Where's Waldo?

A Report of Motivations, Methodologies, Results, and Future Work of Detecting Waldo Using Computer Vision

by Nathan Pham and Youssef Beltagy

Table of Contents:

I. Introduction

The objective of this project was to apply what we learned in CSS 487 in a practical problem. We decided to make a program that solves the Where's Waldo puzzle. Where's Waldo is a puzzle in which you look for a cartoon character, Waldo, in a large image, a Waldo World. Look for the black ellipse to find waldo.

Figure 2 Waldo World Example

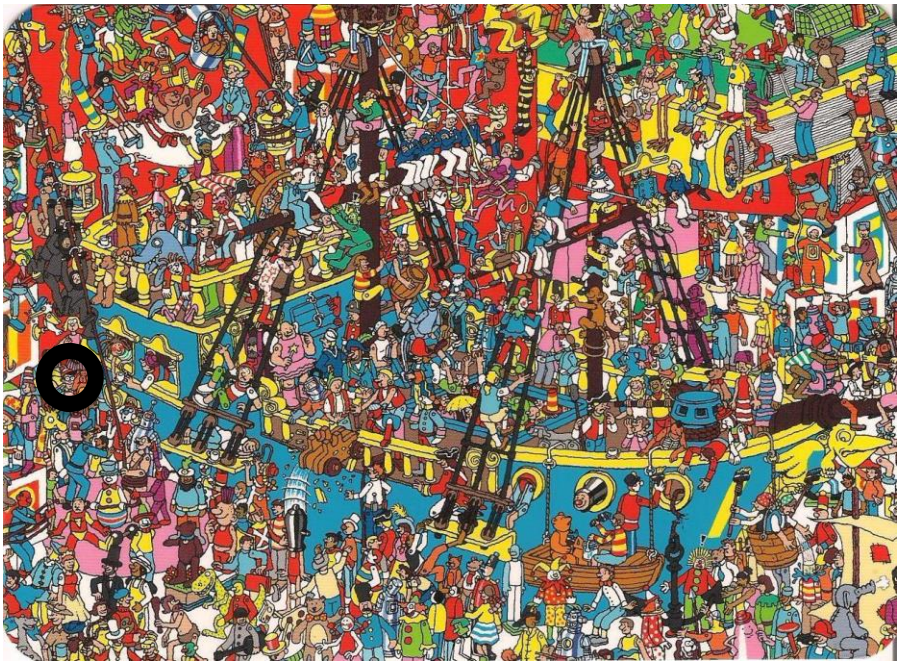


Figure 1 Waldo

In the context of Computer Vision, Where's Waldo is an object detection in an extremely noisy environment problem. Waldo doesn't only appear in random places, he is often distorted as well. Thus, template matching is unlikely to work (especially since each Waldo World has dozens of faces that can confuse template matching). We thought that Where's Waldo is a challenging problem that will allow us to learn even if we fail at delivering a working final product.

To be more specific, our program should take an input image from the user. We assume that the input image will be a Waldo World, and will have Waldo. Our output would be an image with one or more possible locations of Waldo. False positives are acceptable in our success criteria because it can be difficult to display the exact location of Waldo, since Waldo maps can have imposter Waldos, or things of a similar color palette to Waldo to confuse the puzzle-challengers.

Through many attempts of a variety of algorithms, we have narrowed down to two plausible algorithms that can be used for further processing to pinpoint the exact location as Waldo. These were the Hough Circles and SIFT algorithm, and the Haar Cascade algorithm. After we implemented our second iteration of these algorithms, we became curious to see if other

programmers attempted to find Waldo as well. Although these attempts were similar to ours, they used higher level languages or tools such as MATLAB and Google's AutoML, and we developed our algorithms independently from them. One instance of these attempts is shown below. Matlab probably has better implementation for hough circles, because it could detect Waldo's glasses, while opencv couldn't.

https://www.youtube.com/watch?v=-i7HMPpxB-Y

https://ohaddan.weebly.com/blog/let-some-computer-vision-algorithms-find-waldo

In addition, we also found a dataset of Where's Waldo images, Last week, and will be using this dataset in this report.

