# Where's Waldo?

**Introduction.**
After 5 intense lectures of computer vision (with more to come!) you must be eager to try something interesting with the skills you learned from the class!

"Where's Waldo" is a series of Wimmelbilder books featuring Waldo [1], along with his friends who set off on "a-world-wide hike". The books feature a series of detailed illustrations (see Fig.1 for an example) and your job is to find Waldo and his friend Wenda and Wizard (see Fig. 2 to get to know them!).

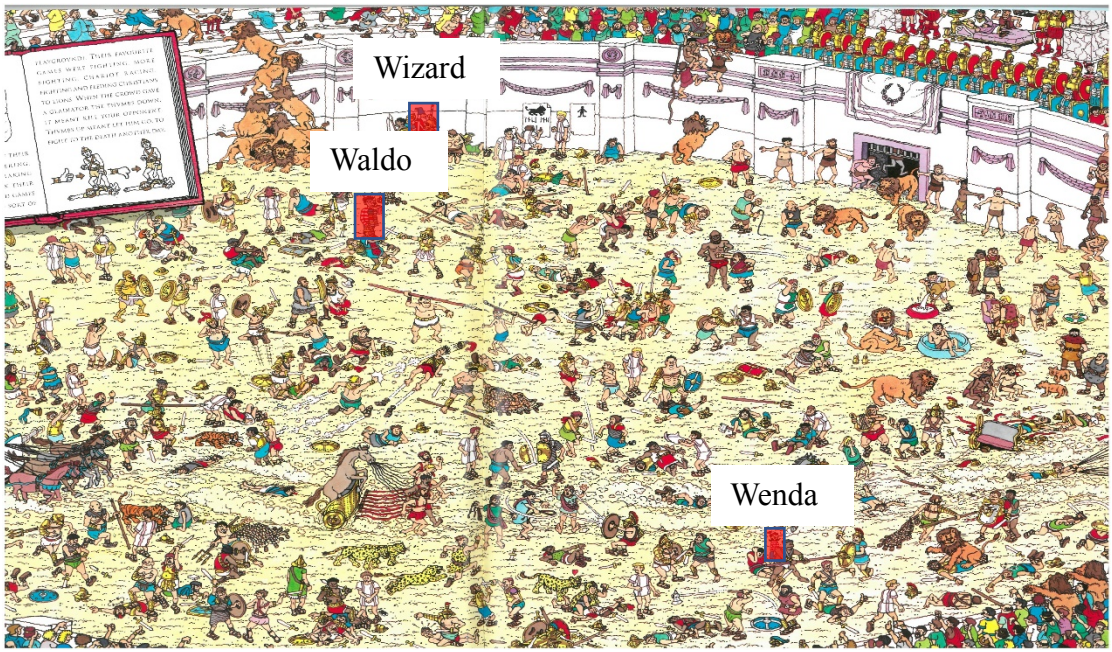



Figure 1. Typical scene in the book "Where is Waldo"

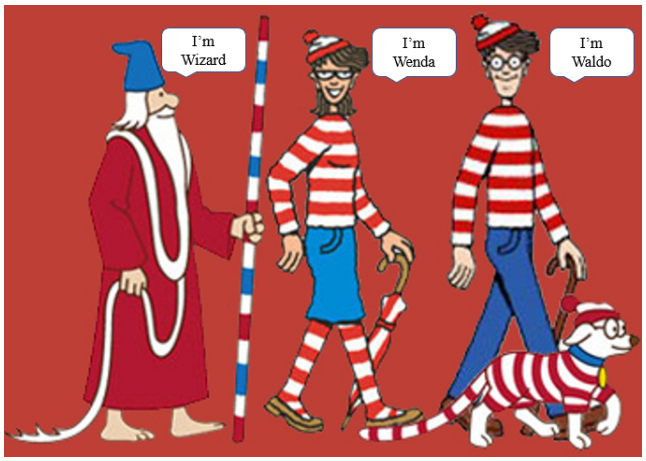



Figure 2. Wizard, Wenda and Waldo.

[1] Waldo is international and has lots of names depending on where the books are published! Originally, his name was Wally (UK), but he also goes by

We have scanned the images from the books and your task is to detect three of the characters (Waldo, Wenda and Wizard) in each scene using any non-deep computer vision algorithm of your choosing.

