FOUR LEAF CLOVER DETECTION

Michael Sheroubi, Lauren Ramanantsoa
École Polytechnique, Paris, France
{michael.sheroubi, lauren.ramanantsoa}@polytechnique.edu

ABSTRACT

The project aims to identify and classify clovers in images based on the number of leaves, distinguishing between three-leaf and four-leaf variants. We propose three different approaches. First, a machine learning approach builds on the YOLOv7 and Faster R-CNN object detection algorithms to accurately localize and classify clovers in images with noisy backgrounds. Second, two alternative geometric methods are proposed for clover classification: a Laplacian of Gaussian blob detection with automatic scale selection and a floodfill segmentation counting the number of leaves on the clovers. Our results demonstrate that machine learning models achieve high accuracy in detecting clovers despite challenging backgrounds and show potential for real-time video classification. Geometric methods offer complementary insights, providing strategies for leaf counting adapted to images of different qualities.

***K*eywords** Object Detection · Classification · Convolutional Neural Networks · Laplacian of Gaussian · Blob detection · Floodfill Segmentation

1 Introduction

The search for four-leaf clovers has long been associated with luck, stemming from their rarity and cultural significance. From these factors were born clover hunters. These hunters can spend hours, days, or even weeks randomly sampling various clover patches in search of a lucky four-leaf clover. Traditionally, the odds of finding a four-leaf clover are estimated to be about 1 in 10,000, and some studies even suggest that the odds could be closer to 1 in 5,000 depending on environmental conditions and genetic factors. In fact, four-leaf clovers are genetic mutations of the common three-leaf variety, Trifolium repens. These numbers stem from methodical sampling, an approach which clover hunters rarely employ. Fortunately for modern-day clover hunters, they exist in an age where scientific techniques can help them make their own luck.

As far as we know, automated clover detection has not been widely explored in existing literature. This project will introduce a new set of eyes, to drastically speed up the detection and classification process when examining a clover patch. In this project, we propose a range of machine learning and geometric techniques to detect four-leaf clovers. Our paper is structured as follows: Section 2 defines the constraints of our four-leaf clover detection problem. Section 3 proposes to fine-tune a YOLOv7 and Faster R-CNN model for both clover detection and classification. Section 4 discusses two geometric techniques for clover classification: a Laplacian of Gaussian blob detection with automatic scaling and flood fill segmentation.

2 Problem Definition

Our goal is to detect clovers in their natural environments: dense clusters of overlapping clovers, grasses and other flora. To best help clover hunters, we need to develop methods capable of processing images at varying scales, with the potential for real-time video detection in future work. Since four-leaf clovers are rare enough, we seek to detect clovers across their natural variations. Indeed, clover leaves exist in many variations affecting their orientation, shapes, colour patterns, and defining lines. Overall, our methods need to be scale and rotation invariant.

To address this, we divide the problem into two key challenges: first, isolating individual clovers from their noisy environments (clover detection) and second, counting the number of leaves on each clover (clover classification). Our

machine learning approach is able to deal with both steps. It extracts clover bounding boxes which are then used as a starting point for the geometric techniques.

3 CNN Clover Detection and Classification

Convolutional Neural Networks play two key roles in this project. The first is to provide a baseline classification accuracy for our geometric classification algorithms. More importantly, CNN takes care of clover detection, returning a bounding box isolating clovers for classification.

3.1 Models

When faced with a detection and classification problem in images, a common approach would be to use a Convolutional Neural Network (CNN). Although they can achieve great results, the results are not explainable and rely heavily on the existence of a lot of domain-specific annotated data. One way to reduce the amount of domain-specific data is to pre-train a model on a broader array of data, then fine-tuning the model to your specific problem set. That is the approach we employ by fine-tuning two SOTA models; YOLOv7 [6] and Faster R-CNN [5] (ResNet).