https://github.com/vc1492a/Hey-Waldo

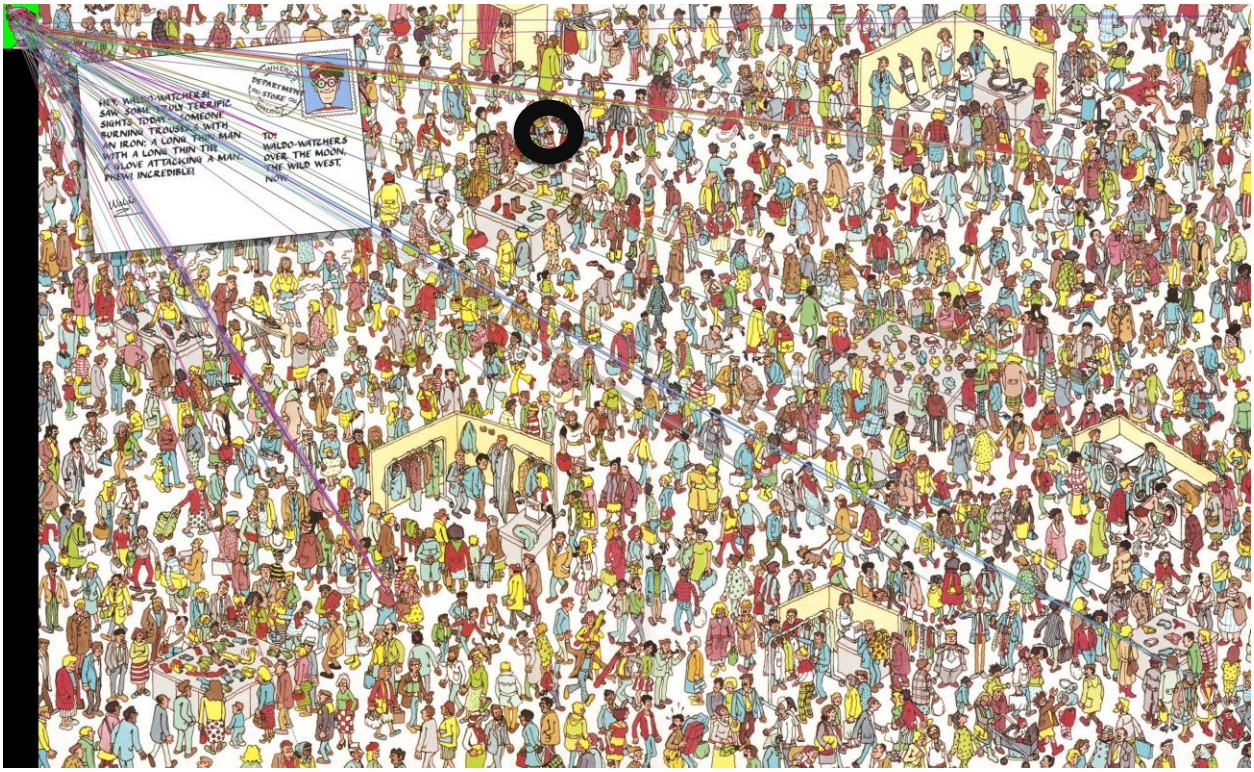
This report is a record of the methods we attempted in the chronological order that we attempted them. We will also write the methods that we considered but decided against using, and we will explain our reasoning for why these methods were not worthwhile.

II. Methodologies and Results

A. SIFT

Our first approach was using SIFT on both an exemplar Waldo image and Waldo World. We were trying to get a clearer idea of both the problem and the versatility of SIFT. We were just experimenting in the beginning.

Figure 3 SIFT OUTPUT



This attempt did not work for a few reasons. The exemplar image could have more than a hundred features, but Waldo in the Waldo World is usually small and surrounded by noise. It was hard to find features for Waldo in The Waldo World. We often found just a few features for Waldo inside the Waldo World (if not just one feature). But even if we found features for Waldo inside waldo world, the features will be distorted and affected by noise, so they weren't perfect matches with the exemplar features.

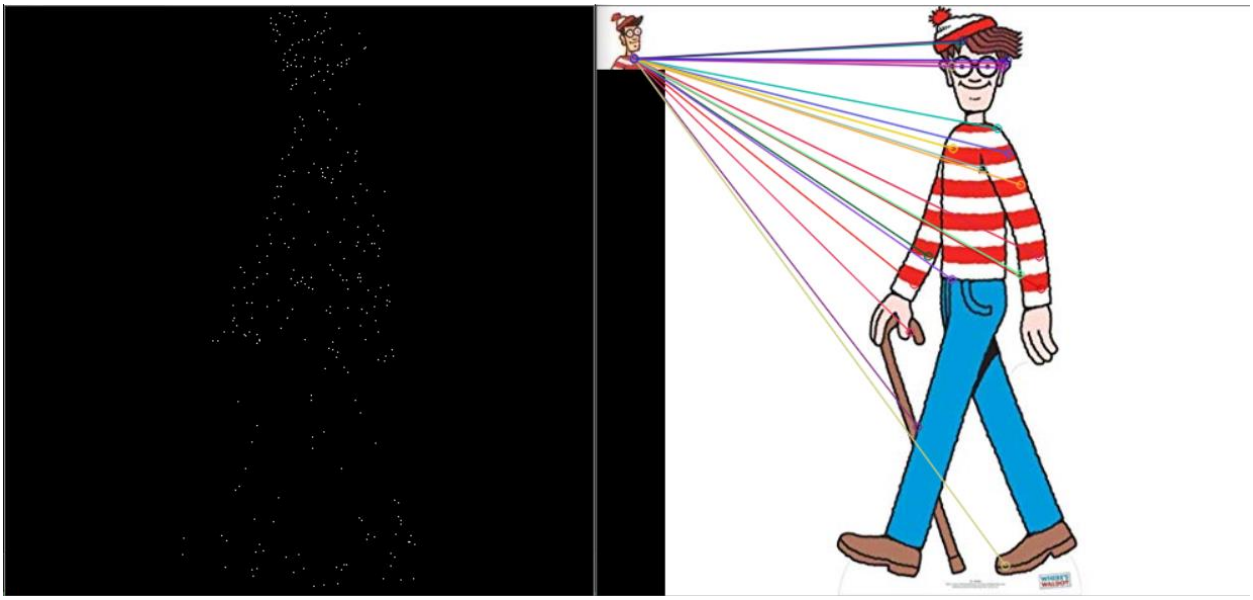
Mainly because of these two reasons, almost all the features in the exemplar image are mapped to random features in Waldo World. Waldo World has so many features, that noise in the image had a better match to the exemplar than Waldo himself.