To facilitate your job, we provide you some annotated samples, which we painstakingly hand labelled. However, the annotations are not so perfect, and certain characters are hard to find sometimes. So, you are welcome to improve the annotations yourself, or search for online resources to help you. The format of the annotations follows the Pascal VOC standard, and the location of the object is given by (xmin, ymin, xmax, ymax) which denotes the top left and bottom right of the bounding box. Please refer to the annotation file for the specifics.

**Evaluation Metric.**
We adopt the mean average precision (mAP) as metric to evaluate your algorithm's performance. We regard the 3 characters as 3 classes and for each class, we calculate the average precision based on the ground-truth and detected characters. You can read more about mAP here:

https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173

We have provided the evaluation script. Note that your algorithm's efficiency and novelty will also be counted into the final grade flexibly.

**Requirement.**
You are free to propose and implement your own algorithms to address the problem. We have a separate hold-out test set to compare your algorithm's performance with everyone else in the class. Please submit your code and format your outputs as suggested in 'REAME.md' and also submit a detailed readme file to ensure that the TAs can run your code successfully.

**Don't know where to start?**

(1) Brute-force it! What about a normalized cross-correlation approach as described in Lecture 2. Hint: this method is will be very slow and unlikely to return Waldo (or Wenda or Wizard) as the top hit.
(2) Interest Point Matching. What about detecting interest points and finding matches in the images? See Lecture 5 / 6, similar to Lab 3.
(3) Detectors?
(4) Classifiers?

# CS4243 Project: Where's Waldo?

Cai Zhuo, Wei Jianxin, Vernon Cher Chu Xiong

November 12, 2019

Our group explored a wide range of methods to find waldos, including pixel-to-pixel correlation matching methods and feature-based classifier methods. All of these methods are either bad in terms of performance or computationally expensive. Finally, we selected a set of templates to directly match them the images and apply an adaboost-based classifier to the proposals generated in the first step. This method achieves both good speed and performance.

## 1 Introduction

The problem of finding waldos and other characters in an image is interesting. It is different from real world object detection problems or face recognition problems.

For object detection problems, the goal is to detect all objects of a particular class, where intra-class variance can be drastic and difficult to describe. In Where's Waldo, we are only interested in three individual characters.

For face recognition problems, the task is usually divided to two phases, face detection and face recognition. In the detection phase, sliding windows approach followed by window classifiers is widely used. Since the number of candidate windows can be very huge (about $10^5 \sim 10^7$ in waldo's detection), fast and accurate detection classifier is needed. Viola-Jones face detection algorithm[1] use harr features, which are fast to compute, and boosting classifier, which is very fast during prediction. During the project, we were partly inspired by Viola-Jones algorithm. In the recognition phase, the difficulty is to find the trivial difference between faces of different people, which naturally requires methods of increasing complexity, while speed is not a major factor of concern. Waldo's face and clothes do not vary much, thus it is possible to build a computationally efficient recognition classifier for waldo. However, this does not mean waldo can be easily detected.

In waldo's problem, the image size is usually very large comparing to the size of waldo's face or body. Besides, the relative size of waldo varies significantly. Therefore, the number of candidate windows are much larger than that in usual object detection problems. Further more, the number of characters that resemble waldo in shape is relatively large. In some images, waldo's face is so small that we almost can't recognize it only through his face.

Object detection has been an interest of research for many years. To accelerate detection, different region proposal methods are proposed in place of sliding-windows convention. Instead of sliding windows densely, region proposal methods yield a relatively small number of candidate windows ($\sim 1000$) that probably contain objects so that more complex and expensive classifiers can be applied. Some of the methods are based on the structure of edges[2], some based on saliency against background[3], some based on closed boundary property[3]. In most of the images in waldo's problem, the edges are densely distributed and objects of similar shape are also densely distributed. Some adjacent objects occlude each other. All of these factors make region proposal methods less attractive for waldo's problem.

