Machine Learning Project Image Segmentation for Object Detection and Target Tracking using the ATLASCAR

Bruno Mendes 68411 Pedro Silva 72645

Abstract—This document contains a description of the process of developing a project for the subject of Machine Learning, which is part of the fourth year of University of Aveiro's course Engenharia de Computadores e Telemática, and falls within the scope of Computer Vision. It showcases the work done, what was considered most important, the main goals, along with the problems and solutions found in order to develop this tool. Suggested and accepted as a final assessment item for the subject, this tool was created with the intention that it would allow for object detection and its use for the ATLASCAR project.
Through an approach that is intended to be simple and organised, this report starts by setting the context and motivation that back this project up. The features, ideas, problems and solutions are then presented, following roughly the real chronological order in which they came about throughout the development of the project. Finally, the main conclusions are discussed, going over the acquired knowledge, the main goals and their completion, and the enriching characteristics of the project, technical-wise and social-wise.

Index Terms—Machine Learning, Computer Vision, Object Detection, Template Matching, K-Nearest Neighbors, K-Means, Correlation Maps.

1 Introduction

In the context of developing the first project for Machine Learning, a subject that is part of the fourth year of University of Aveiro's course Engenharia de Computadores e Telemática, this idea of creating a tool that would allow to create a "`interest zone`" in an image that can be used for object detection for objects present on the road and the consequent labelling of these said objects was suggested and accepted as a final assessment item. It is interesting to create tools that have real use, particularly in the scope of a bigger project. In this case, the work shall prove of particular interest in the context of the master's thesis of one of the authors, focusing on semi-automatic labelling of objects using adaptive perception for the ATLASCAR project of the LAR personnel of the DEM department of Aveiro University.

This document is made up of 4 sections. After this introduction, the motivation behind this project is explained in section 2, contextualising the work carried out. Then, in section 3, the dataset tool is exposed, discussing some of the problems found and solutions developed in order to obtain the results showcased. Afterwards, in section 4, the machine learning algorithm used in this project will be showcased as well as its results for the experiment at hand. Finally, in section 5, some conclusions about the development of this project are presented.

2 The motivation

In the scope of the master's thesis of one of the authors, which focuses on semi-automatic labelling of objects present in the road that can be detected by the sensor setup of the ATLASCAR2 indicated in figure 2. The idea behind this is that, using a series of pictures of a scene, through a technique called Template Matching it is possible to define a zone in the image that can be labelled as a "`interest zone`"

for the target object that we wish to label. This technique involves the use of templates to make a first guess of the template object in the scene, indicated in figure 3, and to define a ground truth a bouding box is designed around the target object in the scene, indicated by the white square in figure 3. Furthermore, in the context of image optimisation, the use of correlation maps between images proved to be the best way to try to visualise and determine the "interest zone" of a given object in the scene. Not only that this also allows us to visually understand and measure the effect of the optimisation and its success. As such, in order to visualise the colour maps, associating colour from each pixel of the image by extracting the corresponding rgb value from the corresponding picture seemed like the way to go.

The tools created in this project were built in a way that would best suit their use in the context of the master's thesis of the student Pedro Silva. The datasets, tools for loading information and its internal structure where developed based on other datasets provided by other people that have worked in the ATLASCAR project.