It is worthy to note that OpenCV SIFT uses gray-scale images and does not discern color. Colors are extremely important to Waldo's identity, and led us to consider ways that would incorporate color information.

B. SIFT Each Color Channel + Clustering

To consider color information with SIFT (a cheap attempt to imitate your SIFT with colors), we tried a SIFT for all color channels, including Gray. We made four images for the Waldo World: one image in which all the colors were eliminated except for red, one in which all the colors were eliminated except for Blue, one in which all the colors were eliminated except for green, and a normal grayscale image. We did the same for the exemplar.

Figure 4 SIFT With colors and clustering output



We tried to match the features in each pair of exemplar and Waldo World in each color. We were using a one-to-many function (knnMatch) that matches every feature in the exemplar to multiple ones in Waldo World. The idea was that we were using SIFT with four slightly different image pairs. We were trying to find waldo in all the four cases. Our hope was that Waldo in Waldo World would get more matches than noise in total. We would then map the match locations to a binary image and use a clustering algorithm (that focuses on distance between points) to find Waldo.

This did not work because of the same reasons as the normal SIFT. We thought we could overcome the noise by numbers (KnnMatch), but the idea wasn’t successful. In images where waldo is easy to see and looks similar to the exemplar, that approach worked. However, when we were trying to solve an actual puzzle, we rarely got matches with waldo. The noise was more powerful than we thought, so the clustering would get a wrong location.

C. Stripes

Since waldo wears an iconic red and white striped shirt, we thought of looking for him by using a pyramid of images and a red and white stripes filter. However, we did not pursue this approach because waldo is often occluded, so his shirt is not visible. The algorithm is guaranteed to fail in these cases and will be weak to other stripe patterns in the image. Waldo Worlds have a lot of stripes to misdirect solvers.

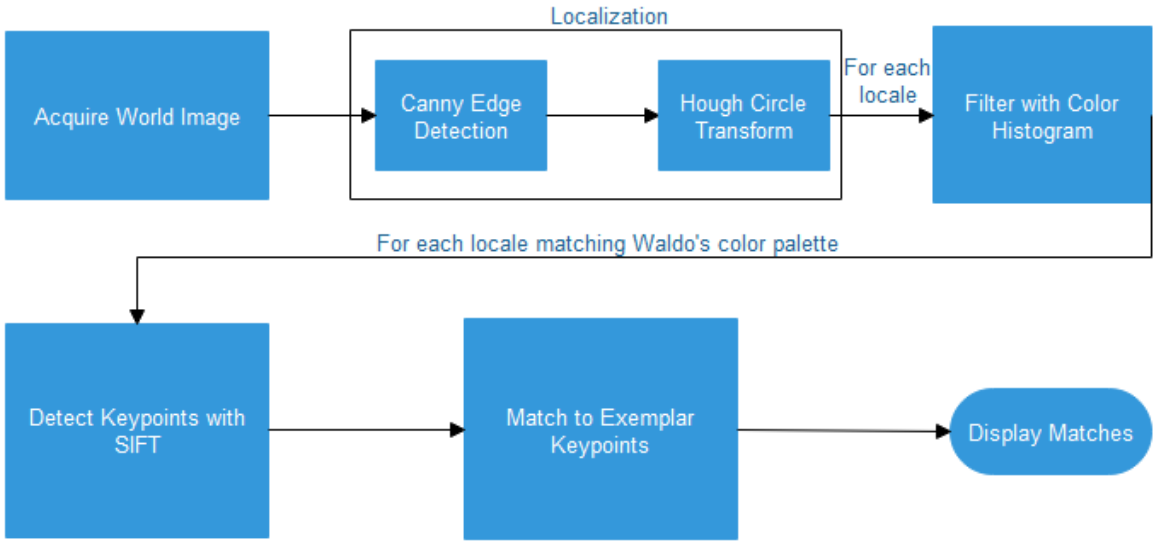
Figure 5 Waldo occluded body

We think that the stripes idea can be used to filter possible locations of waldo, but it is not versatile enough to detect waldo. It might even fail as a filter and result in false negatives if waldo is drawn with only his head, or in the case of "New Waldos" where he has hiking/camping gear that occludes his body. It would be useful only in the case in which we know waldo is drawn with his body, with stripes exposed.

D. Hough Circles and SIFT

In this method we attempted to implement all that we learned about the importance of color as an identifier and the use of localization to reduce search areas. The Hough Circles and SIFT method follows the general pipeline shown in Figure 6.