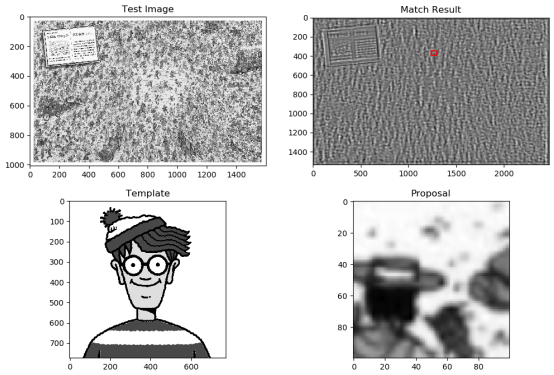
In Viola-Jones face detection algorithm, extremely low false positive rate is required to ensure the detected windows are mostly people's faces. If we follow the detection and recognition convention, the false positive rate is not required to be as low as it is in pure face detection problem, since our goal is to detect a specific character and we can use more complex classifiers to remove the false positives in later phase. However, the faces of cartoon characters vary significantly, different from human faces. Therefore, it may be much more difficult to build a good cartoon face detector, though it may be easy to build a high-performance face recognition classifier later.

## 2 Methods Explored

### 2.1 Naïve Template Matching(by Pixel)

We start with naïve template matching. More specifically, we use an image of waldo's face(fig:naive_matching_pixels) as a template. Then we match all the candidate windows of an image with this template. The candidate windows with a good matching results are classified as waldo.

Theoretically, since waldo's face does not vary too much across images, it is possible for the windows with waldo to have good matching results with the template. In order to address scale variance, we match in multiple scales. To address some of other variance, such as rotation, we can use more templates to match individually and a candidate will be classified as waldo if matches well with at least one of the templates.




Naïve Matching by Pixels

In implementation, we use cv2.matchTemplate, which is equivalent to do sliding windows of a specific scale at all possible positions, if matching with template directly is considered as a classifier. After every matching, we pick K candidates that matches best.

However, we could not find waldos on validation dataset. Moreover, it is super slow, which discouraged us from doing more analysis on the results. The problem of matching speed was tackled later by resizing the templates and images to smaller size.

Since our final methods also used the idea of template matching, more details about template matching is introduced in Final Method section.

## 2.2 Naïve Template Matching (by Features)

We attributed the reason of failure of to be the weakness of pixels. Therefore, instead of matching between pixels of candidate windows and templates, we match between features (sift and hog) of candidate windows and templates. Here, we are also using sliding windows approach. For the measure of matching result, we used sum of squared difference of features. The results are also bad and the algorithm is very slow.

## 2.3 Harr Features + Adaboost Classifier

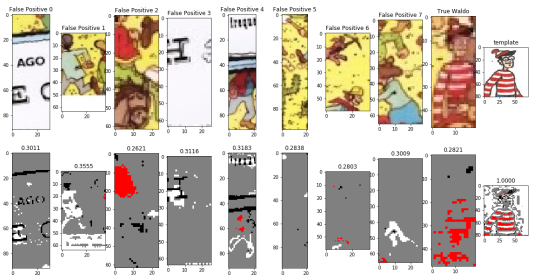
Inspired by Viola-Jones, we tried to use the same method for waldo. We built an Adaboost classifier on harr features of candidates to classify a window to be waldo or not waldo. The positive dataset consist of waldos cropped from given dataset and some other images of waldo from Internet. The negative dataset is background images cropped randomly from given dataset.

Although harr-features + Adaboost classifier is believed to be very fast, in our experiment the classifier is not that fast. It took more than 10 seconds to classify about $2 \times 10^5$ candidate windows. The time needed is dependent on the parameters, including the number of estimators used in Adaboost classifier and the number of sliding windows used. Our group don't want to use only harr-features + Adaboost classifier method to build a strong waldo detector. Instead, we hope to propose something novel, though may inspired by proposed methods.

## 2.4 Naïve color methods

Our first color methods is the stripes in waldo's shirts. When we search for waldo ourselves, we usually utilize the information or prior knowledge that waldo's shirts have salient red-white stripes if visible. Therefore, we try to use stripes as templates to do template matching. However, the result is always not good. Many other objects such as tents have similar stripes and are even more outstanding in terms of matching score.

The second color methods we tried is local color density distribution. Usually, if body is visible, waldo has black hair in upper-right area, red hat in upper-left area, red-white stripes in lower area. We may use the density of different colors in a few grids as a simple feature to indicate if an image is waldo or not. For simplicity, we picked 3 colors, black, white and red (??). Then we discretize the images based on thresholds set for these colors. We separate a window to $3 \times 2$ grid and calculate the color densities of 3 colors. If the color density values are close to a specified value, the window is assumed to be waldo.



