Computer Vision for road signs detection
Project Report

Dargier Antoine
CentraleSupélec, MDS - SDI
antoine.dargier@student-cs.fr

Douilly Thomas
CentraleSupélec, MDS - SDI
thomas.douilly@student-cs.fr

January 31, 2023

Abstract

The objective of our work was to understand how we are able to detect road signs in all conditions, to implement our own algorithms, and to compare them to implemented methods when it is possible.

We began our study with a literature review to understand how elements can be detected in an environment. There are in fact a lot a different methods, and we decided to focus on two of them: Scale-invariant feature transform (SIFT) and Viola-Jones.

SIFT is a quite old computer vision method, able to detect similarities between two images. We used it in this project as a way to try to find road signs and traffic lights in dashcam pictures. We tried to code from scratch the main part of the method, namely the construction of the keypoints. However, due to the computation time, we made experiments on the equivalent method found in the OpenCV module. According to the cutting of our dataset, we were able to find satisfactory results of about 60% on the pictures dedicated to training.

The Viola-Jones algorithm uses computation of features, called Haar-like features. They allow the algorithm to detect specific lines, edges, or corners to detect the specific element. It was one of the most efficient method, able to detect elements in real-time. We tried here to compute the main steps of the algorithms (the computation of features, selection of the most relevant, test on new images). We obtain an accuracy of 60% on the new set of images, which is not so good, but acceptable for the size of our dataset.

1 Introduction/Motivation

1.1 Interest

Traffic signs detection has become nowadays a very important topic with the quick development and large amounts of time and money spent on self-driving vehicles and various kinds of help for driving. This comes with a significant improvement in techniques used to analyze the vehicle's environment thanks to computer vision.

The goal of this project is to try to detect traffic signs, either traffic lights, stops, speed limits or crosswalks. In particular, we want to reach two different steps :

- First, we want to be able to detect the presence or not of such signs on a picture, particularly on pictures of roads.
- Then, in case the presence of a sign was detected, we want our code to be able to understand it in order to say which sign it is and especially the behaviour the drivers (or the car) have to adopt when facing it.

The problem has become very important nowadays with the development of self-driving cars. Moreover, it can also be very useful in the IoT and all the new images and cameras in the cities for security, and

modelisation, ... If the solutions work well, are accurate and have a reasonable runtime, they can be quasi-directly implemented in the onboard computer of self-driving cars as part of an environment recognition module. It can be a solution too for GPS systems e.g. Waze, Google Street View, ..., which must take into account several factors, that must be updated regularly:

- Speed limitations (often displayed for the driver)
- No-way streets or restricted areas/roads
- Traffic lights

1.2 Limits to overcome

In order for computer vision algorithms to be effective in the context of driving safety, they need to be able to accurately identify different road signs in a variety of different conditions. These conditions can include different weather conditions, such as sunny, rainy, and snowy, as well as different lighting conditions and different resolutions depending on the speed of the cars. This is a challenging task because computer vision algorithms are often sensitive to changes in these parameters, and small variations in the image can cause the algorithm to produce inaccurate results. Therefore, there is a need for computer vision algorithms that are robust to these variations and can produce precise and accurate results in real-world scenarios. This is an active area of research in the field of computer vision and is important for the development of autonomous driving systems. For some applications, algorithms need moreover to be in real-time, because for instance with self-driving cars, the decision must be taken very quickly to avoid accident.

1.3 History of research

Computer vision research on object detection has a long history, dating back to the early days of computer sciences. Early research focused on developing algorithms for detecting simple geometric shapes, such as lines and circles. In the 1980s and 1990s, researchers began to focus on more complex objects, such as faces and cars.

One of the major breakthroughs in object detection was the development of the Viola-Jones object detection framework in 2001, which used a combination of Haar-like features and a cascaded classifier to detect faces in images. This framework was used as the basis for many subsequent object detection algorithms.