Figure 6 Hough Circle and SIFT Pipeline



Where each locale is represented as a circle detected in the Hough Circle Transform. The ideal goal of this localization is to find Waldo with his glasses. His glasses are semi-circular and no imposters in the Waldo books have circular glasses, removing the situation of imposters entirely. A Canny Edge Detection threshold of 750 in the x-direction and 750 in the y-direction is used to

find his glasses, since they are always in a situation of high contrast. Empirically, we found that using a canny edge detector before hough circles improves the output. The parameters used in the Hough Circle Transform are shown below:

```
HoughCircles(CannyImage, outputVector, HOUGH_GRADIENT, 1, 4, 800, 15, 0, 20);
```

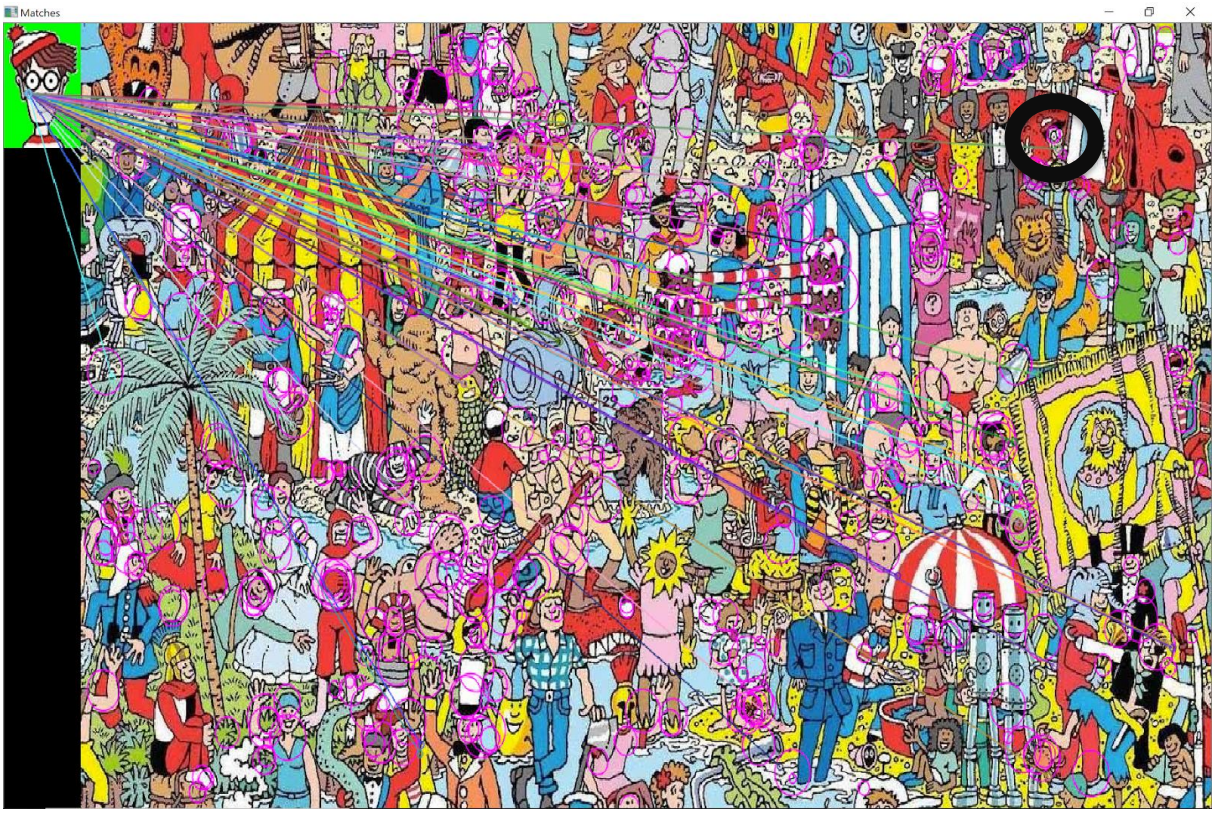
The method says for each pixel whose resolution is the same as the Canny image, use the Hough gradient to store the best circle into the output vector whose radius is from 0 to 20 pixels, that is 4 pixels apart from any other circle, and has a lower gradient threshold of 100. The 15 stands for the accumulation threshold for an object to be considered as a circle during the Hough Circle voting. Through tweaking with the parameters for specific maps, 15 was reasonable. 10 detected too many false circles, and 20 detected no circles at all. The 20 pixel max radius is to illustrate that Waldo is very tiny in the world. This results in the circle being Waldo's face as a whole, or includes the top of his hat. Because we got waldo's face as a circle, we thought that we could use Hough Circles as a method to localize Waldo's face.

For each circle detected, we make a color histogram comparison to filter circles that do not match Waldo's color palette. A histogram bin of 36x10x10 in the HSV scale was used. The color histogram comparisons between the example image and the circle is compared with the Bhattacharyya Distribution since it normalizes both histograms for comparisons. If the comparison is at least 15% similar, the circle is kept. Otherwise, it is removed. The main reason for this choice is to take into account that the eyes or tip of Waldo's hat does not contain his skin color, hair color, etc, in addition to picking up artifacts from the square that inscribes the circle. After performing SIFT on each circle, the matching algorithm uses a FLANN based KNN matcher, finding the top 50 matches for each feature (possible locations for Waldo), and each of these matches are displayed onto the screen.

With this algorithm, Waldo can be detected from 6 out of the 10 test maps. Although the chances of single point mapping exactly to a world Waldo point is extremely low, the cumulative probability of mapping Waldo with these matches was a 60% success. The exemplar is searched for as many keypoints as possible, including his glasses, because we will be unsure which one keypoint will be extracted from the world Waldo. Areas such as World 5 (The Castle Siege level), where immense noise such as the other soldiers is around, it is still possible to find him by reducing the Canny Edge detector to 450 in both x and y directions. Waldo is very grainy, but

can still be detected. However, run-time worsens to six minutes due to the massive amount of circles to filter out with the color histogram (around 12000).