Grid Color Histogram

This method is similar to gridded color histogram. The main difference is that the color discretization specifically designed for waldo's images and waldo's colors do not change much.

However, due to pixel resolution issue, the standard value for waldo in one image may not apply to other images. In some waldo images, the shirt is simply red and white. In other waldo images, there exist black pixels between red and white. The background can also introduce colors that terribly influence the results.

## 2.5 Random Linear Combinations + Adaboost Classifier

As for image classification problem, using support vector classifier(SVC) on sift or hog features, or enhanced by bag-of-words is a good choice. In our testing, we used another classification method other than SVC.

Inspired by neural network and Viola-Jones face detection algorithm, we proposed a classifier using adaboost classifier on a set of random linear features. Different from harr features used in Viola-Jones face detector, random linear features are simply random linear weights of the same shape as the image or data. For each weight, multiply the weights and data pixel-by-pixel and take the summation as the feature value. The feature value of image I with the $k$-th random weights $\mathrm{w_k}$ is

$$f_k = \sum_{i=1}^{h} \sum_{j=1}^{w} \mathrm{w_k}(i,j)\mathrm{I}(i,j) \tag{1}$$

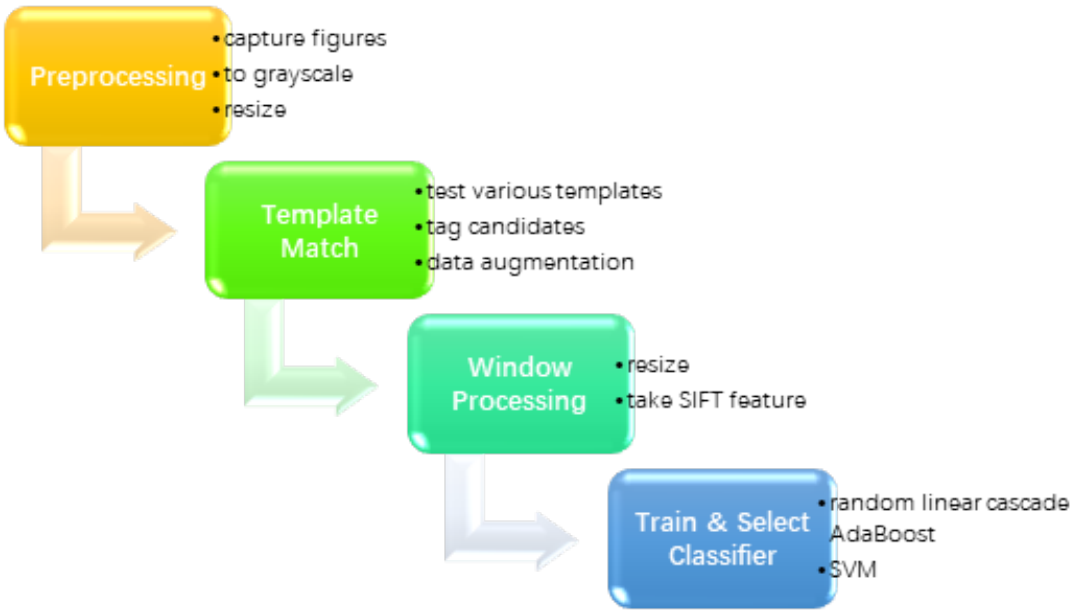



Figure 1: Pipeline of training process

where $h$ is the height of the window and $w$ is the width of the window.

The underlying assumption is that some linear combinations of pixels can be indicative of waldo, though individual pixel may not be a good indicator of waldo's identity. This is partly inspired by neural networks. Although random linear combinations seem to be nonsense, there must be some weights that happen to be good if we try many random weights. Adaboost can help find these good random weights.

This classifier can be further enhanced if we take random linear combinations of sift features instead of pixels. More specifically, for every image, calculate its 128-dimension sift feature. Then we use the sift feature to represent the image.