In recent years, convolutional neural networks (CNNs) have become the dominant approach in object detection. The region-based CNN (R-CNN) and its variants such as fast R-CNN and faster R-CNN, were proposed in 2013 and 2015 respectively. These methods used a CNN to extract features from regions of an image and then used a separate classifier to determine whether each region contained an object.

YOLO (You Only Look Once) and Single Shot MultiBox Detector (SSD) are two popular one-stage object detection methods that have been proposed in recent years. Both of these methods use a single CNN to simultaneously predict object bounding boxes and class probabilities in an image. These methods are faster and more efficient than the two-stage methods, but slightly less accurate.

Recently, RetinaNet, a one-stage object detection algorithm proposed in 2017 by researchers from Facebook AI, has been widely used and it has become the state-of-the-art method in object detection.

Overall, the field of object detection has seen significant progress in recent years, with deep learning-based methods achieving near-human-level performance on many benchmark datasets.

1.4 Examples

Object detection algorithms can be used in a variety of applications, including:

- Self-driving cars: Object detection algorithms are crucial for the safe operation of self-driving cars, as they enable the car to detect and respond to other vehicles, pedestrians, and obstacles on the road.

- Surveillance: Object detection algorithms can be used in surveillance systems to automatically detect and track people, cars, or other objects in real-time video feeds.
- Robotics: Object detection algorithms can be used to enable robots to navigate and interact with their environment. For example, a robot with an object detection algorithm can identify and pick up specific objects in a warehouse.
- Medical imaging: Object detection algorithms can be used to analyze medical images, such as identifying tumours in scans or detecting abnormalities in X-rays.

These are just a few examples of the many ways object detection algorithms are being used today and as technology continues to improve, it is likely that these algorithms will be used in even more innovative ways in the future.

2 Problem Definition

The goal of this study is to develop a system that can detect and classify road signs in images from a dataset. Specifically, we aim to identify traffic lights, crosswalks, speed limit signs, and stop signs in the pictures. To accomplish this, we will utilize techniques from computer vision and image processing to analyze the images and detect the presence of road signs: SIFT and Viola-Jones algorithms. Once a road sign is detected, we will then use machine learning algorithms to classify the sign and determine its type. A bigger deep into the problem for each one of the two considered method will be done later, in the Methodology and Evaluation sections.

3 Related Work

The field of computer vision has been the subject of extensive research and documentation over the last decades. One of the key areas of focus within this field has been the detection of objects using various techniques. Some of the most well-known and widely used methods for object detection include SIFT and Viola-Jones. These methods have been put to the test in a wide range of applications and have been shown to be highly effective. However, despite their established success, it is still valuable to continue researching and developing these models. This is because by focusing on specific tasks, we can gain a deeper understanding of their capabilities and limitations, and identify opportunities for improvement. This ongoing research will not only enhance our understanding of these established methods but also help us to develop new and more sophisticated techniques for object detection in the future.

These searches are pretty old, but may be still powerful enough to the current challenges we already told about. This is what we aim at with this project, namely trying to evaluate the performances of these algorithms on more recent data in terms of accuracy and time computation. Indeed, due to the recent breakthroughs in deep learning and self-driving vehicles, we need to have very accurate and quick methods to analyse the car's environment. Our project roots in this long-run research by comparing two old computer vision algorithms and see if the best of the two could still have an interest nowadays, even with the surge of very powerful tools such as

4 Methodology

In the following section, we will describe the steps taken during our study.

4.1 Data collection process

The dataset used has been found on a Kaggle Competition on quite the same problem addressed here. All the data is collected by a Python script we coded which imports in a dictionary the picture as a numpy matrix which has been truncated to keep only the road sign (the original data contains the edges of the box where the sign is), as well as the category of the sign ("road sign", "traffic light", "Stop" or "Crosswalk") which was kept into the xml file linked to the png picture.

4.2 The SIFT method

The Scale-invariant feature transform or SIFT is a classical computer vision method developed in 1999 by David Lowe [4]. It allows to detect similarities between pictures, which can be used in our case to detect similarities between our test set and real pictures.