Figure 7 Success Mapping Case 1: World 0

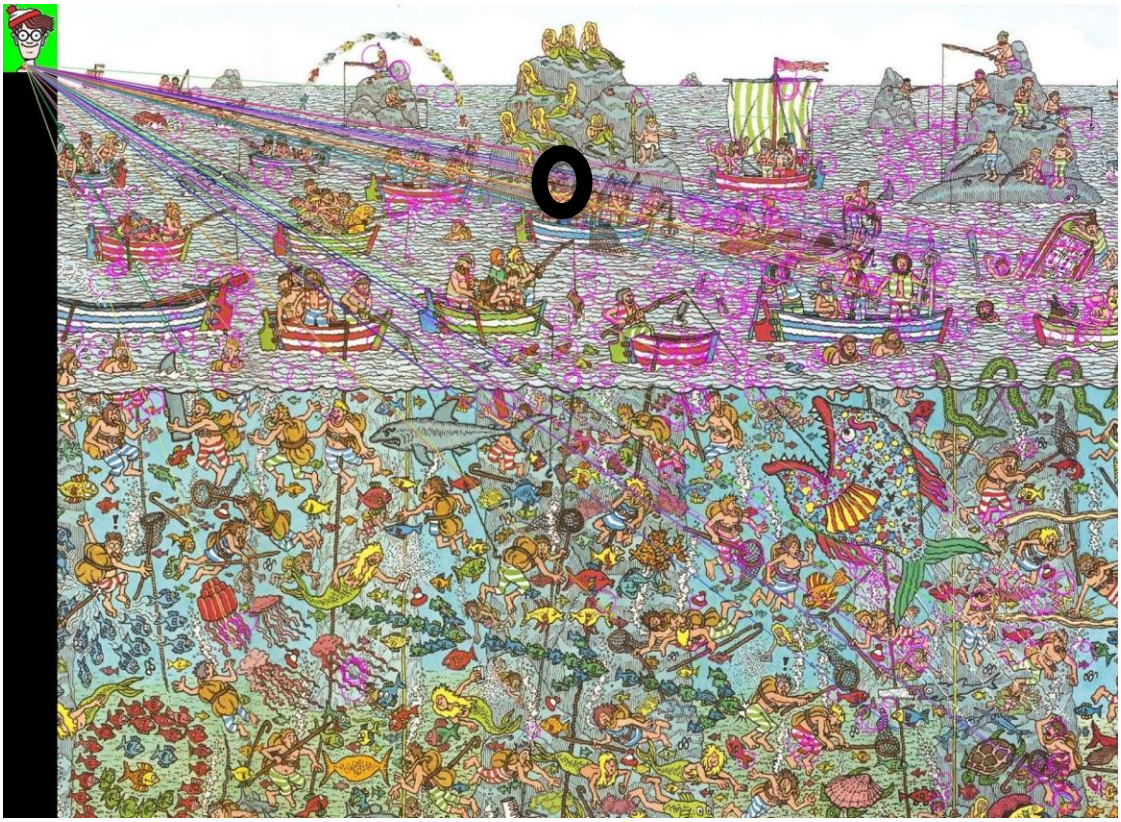


The failures of the other four can be described with one example. An example of a fail case occurs in World 6. Here, the Canny edge detector set at the default 750 removes Waldo entirely. However, when we reduced the Canny edge detector threshold to 450 in both directions, we managed to get one match to Waldo in World 5. But, the histogram color filtering removed this circle since his skin color is more peachy-tan, than a shade of white to light-tan skin.

Figure 8 Canny edge detector of failure case

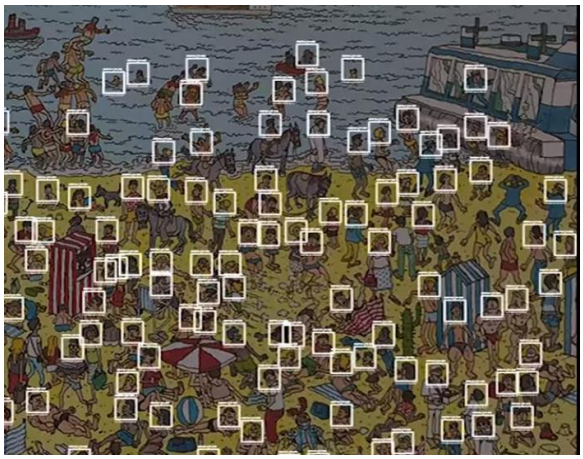
Figure 9 Hough circles of failure case

Figure 10 fail case of Hough Circle method



E. Haar Cascade

Figure 11 Ideal goal of Haar Cascade



Since the Hough circles often missed Waldo, had too many false positive, needed us to tweak the parameters, and when there were matches, the matches didn't contain a good enough portion of waldo to be used later in the pipeline. We decided to entertain a different approach. The only thing that waldo is guaranteed to be drawn with is his face (and beanie), we thought of using Haar Cascade to detect faces in the Waldo World. Then if we found all faces in the Waldo World, we could use a face recognition algorithm to identify waldo. Basically, the idea at this point was to change the problem into a face detection and recognition problem.

Our first attempt with Cascade classifiers was using OpenCV's pretrained face classifiers. OpenCV had multiple classifiers for faces, but none of them detected anything in the Waldo World images. Because the face models for Haar cascade didn't find anything in the image, we expected that face recognition algorithms would not work in the pipeline either.