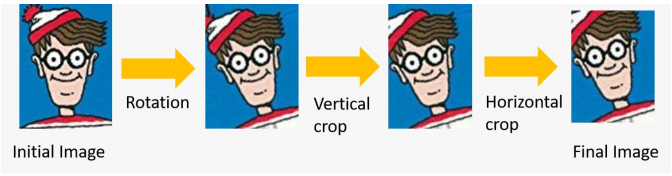
In our task, the probability of a window to be waldo is very small. If we want a sliding window + classifier to work, the false positive rate should be smaller than about $10^6$, which is very difficult. Therefore, we need to build a cascaded classifier to reduce false positive rate. It is possible that, since the actually important random weights are somehow determined by the data presented, if we train the cascade iteratively, the performance of these classifiers can be close to independent. However, this has not been proved.

The comparison between SVC classifier and random-linear-adaboost classifier is not extensively explored, either. Among the waldos cropped from our dataset, it is obvious that there are some natural clusters. Difference within a cluster is very small, whereas difference between clusters is significant. In all, the dataset is not good enough to compare the general performance of different methods.

A drawback compared to Viola-Jones face detection is that it can't utilize the image integral to speed up computation, so this classifier is better used in recognition in later stage rather than sliding window detection.

### 2.6 Data Augmentation

The given dataset is not large enough. The number of waldos are smaller than 100. In order to build strong classifiers, we need more data. Therefore, we perform data augmentation on the waldos and other characters cropped from the dataset. Since the intra-variance of waldo or other characters is limited, we only do translation, cropping and very small rotation.




Data Augmentation

Theoretically, we would like a perfect classifier that can detect waldos of all possible variance. However, in our application, the intra-class variance is relatively small. It is very difficult or even impossible to build a classifier that knows all variations of a specific characters. If we decrease our requirement on generality of the classifier, we can probably get classifiers that are better for our dataset.

Sometimes, waldo's entire body is visible. Sometimes, only the face is visible. At other times, face and incomplete body are visible. It will be difficult if we want to train a classifier that can work well in all of these situations. Therefore, we decided to build a classifier for face and a classifier for body respectively. For face classifier, we cropped the images in the dataset to be suitable as faces. Same for body classifiers. This greatly increased the classification performance in this specific problem.

## 3 Final Method

### 3.1 Overall Structure

The random linear weights + adaboost classifiers can easily achieve classification accuracy of about 0.98 on training data, which looks good for a classifier. However, since we have $10^5 \sim 10^7$ candidates for each image and typically only 1 character is within it, such good classifier is not enough in terms of speed and accuracy. We need simple region proposal method that work for

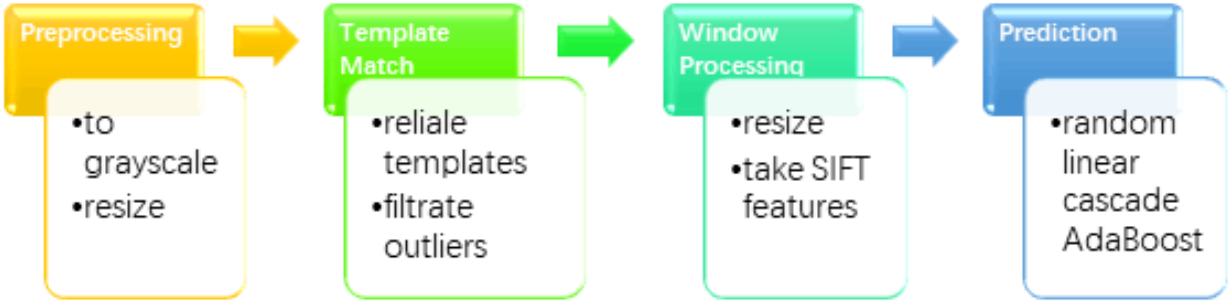



Figure 2: Pipeline of testing process

waldo's problem. The methods in naïve color methods section are not expected to be a good detector that can return one waldo from numerous candidate windows. If it can be a good region proposal method that returns hundreds or thousands of images, it is already good enough for us.

Later we found that some very valuable parts of templates are very good region proposal methods. Besides its performance as a region proposal method, the false positives returned by these region proposal methods are usually very different from waldo (at least from the perspective of us humans), thus making it easier to build classifiers in later phases. Our final method is part matching + Random Linear Adaboost Classifiers

## 3.2 Part Matching

Template matching is a technique for finding areas of an image that match (are similar) to a template image (patch). We need two primary components. One is the source image–the image in which we expect to find a match to the template image. Another is the template image–the patch image which will be compared to the template image. Our goal is to detect the best matching area.