Model	**P**	**R**	**mAP@.5**	**mAP@.5:.95**
YOLOv7	0.786	0.779	0.796	0.660
Faster R-CNN	0.655	0.5196	0.421	0.655

Table 1: Classification Accuracy

3.1.1 Model Selection

YOLOv7 and Faster R-CNN are both models that specialize in the object detection and classification. Each model offers its own advantages. **YOLOv7** with its speed and efficiency, making it ideal if the goal is to process images in real-time video capture (this is the best way to help clover hunters). It can handle fairly high frame-rate with impressive accuracy. However, theoretically it would not achieve the same level of precision as multi-stage models like Faster R-CNN. The key to YOLOv7's speed is that the architecture uses a single CNN. It treats object detection as a regression problem, predicting bounding box coordinates and class probabilities directly from the image in one step [6].

While still being fairly fast, **Faster R-CNN** is better at detecting smaller objects in more complex environments. For this it uses multiple CNNs. Firstly a feature extractor using a pre-trained model (in this case ResNet), a Region Proposal Network (RPN) which allows predicts object bounds and scores at each position on the feature map, and then a R-CNN network for object detection. The result is a model that works well for complex images, which makes it very suitable for our task (since leaves overlap and there is a lot of foliage). However, it can be slower and more resource-intensive than YOLOv7, which may limit our ability to run our model in real-time or on a mobile device.

3.2 Datasets

At the time of writing this report, this area of study is not very developed and the existing datasets are sparse. Most of the available data are unlabeled or synthetic. Here is an outline of the main datasets explored:

Four-Leaf Clover (FLC) Dataset [3] This dataset is extensive, comprising approximately 67 GB of data with 100,000 images. It includes 2,151 annotated instances of clovers with four, five, and six leaves. The dataset is divided into training and testing sets sourced from two different clover seasons. It is annotated for tasks such as object detection, semantic segmentation, instance segmentation, object parsing, and semantic boundary detection.

GrassClover (GC) Dataset: With around 73 GB of data, this dataset is also substantial. The annotated images are synthetic, and the sampled data is unlabeled.

Hunting for Four Leaf Clovers (H4FLC) [1]: This is the most concise and topic-specific dataset among the three. It contains approximately 113 MB of data focused on classifying clovers by the number of leaves. The dataset includes 1,062 training images and 131 validation images.

Despite being very large datasets, both FLC and GC do not contain a lot of annotated data, with GC's data being fully synthetic. Due to computational restraints, we will move forward with **H4FLC** for training both our models and testing our geometric approaches. The H4FLC clovers are annotated the clover class and poly points that would be converted into bounding boxes. On top of these datasets, we also collected a sample of images and videos around the L'X campus.

Dataset	Size	Annotations	Total
FLC	$\sim 67Gb$	2151	> 100k
GC	$\sim 71Gb$	8000	$\sim 40k$
H4FLC	113 Mb	$\sim 1.2k$	$\sim 1.2k$

Table 2: Size Comparisons

3.2.1 Evaluation of YOLOv7 and Faster R-CNN

Both models were trained on the H4FLC dataset. The data was downloaded in YOLOv7 format for YOLOv7 and COCO format Faster R-CNN. Both models were trained for 50 epochs. Both models showed relatively close performance for inference speed, with YOLOv7 training 2x faster than Faster R-CNN. YOLOv7 returned more false positives and was more sensitive to overlapping leaves compared to Faster R-CNN. For information on the accuracy of the models, see table 1. The metrics we used to measure performance are; precision P, recall R, mean Average precision mAP at 0.5 Intersection over Union (IoU) .5, and the average for the mean average precision using increments from $.5 \rightarrow .95$. Intersection over Union is a measure between a predicted bounding box and ground truth, calculating the area of overlap over the area of union.

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

Mean average precision measures if the IoU between the predicted bounding box and the ground truth is greater than or equal to a certain threshold.

4 Geometric Classification

Our use of CNNs has proven to be a highly effective approach for 4-leaf clover detection. In the following section, we explore alternative methodologies as part of a broader investigation. We propose two geometric techniques that offer interesting insights into clover feature detection and classification from a different perspective.

As section 3's CNNs performed slightly better at clover detection (bounding box localization) than classification, we focus our geometric techniques on classification to complement our methods. In the remaining sections, we start from images that bound clovers, as can be identified by the CNNs.