4.2.1 Architecture

The steps used in the SIFT method are quite common in Computer Vision even though we can denote several distinctive features. Let one of our picture, denoted I, containing pixels written $I(x, y)$ according to their (x, y) position in the picture. Each of these steps have been coded from scratch in Python as part of this project.

- The first step is to apply to this picture a Gaussian Filter parameterized by its standard deviation σ, which means that for a gaussian kernel written $G(\sigma)$, we compute the convolution with our picture $G(\sigma) * I$.
  Given a parameter k, we will apply the same way Gaussian Filters of parameters $k\sigma$, $k^2\sigma$, $k^3\sigma$, ... to our picture.

- Then, we will compute the Difference of Gaussians (DoG), which is for some i, the difference between the gradient with parameter $k^{i+1}\sigma$ and the gradient with parameter $k^i\sigma$:

$$D(\sigma, k, i) = (G(k^{i+1}\sigma) - G(k^i\sigma)) * I$$

- We can repeat these steps by taking as entry the same picture with a resolution divided by the powers of 2. This allows us to build what is called the DoG pyramid of the picture, similar to the example 1.

- In order to find potential extrema, we have to look at each pixel whether its neighbours are all greater are lower. For each pixel of coordinates (x, y, i) of the DoG, is considerate as a neighbour all pixels $(x+dx, y+dy, i+di)$, where $dx, dy, dz \in \{-1, 0, 1\}$

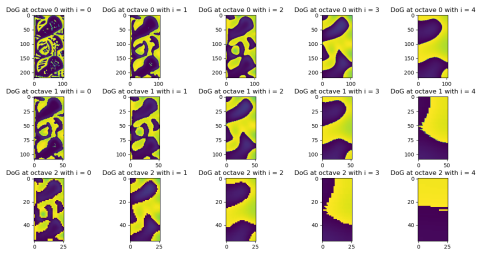

Figure 1: Overview of a DoG pyramid

- This process gives us a high number of Keypoints, so that we can use filters on pixel so that we only keep pixels with a sufficient magnitude and that are not on the edges of the picture.

- Now that we have our estimated keypoints, we can calculate the x and y gradients on the picture in order to calculate the magnitude and orientation of it using the two formulas Magnitude $= \sqrt{G_x^2 + G_y^2}$ and Orientation $= \arctan \frac{G_y}{G_x}$.

- Finally, we can compute the descriptor of each keypoints by summing up all magnitudes and gradients of points situated in a 16x16 neighbourhood of the considered keypoints.

Once this is done, we will use the post-processing proposed by this Open Classrooms course [1]

First, we used the K-Means algorithm to create a "bag of words" from the descriptors found in the previous steps. The K-Means algorithm creates some categories of features. With this, we can build an histogram displaying the proportion of each feature in each picture. This histogram will feed a supervised classification model such as KNN in order to be linked to the category of the picture. As a quick reminder, each picture was labelled by its category ("Traffic Light", "Speed Limit", "Crosswalk" or "Stop") and by a list of coordinates, edges of the box where the corresponding sign was in the picture in order to crop it correctly.

4.2.2 Advantages

One of the biggest advantage of SIFT method with regard to other computer vision algorithms is that the features can be detected even when the orientation or the zoom is different from an image to another. This is a particularly precious advantage in our case as real-life pictures are never taken in the same way, may that be in angle or in distance from the considered item (in our case road signs).

4.3 The Viola-Jones method

The Viola-Jones algorithm [6] [2] is a computer vision method developed by the researchers Paul Viola and Michael Jones in 2001 [5], in order to detect objects from numeric pictures. At the origin, the goal was to detect faces in pictures, but it can be now used for a lot of detection, such as cars or planes. It was a big improvement in computer vision, because this method allows real-time detection, and was one of the most efficient.

