org/10.3390/s22197267
[29] M. Zhao, X. Jia, D.-M. Yan, An occlusion-resistant circle detector using inscribed triangles, Pattern Recognit. 109 (2021) 107588.
[30] M. Zhao, X. Jia, L. Ma, L.-M. Hu, D.-M. Yan, Coherent chord computation and cross ratio for accurate ellipse detection, Pattern Recognit. 146 (2024) 109983. https://doi.org/10.1016/j.patcog.2023.109983
[31] M.F. Ercan, A.L. Qiankun, S.S. Sakai, T. Miyazaki, Circle detection in images: a deep learning approach, in: Global Oceans 2020: Singapore - U.S. Gulf Coast, 2020, pp. 1–5. https://doi.org/10.1109/IEEECONF38699.2020.9389048
[32] T. Hao, D. Xu, Circle area detection based on convolutional neural networks, in: 2022 IEEE International Conference on Mechatronics and Automation (ICMA), IEEE, 2022, pp. 1123–1128.
[33] Y. Song, T. Wang, P. Cai, S.K. Mondal, J.P. Sahoo, A comprehensive survey of few-shot learning: evolution, applications, challenges, and opportunities, ACM Comput. Surv. 55 (13s) (2023) 1–40.
[34] L. Alzubaidi, J. Bai, A. Al-Sabaawi, J. Santamaría, A.S. Albahri, B.S.N. Al-Dabbagh, M.A. Fadhel, M. Manoufali, J. Zhang, A.H. Al-Timemy, et al., A survey on deep learning tools dealing with data scarcity: definitions, challenges, solutions, tips, and applications, J. Big Data 10 (1) (2023) 46.
[35] K. Crowston, Amazon mechanical turk: a research tool for organizations and information systems scholars, in: Shaping the Future of ICT Research. Methods and Approaches: IFIP WG 8.2, Working Conference, Tampa, FL, USA, December 13–14, 2012. Proceedings, Springer, 2012, pp. 210–221.
[36] D.K. Prasad, M.K.H. Leung, S.-Y. Cho, Edge curvature and convexity based ellipse detection method, Pattern Recognit. 45 (9) (2012) 3204–3221.
[37] G. Griffin, A. Holub, P. Perona, et al., Caltech-256 object category dataset, Technical Report, Technical Report 7694, California Institute of Technology Pasadena, 2007.
[38] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, et al., Yolov10: real-time end-to-end object detection, Adv. Neural Inf. Process. Syst. 37 (2024) 107984–108011.
[39] H.K. Cheng, J. Chung, Y.-W. Tai, C.-K. Tang, Cascadepsp: toward class-agnostic and very high-resolution segmentation via global and local refinement, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 8890–8899.