4.1 Laplacian of Gaussian Blob Detection

In our first geometric approach, we consider each leaf of a clover to be a blob that can be detected by a Laplacian of Gaussian (LoG) filter. We aim to detect each leaf blob to count the number of leaves and classify three- and four-leaf clovers.

4.1.1 The LoG filter

The LoG filter is defined as the second derivative (Laplacian) of a Gaussian function (Equation 1). When convolved with an image, it will detect regions with significant changes in intensity such as edges and circular blobs: light patches on a dark background or dark patches on a light background. The parameter σ controls the size of the filter. When convolved with an image, the filter produces a strong response in the center of a blob that has approximately the same size and shape as the filter (Figure 1). Therefore, with the correct filter size, a roughly circular leaf will produce a strong response. The formula for the LoG filter is as follows:

$$\nabla^2 G(x,y) = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) G(x,y) = \frac{1}{\sigma^4}\left(x^2 + y^2 - 2\sigma^2\right) \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{1}$$

where G(x,y) is the Gaussian function.

4.1.2 Automatic Scale Selection

To find the correct σ to detect leaves, we implement automatic scaling, similar to the approach described by [4]. We convolve the image of a clover with a range of σ values. For each pixel in the image, we retain the value σ that produces the most extreme response within a 3 by 3 window. Depending on the size of the filter, these extremes correspond to important features such as leaves, or to unwanted edges or background (Figure 2).

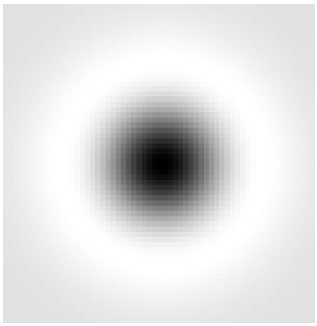
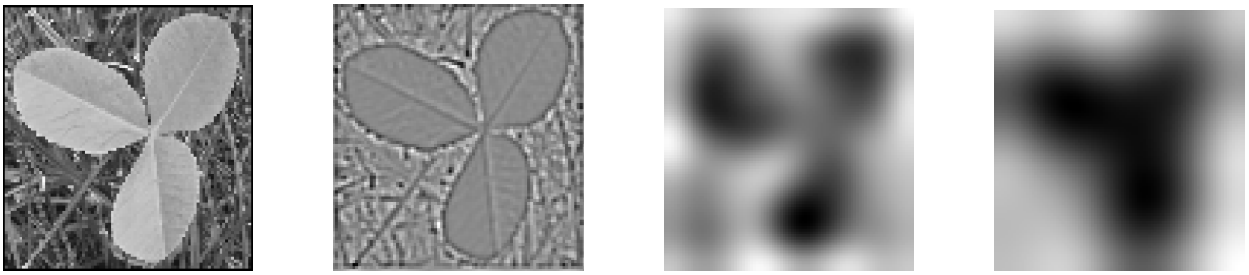


Figure 1: 62x62 Laplacian of Gaussian kernel with $\sigma = 10$. This kernel produces a maximal response for dark blobs on a light background and a minimal response for light blobs on a dark background.



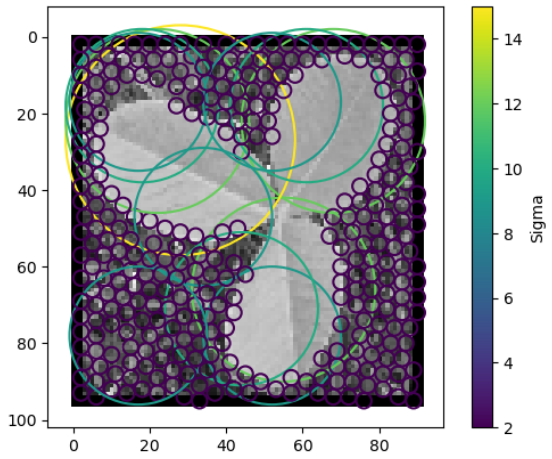
((a)) Original image ((b)) $\sigma = 1$ ((c)) $\sigma = 10$ ((d)) $\sigma = 18$

Figure 2: LoG response of a 78 by 100 clover image (2(a)) at multiple scales, where dark minima correspond to light blobs on dark backgrounds in the original image.