4.3.1 Architecture

We will now describe how this algorithm works. There are 4 main steps:

- Selecting Haar-like Features: Haar Features are essentially collections of pixels in rectangular shapes, with one zone where pixels are summed and another one where they are subtracted. This allows to detect lines, edges, corners, among the shape of the rectangle. Here are some examples:

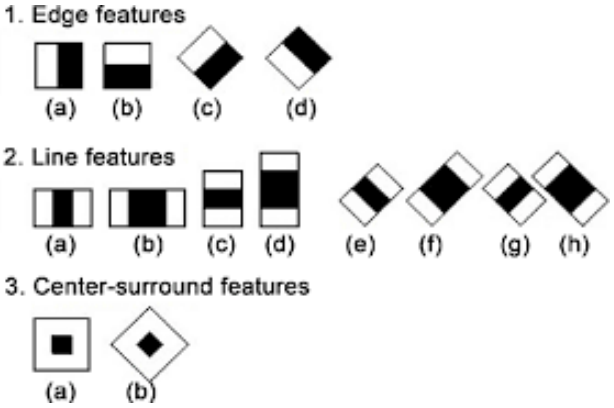

Figure 2: Illustration of Haar-like features

- Creating an Integral Image: the integral image is a version of the images where each pixel is the sum of the above and left corresponding pixels in the source image. It's just a way to compute Haar-like features very quickly and it avoids to calculate the sum of pixels again and again

- Running AdaBoost Training: with that architecture, we obtain a lot of features, because the different Haar-like features are tested at each position of the images. The AdaBoost Training or other classifier allows us to keep only the most relevant features for our study. It will be very interesting to have a fast algorithm

- Creating Classifier Cascades: it's a group of weak classifiers that we train to be good together. These classifiers are trained on thousands of positive and negative examples of elements, and they work by looking for specific patterns of pixels, such as edges and lines, that are characteristic of the search elements. The algorithm starts by scanning the image at different scales, and at each scale, it applies a set of classifiers to the image. If a classifier detects a feature, it will pass the region of the image where the feature was detected to the next classifier. This process is repeated until a final decision is made about whether or not an object is present in the image. This methodology is called cascade detection.

4.3.2 Advantages

The algorithm has a lot of advantages: it is known for its high speed and good accuracy, making it well-suited for real-time applications. It is relatively simple to understand, and it needs not such much data compared to other methods. There is no need of resizing images, because the model does it alone, and the results are much more interpretable than contem-

porary models.

However, there are still disadvantages. The training time is very slow, and the results are extremely dependent on the quality of the data we give as inputs. The model may be sensitive to very high/low exposure (brightness)

5 Evaluation

5.1 Dataset

For our study, we will work with a Kaggle Dataset consisting of a collection of traffic signs. This dataset contains 877 images of 4 distinct classes targeting the goal of road sign detection, and annotations provided in the PASCAL VOC format. The studied classes are:

- Traffic Lights
- Stops
- Speed Limits
- Crosswalks

Figure 3: Illustration of the Dataset given by Kaggle

These classes will allow us to make a first comparison between all algorithms and are a basis on which it would be possible to define finer classes (50mph speed limit, red traffic light, ...) using new techniques (NLP and others), but that part will not be covered in this study.

5.2 The relevant metrics

To evaluate the performances of the 3 algorithms and compare the results, we can while consider different quantitative values. First we may want to consider the confusion matrix, because we have annotated data, and we can compare our results with reality. In particular, one thing to follow is the number of false positive (and negative). Indeed, for applications such as self-driving cars, safety is essential and it is not possible to have any false negative (the algorithm does not detect anything even though there is a traffic sign). We will then try to have this number as lower as possible. Another metric really important for these algorithms is the time of detection, so we need to measure whether the algorithms take too much time or if they are heavy energy-consumers. Indeed, the industry requires the fastest possible algorithms in order to be able to integrate them into navigation systems.

5.3 Our results

5.3.1 SIFT algorithm

We tested two implementations of SIFT method to determine the keypoints on the picture : one built from scratch using the steps described in the previous section and one using the "SIFT_create" method coming from the OpenCV library. The parameters of the model (in particular the values suited for the thresholds) have been taken from the SIFT Github [3], which also proposes a version of the SIFT method from scratch. However, we quickly found out that the computation time for the OpenCV method is a lot better than for the method built from scratch. Therefore, from now on, we will only consider the OpenCV one as basis for our experiments.