Fig. 1: ATLASCAR2

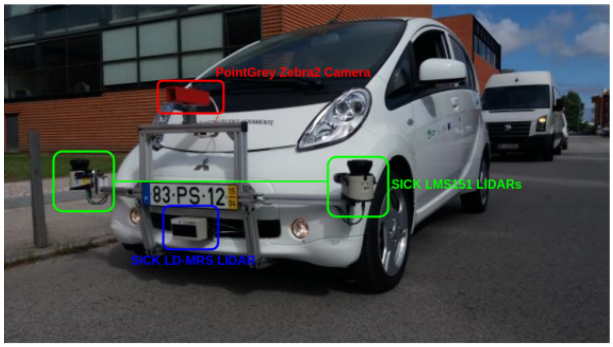

Fig. 2: ATLASCAR2 Sensors

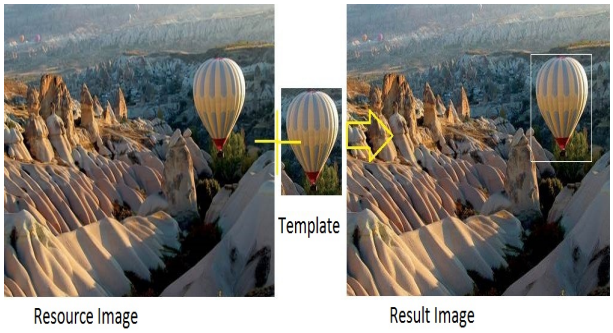

Fig. 3: Template Matching Example

3 Dataset Tool

As stated before the datasets, the tools for loading and storing information, used in this project were developed based on datasets created in previous experiments with the ATLASCAR using its sensors.

These datasets were saved in the .rosbag format, which is a ROS – Robot Operating System package that contains a set of tools used for recording from and playing back data from ROS topics present in the ATLASCAR system. From these experiments, 3 rosbag files were created, one from each experiment with the ATLASCAR.

Following this, using a Python script, we extracted the frames from each dataset by first turning the rosbag into a .mp4 video file and from that file we saved each frame onto a folder in the .jpg format in a 400x400 resolution.

From the first dataset, 2589 pictures were created as seen in figure 4. From the second dataset, 128 pictures were created as seen in figure 5 and from the third dataset, 699 pictures were created as seen in figure 7.

Fig. 4: Frame example from first dataset

Fig. 5: Frame example from second dataset

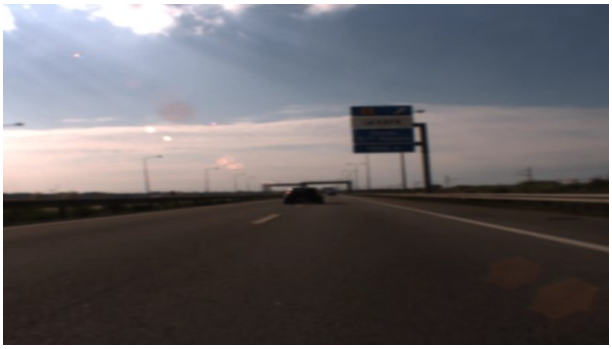

Fig. 6: Frame example from third dataset

The templates used in the `Template Matching` algorithm are screenshots taken from the frames of the dataset. These are some of the templates used in this project:

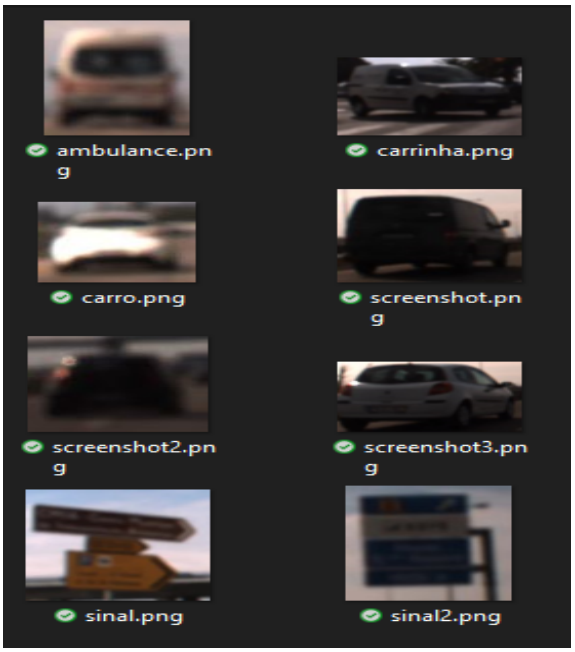

Fig. 7: Template Examples used in Template Matching

4 Machine Learning

In order to determine the "`interest zone`" of a given object in a scene to be used in conjunction with the `Template Matching` technique various `Machine Learning` algorithms were used in order to improve the image optimisation and the correlation map construction processes. From all of these algorithms, the `Knn-Means` algorithm proved to be one of the most suitable algorithms to use in this context according to a certain `Python` script: `compare_algorithms.py`.