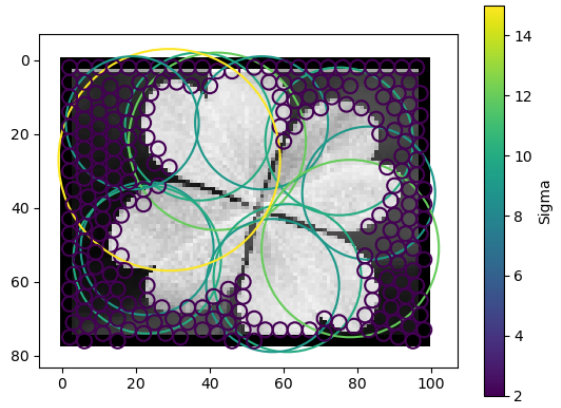
To eliminate blobs that do not correspond to the feature of interest, some previous works such as [4] select blobs by thresholding the response value, retaining only the strongest ones. However, this approach requires manual tuning and may eliminate leaf blobs that respond less strongly than the background in noisy data, particularly in cases where the clovers are not well contrasted with their green background. [2] also proposes a generalized LoG filter by including angled variations of the LoG kernel in scale selection to detect oriented blobs. While a kernel oriented in the direction of a leaf would increase the LoG response on leaves, testing all orientations at each scale would significantly increase the computational time.

Instead, we start by eliminating any blobs that significantly overlap with other blobs of the same σ value as they likely correspond to the same feature. We are left with two types of blobs: those detecting leaves and those detecting the edges of the leaves. The number of latter blobs is not consistent and is irrelevant to our classification task. To distinguish between the two, we incorporate a background σ value in our scale selection process. This background σ has a much smaller value than the expected leaf size, and its response dominates in the background areas outside the clover shape (Figure 3). For example, $\sigma_{\text{background}} = 1$ was used for clover bounding box of 78 by 100 pixels.

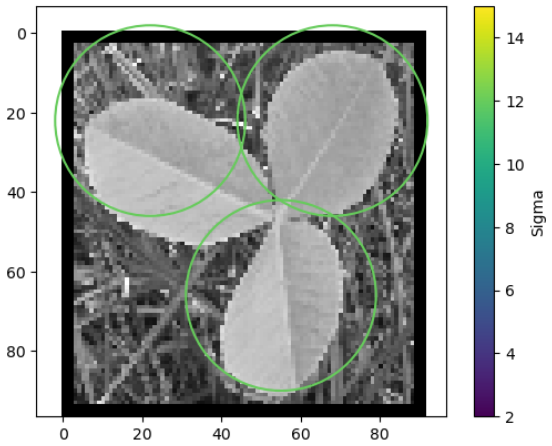
Assuming the leaves in a clover are approximately the same size, we can group the remaining blobs by their σ values. We then select the σ for which the blobs have the smallest proportion of overlap with the $\sigma_{\text{background}}$ blobs, distinguishing the most likely leaf blobs from the edge blobs. The number of blobs in this final group determines the predicted clover classification.



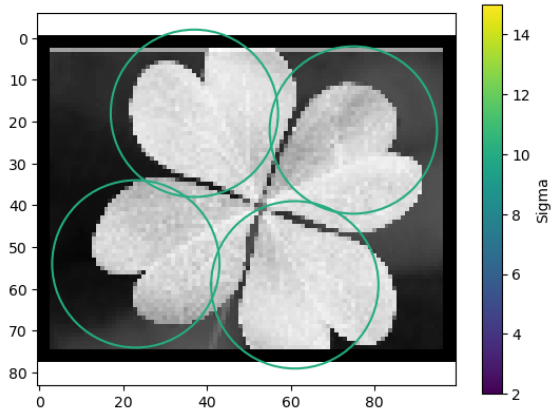
((a)) 3-leaf clover blobs



((b)) 4-leaf clover blobs



((c)) 3-leaf clover final leaf blobs



((d)) 4-leaf clover final leaf blobs

Figure 3: 3(a) and 3(b) show all blobs that produce the most extreme responses at their center positions and have less than 5% overlap with any other blobs of the same σ. 3(c) and 3(d) show the final blobs whose σ group has the least overlap with the $\sigma_{\text{background}}$, indicating that these blobs correspond most strongly to the clover leaves.