We decided to train our own Cartoon Character classifier. Ideally, we wanted an output that looks like figure 11. We would then use color histograms and sift to narrow down Waldo's location. Training the classifier took a lot of time, we had to retrain the classifier multiple times, and we started this process in this last week. Thus, we were unable to finish this project, nor this attempt. However, we believe it is the most promising one yet because it has better output, and is proven by others.

Haar cascade is trained by providing a training command with thousands of positive and negative images. It is a cascade of weak classifier. It finds features in Waldo looks for those features in Waldo World in steps. The idea is that if the cascade has 20 stages, and a part of an image doesn't succeed stage 2, it will stop being considered as a possible location

To train a Classifier, you have to provide it with thousands of both positive and negative examples. The process is slow and takes hours to run (if not days). My computer froze multiple times and wasted my efforts, so I learned how to buy and use a remote Linux server. Most of the positive images are trained by distorting one positive image and overlaying it over a negative image. However, because the command that distorts and overlays positive images can't take more than one positive image, we were not able to train the classifier with a variety of different Waldo exemplars that I prepared. This is one of the things we want to improve later.

Figure 12 Waldo Exemplar, Negative Image, Positive Image made by overlyaing an exemplar over a negative image



Haar cascade often detected waldo. We cannot give definitive numbers because the program is incomplete, and we couldn't test it thoroughly. But in at least 80% of our sample images, the classifier detected waldo. The problem was that it also detected a lot of false positive (in couple of hundreds and not thousands like the Hough circles), and often the detected square is eight times as large as the waldo inside.

Figure 13 Haar Cascade Successful output

Our solution for the false positives is a pipeline of processes that each eliminate what it deems not a match. We focused on methods that would avoid false negatives (eliminating a rectangle that has Waldo). We implemented and tested a sift process that eliminates at least half the false positive. We changed our approach from the Hough Circles, so that we make sure we don't lose the localization of each rectangle. Notice that there are significantly less circles in Figure 14 compared with 13.

Figure 14 Haar Cascade + Sift successful output

We tried using color histograms with a threshold for the response. Because the Waldo submatrices can have a large background, the histogram function didn't work well. We implemented another histogram elimination process that would eliminate around half the

submatrices by comparing the responses, but we didn’t have enough time to test it or experiment with it.

Haar Cascade seems promising and has a lot of room for improvement. We can solve the problem in which I could only make positive examples from one exemplar and retrain the model. We can use Haar cascade inside each rectangle, so that if waldo is small, we get a rectangle around him. We can then use our new Histogram function.

We also want to use a Histogram of Oriented gradients and template matching (with a pyramid of images) for every rectangle that appears in the end. However, that will depend on whether the other processes mentioned above were not enough to pinpoint Waldo.

III. Lessons Learned, Future Work, and Improvements

One of the lessons learned from this project is that pipelining is a great way to narrow down the points of interest of an image. We experimented with a variety of different openCV algorithms. We also Learned how to download and compile a library from source code, how to make a server and train a cascade model, and how to develop a system to pipeline processes. While we were not successful in making a program that finds Waldo just with openCV in the allocated time, we are confident that we can still solve this problem with more time.

Future work for the Waldo Finder would include improving the Haar cascade Classifier (retraining it) and implementing more methods to narrow down the exact location of Waldo given a vector of possible locations. We would also eliminate warnings and check the code for any bugs. We couldn't clean the code to the best possible because we were trying different techniques to the last day.

For pinpointing Waldo, here is a list of methods that we have in mind for next steps:

1. We can use Haar Cascade through every Rectangle to tight fit a rectangle for waldo.
2. We can then use the new histogramElimination function to eliminate the worst half matches.
3. After that we can use template Matching or Histogram of oriented gradients to look for rectangles that have a similar structure to waldo.
4. Eliminate warnings and test the program.

IV. Appendix

Hough Circle Success Cases:

Figure 15 Success Mapping Case 2: World 3

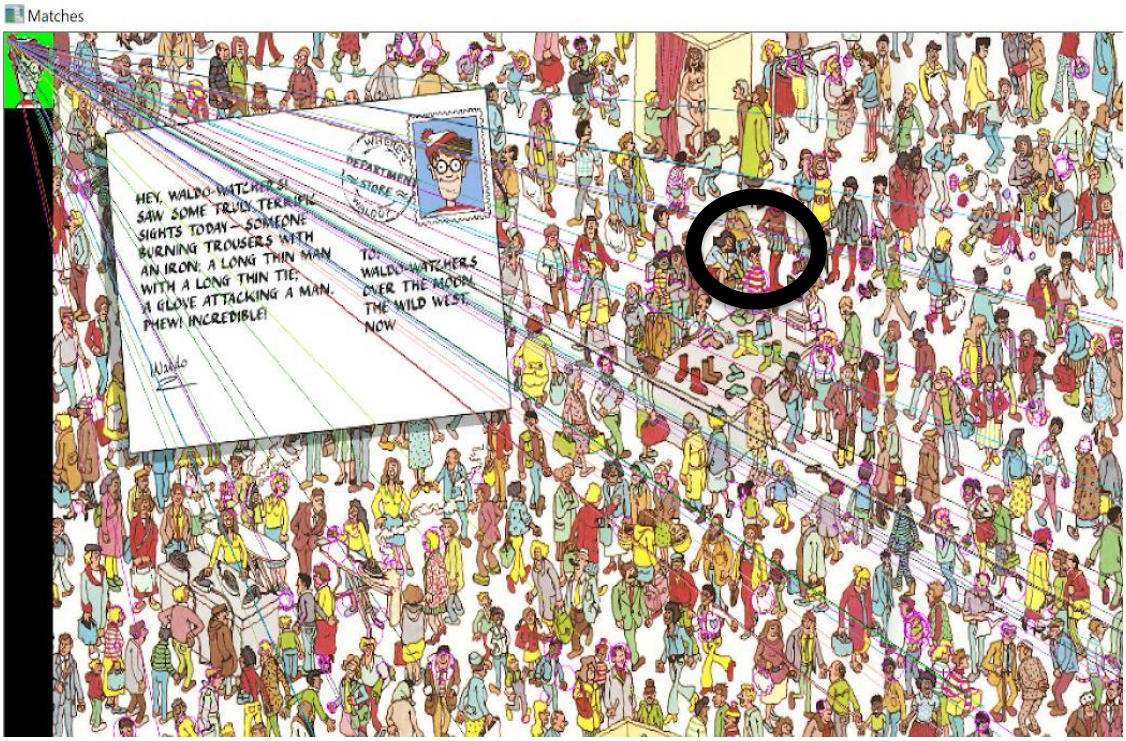


Figure 16 Success Mapping Case 3: World 4

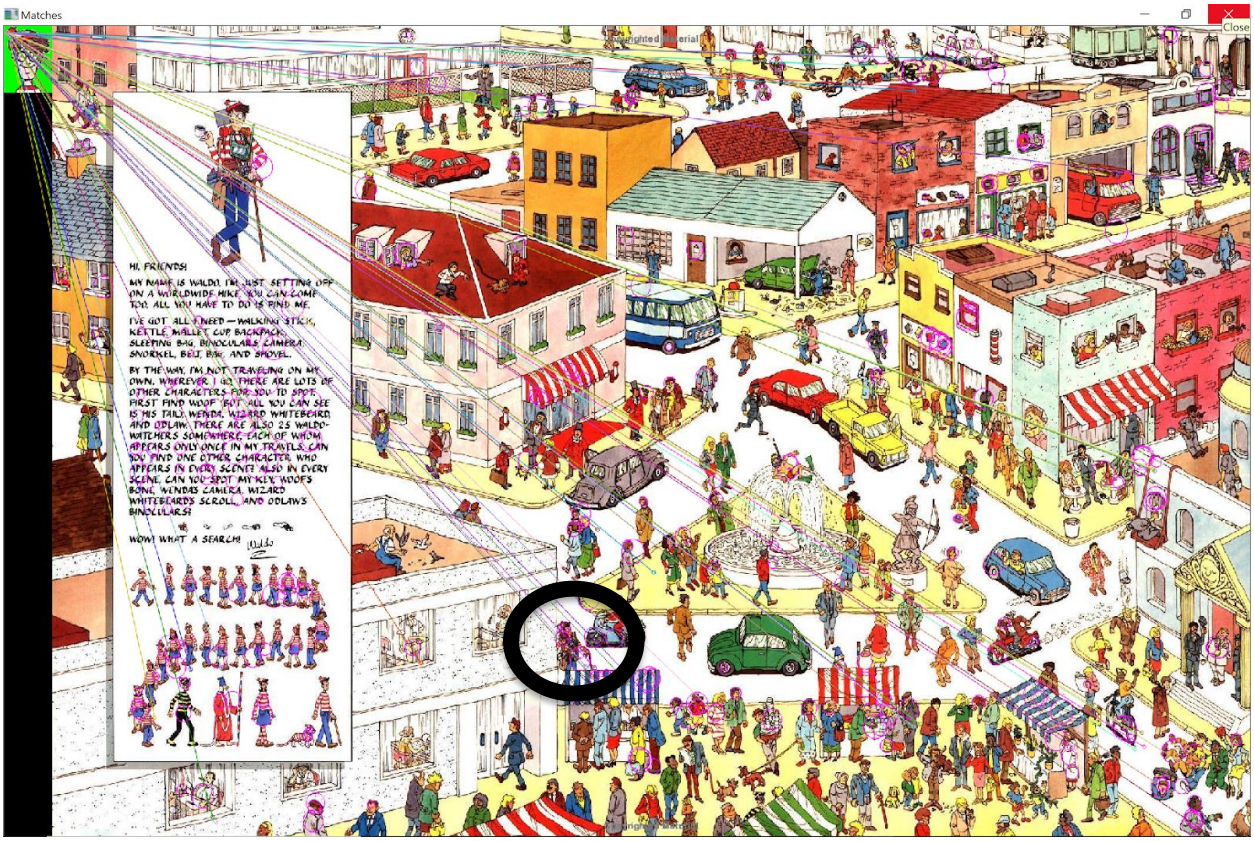


Figure 17 Success Mapping Case 4: World 5