As explained before, once we obtain the keypoints are obtained, we will use a K-neighbours architecture to determine which kind of roadsigns is on the picture. To do so, we made tests based on the global accuracy metrics to choose the best parameter used for the architecture. The results of these tests are shown below in Figure 5. As we can see on it, with a number of neighbours chosen between 40 and 50, we are able to reach quite safely a 70% accuracy on the training set. For the following, we will use the medium value of 45 neighbours.

Now that we fixed the parameter of our K-neighbours architecture, we can see the results it give on our test dataset. To determine the consistency of

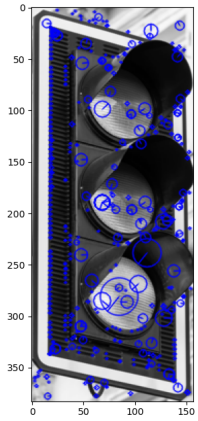

Figure 4: Keypoints found using the OpenCV SIFT method on one picture of the dataset

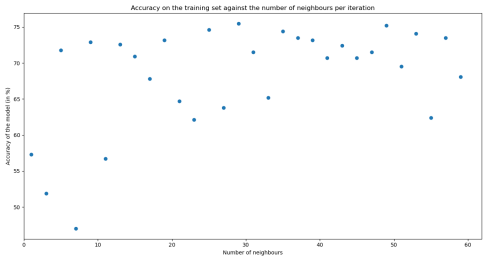

Figure 5: Accuracy on the training set with SIFT method according to the number of neighbours considered by the KNN

our model, we built the confusion matrix shown on Figure 6, based on the four categories our test data is labelled with, namely "Traffic Light", "Speed Limit", "Crosswalk" and "Stop".

As we can see on it, the method predicts pretty well the presence of speed limits on the pictures, reaching an overall accuracy of 68.9 %. However, on all the remaining categories ("Traffic Lights", "Crosswalk" and "Stop"), the results are pretty bad, what could be solved by considering two parameters : the training dataset and the parameter of the K-Means algorithm.

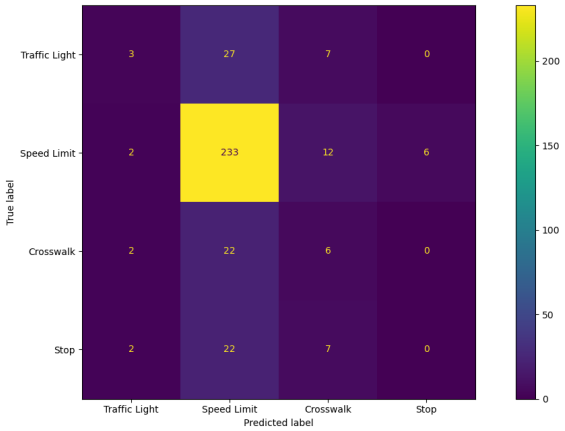

Figure 6: Confusion Matrix of the OpenCV SIFT model on the test set.

We tried to see the effect of this second parameter. The results are shown in the following picture :

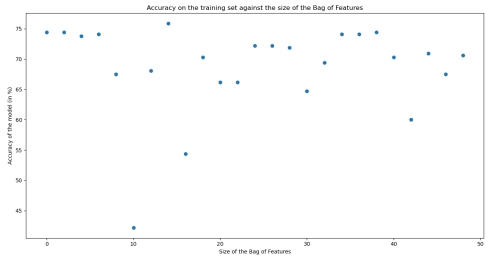

Figure 7: Evolution of the Accuracy of the SIFT model according to the number of neighbours considered by the K-Means method

Nothing really changed here and this is probably due to the quality of our training dataset, which is maybe too imbalanced between all categories. All the same, the results obtained with our SIFT model are respectable with almost 75% of accuracy in predictions on the training dataset and around 60% (Figure 8) on the testing dataset.