4.1.3 Evaluation of the LoG method

The effectiveness of this approach depends on the contrast in the image. In some cases, the leaves are very close together, and their boundaries may lack enough contrast to produce a strong response at the leaf scale. Furthermore, significant clover variations, such as those whose half-leaf lines are almost as strong as their leaf contours, may cause more blobs to be detected than there are leaves, yielding an incorrect classification.

To achieve good results, the choice of σ values to try as well as the overlap threshold for blobs with the same σ needs to be carefully tuned. This tuning depends on whether the image contains a single clover or a field of clovers in which case many more scales need to be tested. In our case, starting from a clover bounding box rescaled to a fixed size mitigates the need to retune for each image.

4.2 Flood Fill Segmentation

The LoG method relied on the ability to distinguish between leaves. To address cases where this distinction is unclear due to clover variation or image quality, we introduce a new flood-fill clover segmentation method that works on single clover blobs as opposed to individual leaf blobs. Preprocessing is done to obtain these blobs. Then, flood fill is implemented from the four furthest points and the resulting fill distributions are used to count leaves.

4.2.1 Data Processing

For the flood fill segmentation, our input needs to be a binary image with a single connected blob representing the clover. The blob does not need to have fine-details, it just needs to have the relative sizes and shapes of the leaves since flood fill classification is based on ratios.

The steps taken to preprocess the data are shown below. Many steps are included to deal with noisy images. These can be adapted for each image (see Appendix A for examples).

1. **Grayscale Conversion:** Image is converted to grayscale to move data into a single channel.
2. **Inversion Based on Brightness:** Image is inverted if the center is darker than the border, in preparation for binary thresholding.
3. **Edge Detection:** Edges are used to complement binary thresholding, to account for contrast changes (see Appendix A)
4. **Binary Thresholding:** Apply a binary threshold to isolate clover from background.
5. **Contour Filtering:** Contours are detected and small or straight lines are filtered out to focus on significant features.
6. **Connected Components:** Isolate connected components and only keep biggest one.
7. **Dilation:** Mask of the largest component is dilated to fill gaps and add volume to clover, which helps the flood fill segmentation.

4.2.2 Flood fill and farthest point algorithms

Given these processed images, the flood fill algorithm simultaneously propagates through the detected clover blob from multiple starting indices (see Figure 4). The algorithm uses a simple propagation but the key to proper segmentation is choosing the right starting points. For this we develop a simple algorithm to find the N farthest points for a given blob. We find the points with the maximum distance to the center of the blob, and maximum distance from other points we found (see Figure 4(a)).

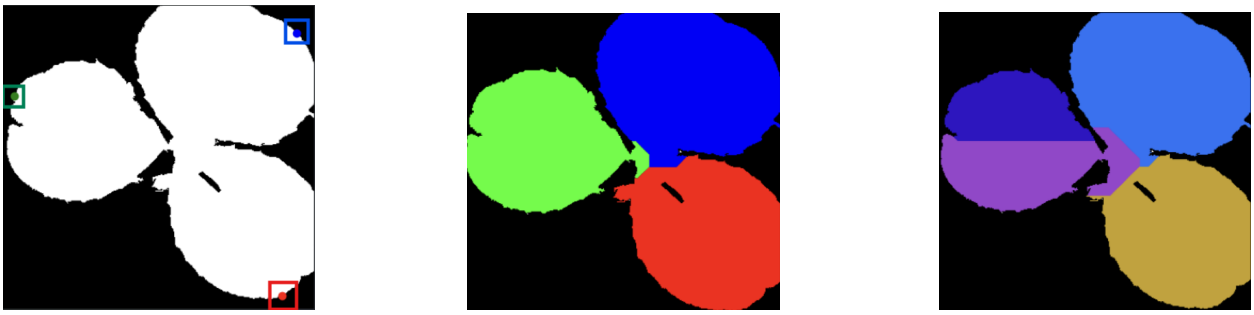
Ensure: Segmented image with blobs colored
function FLOODFILL(img, starting_points, max_iter)
  for each starting point **do**
    Initialize propagation queue
    Assign a unique color to queue
  end for

  for $i <$ *max_iter* **do**
    for each queue **do**
      Get all points in the queue
      Update with starting color
      Update queue with neighboring points
    end for
  end for
return Segmented image
end function