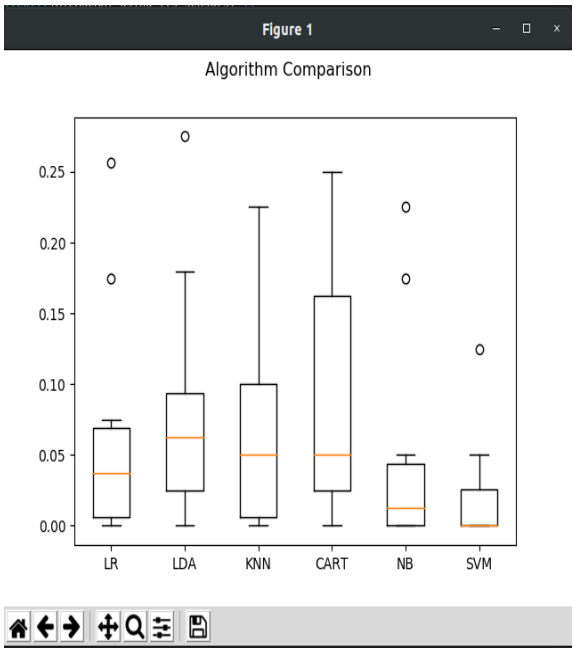

Fig. 8: Algorithm Comparison Results for a Frame Dataset

	Mean Accuracy	Standard Deviation Accuracy
LR(Logistic Regression)	0.065641	0.080721
LDA(Linear Discriminant Analysis)	0.082949	0.080421
KNN(K Neighbors Classifier)	0.065128	0.066305
CART(Decision Tree Classifier)	0.095577	0.088758
NB(GaussianNB)	0.050064	0.077439
SVM(SVC)	0.022564	0.037837

Fig. 9: Algorithm Comparison Results for a Frame Dataset

4.1 Knn-Means Algorithm

The `k-nearest neighbors` algorithm is a `Unsupervised Machine Learning` algorithm which is used when you only have input data (X) and no corresponding output variables and the ultimate goal of this algorithm is to model the underlying structure or distribution in the data in order to learn more about the data. This algorithm can be used for both data `classification` and data `regression` and in both cases the input consists of the k closest training examples in a given `feature` space. The output of this algorithm depends on whether this algoritm is used for `classification` or `regression`.
In `knn-classification`, the output is a class membership, in which an object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its `k` nearest neighbors where `k` is a positive integer, typically small. If `k = 1`, then the object is simply assigned to the class of that single nearest neighbor. In `knn-regression`, the output is a property value for the object and this value is usually the average of the values of its `k` nearest neighbors.

Input:
- K (number of clusters)
- Training set (no labels)

Randomly initialize K cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$
Repeat {
Cluster assignment => step
for i = 1 to m
$c^{(i)}$:= index (from 1 to K) of cluster centroid closest to $x^{(i)}$
Move centroid => step
for k = 1 to K
μ_k := average (mean) of points assigned to cluster k
}

Fig. 10: Knn-Means Algorithm

In the context of this project, the `K-Nearest Neighbors` algorithm will be used in order to perform `K-Means` clustering on the pixels of the given frames from the dataset, or in other words, this algorithm will be used to compress each frame given by the dataset. In order to do this, we first have to run the `K-Means` algorithm on the colors of the pixels in the given frame and then map each pixel to its closest centroid.
First we load the frames from the dataset and extract each RGB dimension from each frame by transforming into it into a numerical format for computations and divide them by 255 so that all values are in the range 0-1. Then we

reshape the image into a mx3 matrix where m is the number of pixels and each row will contain the red, green and blue pixel values. This matrix is equivalent to the dataset matrix X that is gonna be used by the `K-Means` algorithm. After that, we need to initialize the centroids using the `kMeansInitCentroids` function, developed in `Lecture 07` of the `Machine Learning` course. This function initializes K centroids that are to be used in the K-Means algorithm on the dataset X by first randomly reordering the indice examples of the dataset by using the `MATLAB randperm()` function and based on those indices we take the first K examples of these indices as centroids. After we calculated the centroids we can use the `runKMeans` function, developed in `Lecture07` of the `Machine Learning` course, for a set number of iterations.

For the purpose of this project we assumed the defaults values of K as 16 and the number of iterations as 10.

After all of this is set and done, we will move on to the image compression proccess using the clusters provided by the `K-Means` algorithm. In order to do this, we first have to find the closest cluster members using the `findClosestCentroids` function, developed in `Lecture 07` of the `Machine Learning` course, that computes the closest centroids based on the `Euclidian` distance between each example and every centroid.