To identify the matching area, we need to compare the template image against the source image by sliding it. However, we cannot know how big the faces or bodies of Waldo, Wenda and Wizard would be in test data set and in fact they are changeable in the train data set and validation data set so that it is hard to determine the size of sliding windows. As shown in 3.2, the face of Waldo varies from $11 \times 13$ pixels to $1105 \times 1441$ pixels. To handle this problem, we use the annotations to find each person in train set and validation set and output their figures. Later when we want to match some templates, we gain knowledge from the figures that how we should set the windows' size. Particularly, we set the window size of each template to equal ratio series so that we have more small windows and less big windows since the very large figures seldom appear in our data set. Moreover, using equal ratio series saves a lot of time than a series with equal difference of 1.

Img003: 11x13

Img032: 1105x1441

Different faces of Waldo

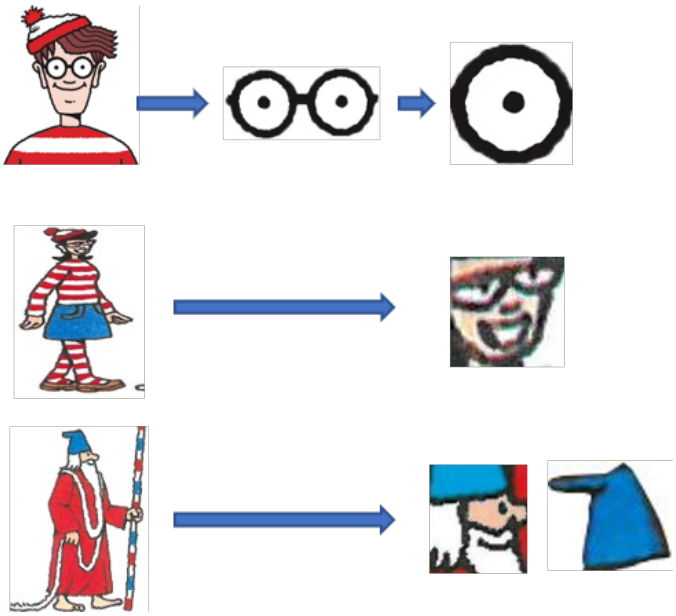
When sliding, we mean moving the patch one pixel at a time (left to right, up to down). At each location, a metric is calculated and returns to a score which represents how "good" or "bad" the match at that location is (or how similar the patch is to that particular area of the source image). There are six available methods provided in OpenCV package: 'CV _ TM _SQDIFF', 'CV _ TM _SQDIFF _NORMED', 'CV _TM _CCORR', 'CV _TM _CCORR _NORMED', 'CV _TM _CCOEFF' and 'CV _TM _CCOEFF _NORMED'. We do lots of experiments on these methods. When using method 'CV _TM _CCORR', 'CV _TM _CCOEFF' and 'CV _TM _SQDIFF', the highest matching scores are very large numbers and it is difficult for subsequent process to set a viable threshold on those matching scores and acquire some reliable candidates while 'CV _TM _SQDIFF _NORMED', 'CV _TM _CCORR _NORMED', and 'CV _TM _CCOEFF _NORMED' all normalize the scores from 0 to 1. However, in method 'CV _TM _SQDIFF _NORMED' and 'CV _TM _CCORR _NORMED', the scores of area similar to the template and area far from the template are both high, which also makes it hard to separate the candidates. Finally, we choose 'CV _TM _CCOEFF _NORMED', shown in equation (2) to calculate the similarity between the template and sliding windows because of its good performance in discrimination.

$$\mathrm{R}(x,y) = \frac{\sum_{x',y'} \left(\mathrm{T}'\left(x',y'\right) \cdot \mathrm{I}'\left(x+x',y+y'\right)\right)}{\sqrt{\sum_{x',y'} \mathrm{T}'\left(x',y'\right)^2 \cdot \sum_{x',y'} \mathrm{I}'\left(x+x',y+y'\right)^2}} \tag{2}$$

After finishing above processes, we get matrix of matching scores for each window size. We then set a threshold and pick local maximums great than the threshold in each matrix. Every local maximum represents an area probably identical to the template. According to the template's proportion and relative position of the whole figure, we enlarge the area to make sure that it contains people's whole figure and convey its size and ab-

solute position to next process.