Ensure: Farthest Points on Blob
function FINDNFARTHESTPOINTS(binary_img, n)
  Find the center of the blob
  $farthest\_points \leftarrow []$
  for $i < n$ **do**
    Iterate through every pixel in blob
    Find *distance* from center
    $distances \leftarrow [\sqrt{(x - center\_x)^2 + (y - center\_y)^2}]$

    Find *distance* from other farthest points
    $distances \leftarrow [\sqrt{(x - points.x)^2 + (y - points.y)^2})]$

    Maximize $\sum distances$
  end for
return $farthest\_points$
end function



((a)) Farthest Points ((b)) Segmentation (n=3) ((c)) Segmentation (n=4)

Figure 4: Stages of flood fill segmentation (n=number of farthest points)

4.2.3 Classification

With our algorithms defined, now we have to define the number of farthest points n we want to sample from each blob. To get a clean segmentation of the leaves of a 3-leaf clover, we can assume that would just be the number of

leaves $n = 3$. The same goes for a 4-leaf clover with $n = 4$. Our classification method is to test whether using n points yields a clean segmentation. So a 4-leaf clover with $n = 3$ will theoretically return an uneven split between the 3 segments. And the same applies to a 3-leaf clover with $n = 4$, returning uneven segmentation. In practice, we have found that using $n = 4$ and testing if a given leaf is a 4-leaf clover provides better results than testing for a 3-leaf.

For the classification, we calculate the ratio of each segment relative to the size of the blob. If the difference between ratios is above a threshold, then the segmentation is deemed uneven and the classification is made (see Table 3).

4-Leaf	27%	23%	24%	26%
3-Leaf	40%	13%	16%	31%

Table 3: Sample Segmentation with $n = 4$

4.2.4 Evaluation of the flood-fill segmentation method

The flood fill algorithm provides relatively consistent results. However, they depend heavily on preprocessing returning a clean binary image. The automatic preprocessing works for clean inputs, but with noisy or blurry images, some manual tweaking of parameters is required to get a coherent output.

5 Conclusion and Future Work

The variety of methods we proposed complement each other in terms of addressing the different aspects of our problem definition as defined in section 2.

The YOLOv7 and Faster R-CNN models have proven to be highly effective for both detection and classification tasks in noisy backgrounds. The advantage is they can easily be improved by being enriched with data. They are, as intended, invariant to scale and orientation as well as being quite robust to clover variations. With a little extra training and combining some data from other datasets, clover hunters will very soon be able to classify 4-leaf clovers in real time. If they however insist on a using a geometric approach, it will take some more work before either the LoG Blob Detection or Flood Fill Segmentation can run in real-time.

While the LoG method requires further tuning to account for the wide variety of clover types, it is a useful geometric approach in that it can easily be extended from clover bounding boxes to larger clover patch scales. Finally, the data processing and flood-fill segmentation technique offers a more robust solution to variations in clover types.

References

[1] Adam Fonagy. Hunting for four-leaf clovers dataset. `https://universe.roboflow.com/adam-fonagy/hunting-for-four-leaf-clovers`, nov 2023. visited on 2024-12-15.

[2] Hui Kong, Hatice Cinar Akakin, and Sanjay Sarma. A generalized laplacian of gaussian filter for blob detection and its applications. *IEEE Transactions on Cybernetics*, 43(6):1719–1733, 2013.

[3] Gustavo Perez Pablo Arbelaez Laura Bravo, Alejandro Pardo. Finding four-leaf clovers: A benchmark for fine-grained object localization. *FGVC6*, 2019.

[4] T. Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):79–116, 1998.

[5] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

[6] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, 2022.