Finally, we can now recover the image from the indices given by the clusters by mapping each pixel to the centroid value and we can now display the compressed image by first reshaping it into its proper dimensions alongside its original image and compare the results obtained.

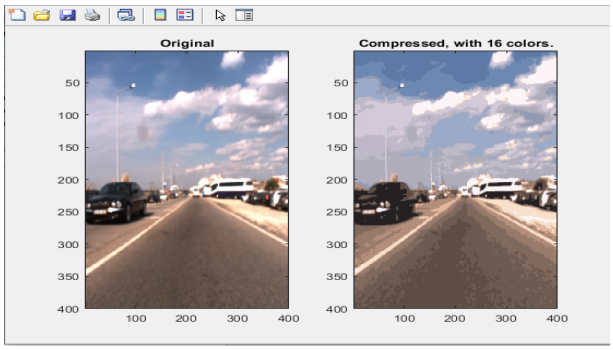

Fig. 11: Frame from first dataset compressed with 16 colors

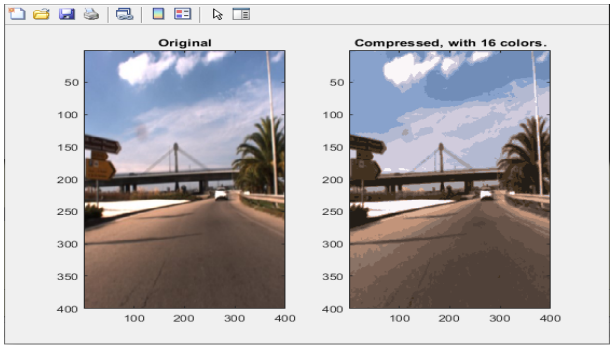

Fig. 12: Frame from second dataset compressed with 16 colors

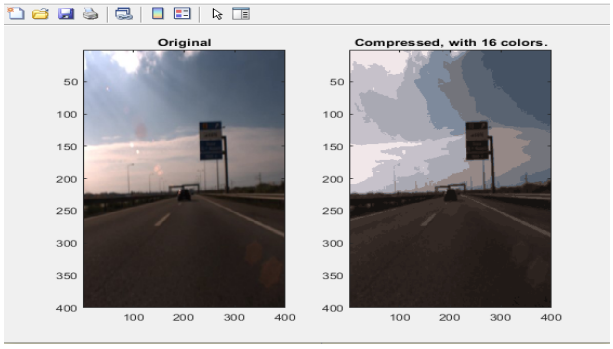

Fig. 13: Frame from third dataset compressed with 16 colors

After the compression proccess, we can proceed to the `Template Matching` proccess in the spatial domain and first we load the compressed image as a `template image` and we load the `target image` which is the actual template used for the `Template Matching`. After that we can construct the correlation map between these two images by using the `MATLAB 'corr2'` command function. This function is

used to find the correlation coefficient between the images. Basically, the `target` image(template) is placed over the `template` image(compressed result) and a correlation coefficient for each pixel in the `template` image is found to construct the correlation map. After sliding throught all of the pixels in the `template` image, the maximum coefficient is obtained from the map and pixel position with the maximum value is assumed to be the starting point of the target image, which is the `image template` used for the `Template Matching`.
One example of this `correlation map` can be seen in figure 14.

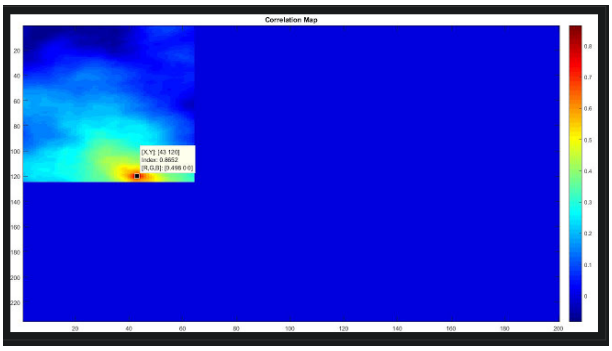

Fig. 14: Correlation Map Example

This `correlation map` defines the `"interest zone"` of the target object that was referenced earlier and this space can be used to narrow down the search area of a target image in a template. Such results can be seen in figure 15 and in figure 16.