In our extensive experiments, we find that the most significant point in template matching is selecting good templates. For Waldo, no matter the shape, size or color of his faces in our data set are whimsical. Consequently, simply matching his whole face would not gain any effective results as the score of true face area is very low. Fortunately, we notice that Waldo always wears round frame glasses where there is only large area of white and black pixels, which theoretically benefits for matching scores. To go a step further, one eye of Waldo appears more frequently in data set so that matching his eye is more accurate than matching glasses. Our experiments show that matching Waldo's eye can catch over 75% of Waldos in our data set. Similarly, Wizard also has some apparent features—his hat and white hair. We build two matching ways that one only uses his hat while another uses the combination of his hat, his hair and part of his face. Our experiments show that these two ways can capture about 80% of Wizards in our data set, even the Wizards in img003, img074, img075 that are not labeled in annotations. For Wenda, whose clothes are always covered by other people, she does not have any distinct feature. And her tilted glasses are noneffective since our sliding windows are all square. Since her faces are always sideways, we can use her face as a good template excluding her hair and cap which are similar to Waldo's to reduce the burden on classifiers in later process.



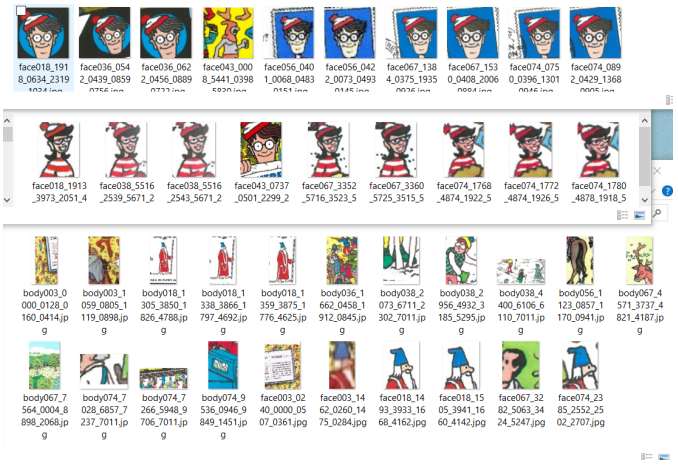
Selecting templates

In addition, another way to reduce the burden on classifiers and increase accuracy is leveraging feature of color. In the template of Waldo, black and white pixels occupy a large proportion while most pixels in the template of Wizard are blue and white. After matching in Gray channel and obtaining area with high score, we extract the area in original image and compute its number of pixels in color black, white and blue. Only when the pixels with specific colors exceed a threshold, such as 50% of all pixels, we consider this area to be a positive candidate.

### 3.3 Face and Body devided

As introduced in Data Augmentation section, we build face classifiers and body classifiers respectively. Although we can only get face from part matching, we can estimate the area of bodies from faces. Therefore, it is possible that we recover the whole body from part of the face.

## 4 Experiment Results



Results on Validation set

The overall performance of our methods are mostly determined by part matching. Since part matching returns only $10 \sim 100$ proposals, where most of false positives are obviously different from true positives from the perspective of human eyes, it is not that difficult to build a classifier that remove the false positives. However, if a waldo is classified to be negative by part matching, following classifiers can not recover it later. In most cases, faces of wenda and wizard are very clear and detectable. However, when waldo's face is too small, he can't be detected by matching the glasses. Maybe we have to use sliding window+classifier method to detect waldo by his tiny body.

## 5 Task Division

Cai Zhuo

- Try sift + bag of words + SVC
- Propose random linear weights + Adaboost

Wei Jianxin

- Propose the valuable parts of templates used in our final method
- Select candidates by color in Section 3
- Try naive sift for different templates

Vernon Cher Chu Xiong

- Try naïve template matching
- Try color methods
- Deal with all the dataset

# References


[1] Paul Viola and Michael Jones. Robust real-time face detection. International Journal of Computer Vision, 57:137–154, 05 2004.

[2] Charles Zitnick and Piotr Dollar. Edge boxes : Locating object proposals from edges. volume 8693, 09 2014.

[3] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. Measuring the objectness of image windows. IEEE transactions on pattern analysis and machine intelligence, 34, 01 2012.