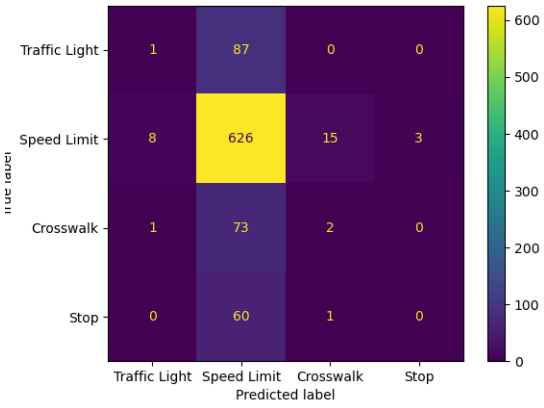

Figure 8: Evolution of the Accuracy of the SIFT model according to the number of neighbours considered by the K-Means method

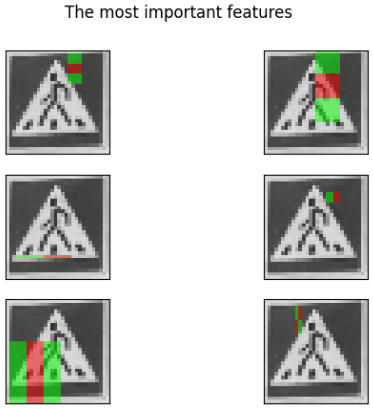

Figure 9: Illustration of the Haar-like features for crosswalks

Figure 10: Detection of speed-limit signs

5.3.2 Viola-Jones algorithm

First, we have implemented the Haar-like features and the integral image. We use for that already developed function of OpenCV. For that, there was an important work of data preparation, organizing the data into two folders of negative and positive elements for each of the four categories. After selecting the best features, we obtain that typical features: Then, we have to create the cascade classifier. The corresponding library in OpenCV is no more up to date, so we have succeed to implement it by ourselves. We use a software called "Cascade-Trainer-GUI". We create then four classifiers for the four classes. We obtain pretty good results with positive examples, but some elements were detected on negative pictures, probably because we haven't enough pictures in different context. Finally, to classify the pictures, we select the category with the most of elements detected. If there are some equality, we classify them randomly. With this method which is of course improvable, we have obtain an accuracy of 60%, and the following confusion matrix:

6 Conclusion

Along this project, we have developed in Python two methods used to detect road signs and classify them between four categories on pictures. We first developped the SIFT algorithm, an old but quite robust method which is used to find similarities between two pictures. The results are pretty good on our testing dataset with an accuracy varying between 60 and 75%, even though the confusion matrix we obtain should suggest to try the model with other training datasets. Then, we also developped the Viola-Jones algorithm, with similar results. But, even with a 60% - accuracy, the confusion matrix obtained seem a little bit better than for the SIFT model, with, in particular, significally better results on all categories except the "Road Sign" one, which is dominant in our

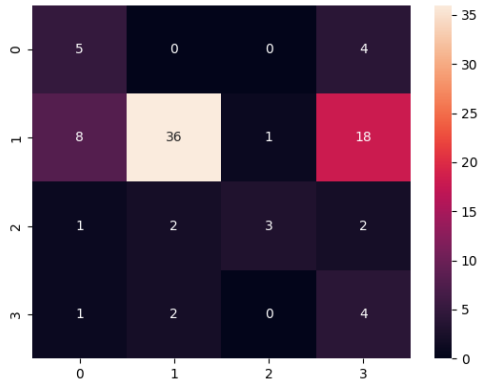

Figure 11: Confusion matrix on the test set - Viola-Jones model

dataset and explain the medium accuracy rate.

References

[1] Détectez et décrivez efficacement les zones d'intérêts dans une image. 4

[2] Facial detection - understanding viola-jones algorithm. 5

[3] Sift. 6

[4] D.G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999. 4

[5] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, 2001. 5

[6] Yi-Qing Wang. An analysis of the viola-jones face detection algorithm. *Image Processing On Line*, 4:128–148, 06 2014. 5