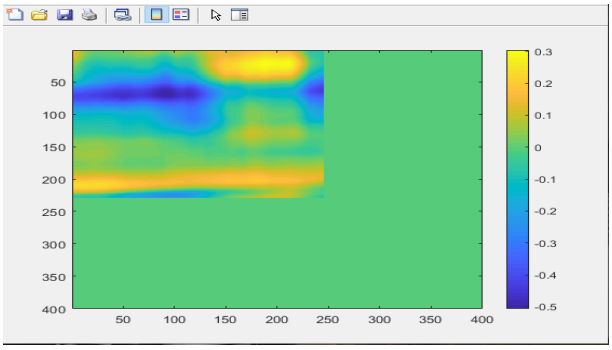

Fig. 15: Correlation Map between target and template images

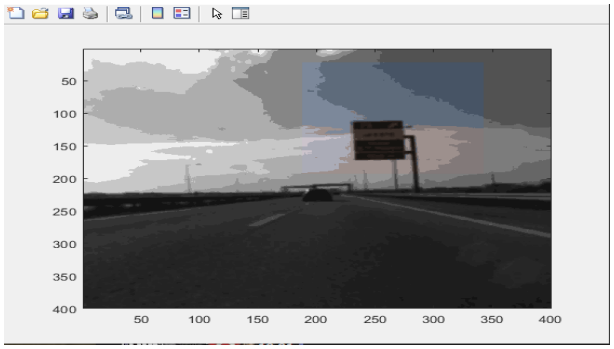

Fig. 16: Template Matching Spatial Domain Result

As we can see in the correlation map the yellow parts represent the maximum coefficient obtained from the map and from the pixel position with the maximum value we can assume that it is the starting point for the `target` image which is the `template` object that we are using in the `Template Matching`.

4.2 Image Results

Fig. 17: Template Object used for the First Dataset

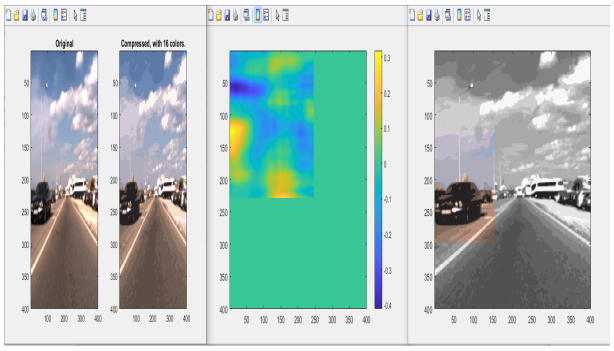

Fig. 18: Template Matching with Correlation Map of a Frame Image from the First Dataset

Fig. 19: Template Object used for the Second Dataset

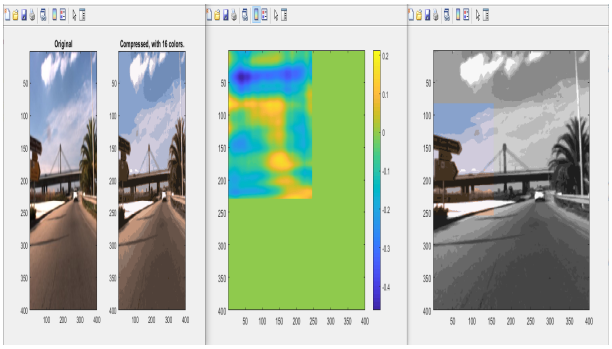

Fig. 20: Template Matching with Correlation Map of a Frame Image from the Second Dataset

Fig. 21: Template Object used for the First Dataset

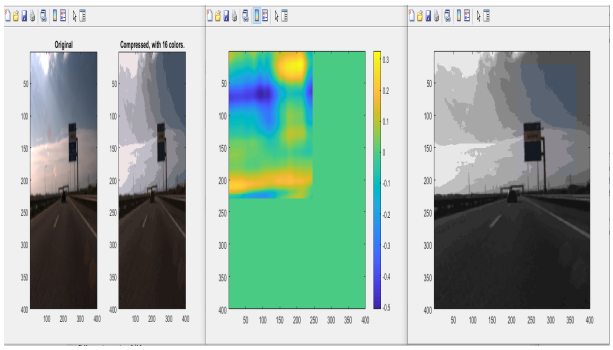

Fig. 22: Template Matching with Correlation Map of a Frame Image from the Third Dataset

4.3 Other Results

These are some of the results with different values of `k` and a different `number` of iterations.