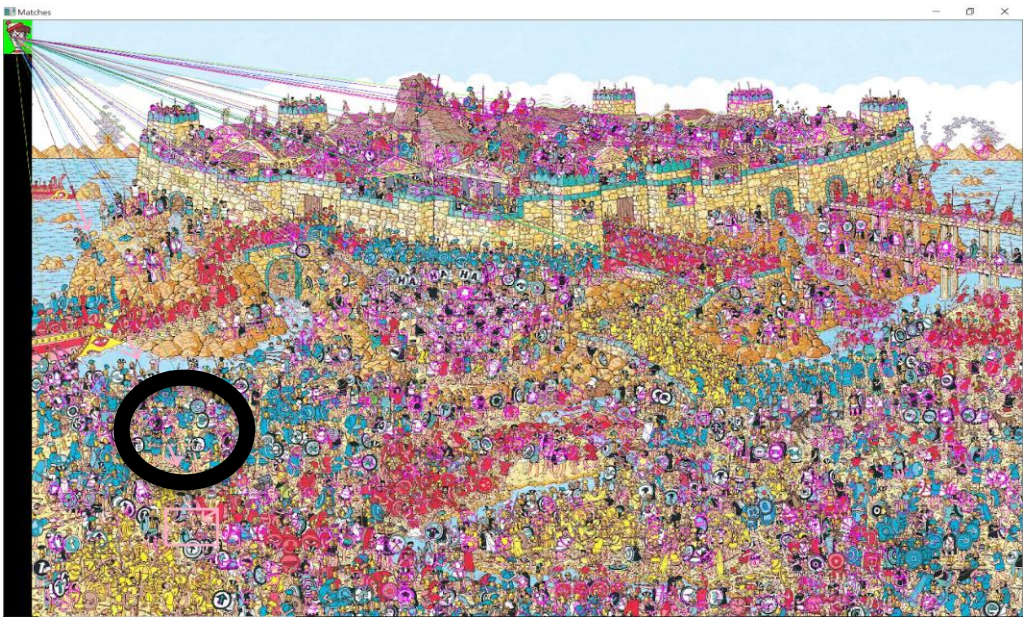
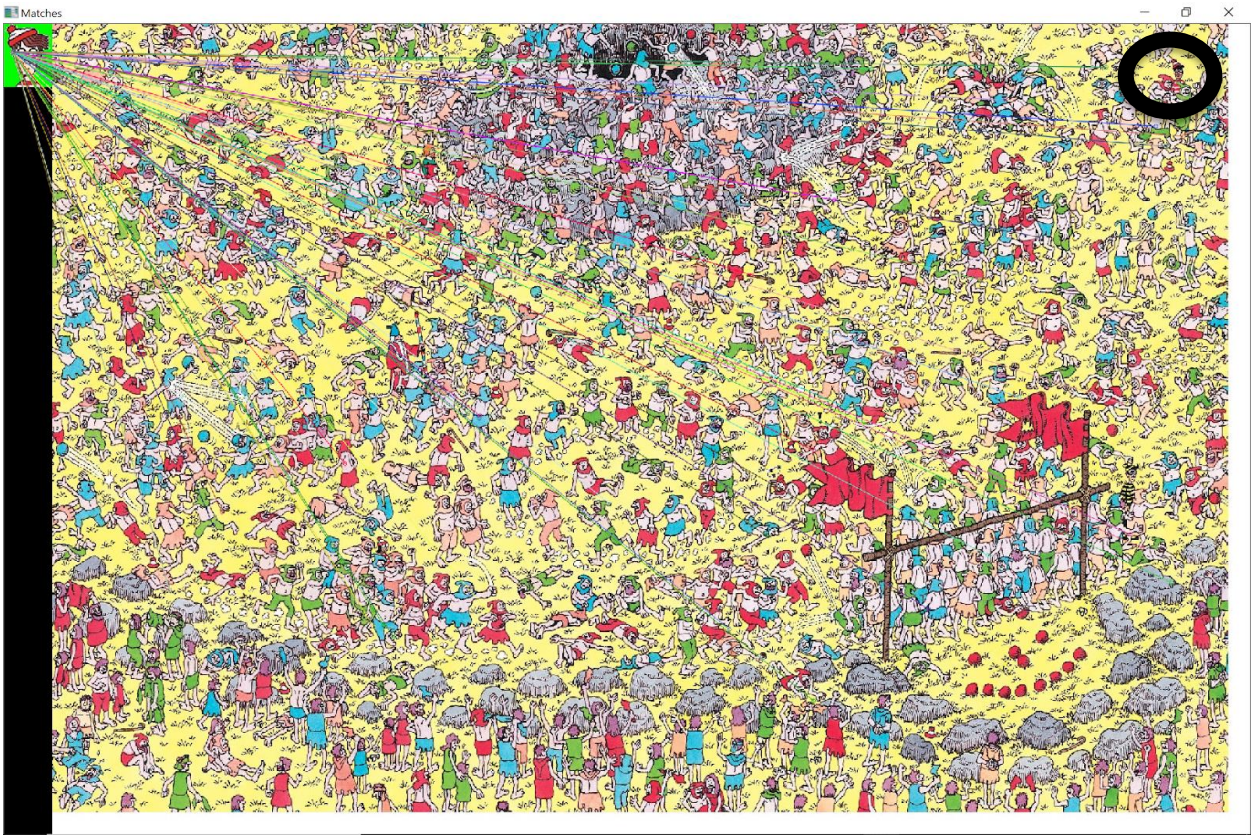


Figure 18 Success Mapping Case 5: World 7

For better clarity, Waldo's face is zoomed in

Figure 19 Fail Case 6: World 8



The Images for Haar cascade are in a folder.