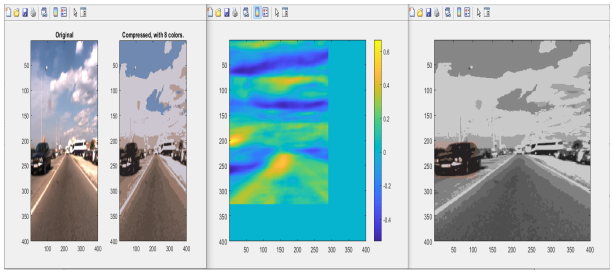

Fig. 23: Template Matching with Correlation Map of a Frame Image from the First Dataset with K=8

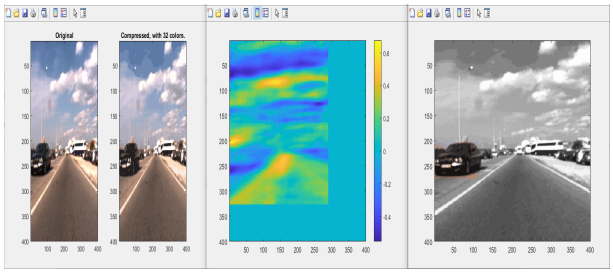

Fig. 24: Template Matching with Correlation Map of a Frame Image from the First Dataset with K=32

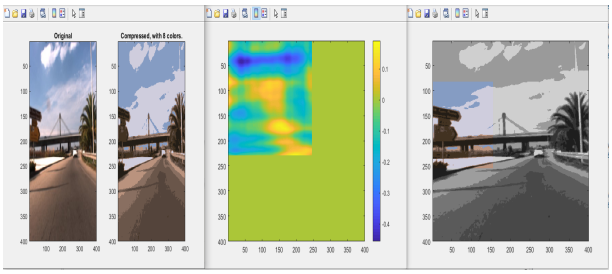

Fig. 25: Template Matching with Correlation Map of a Frame Image from the Second Dataset with K=8

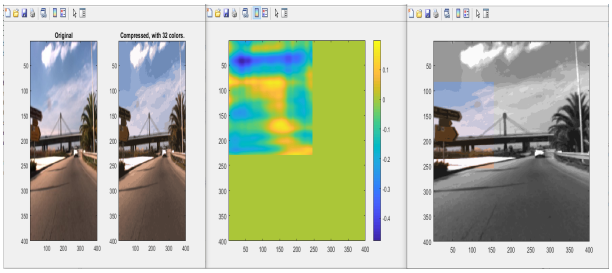

Fig. 26: Template Matching with Correlation Map of a Frame Image from the Second Dataset with K=32

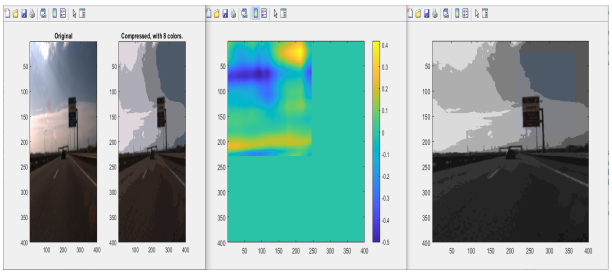

Fig. 27: Template Matching with Correlation Map of a Frame Image from the Third Dataset with K=8

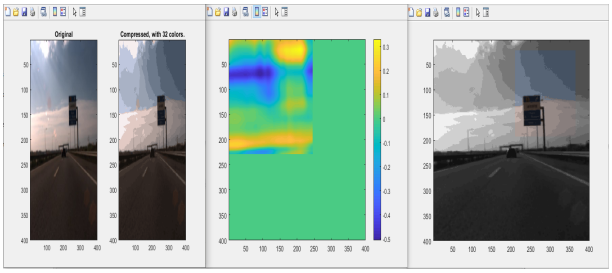

Fig. 28: Template Matching with Correlation Map of a Frame Image from the Third Dataset with K=32

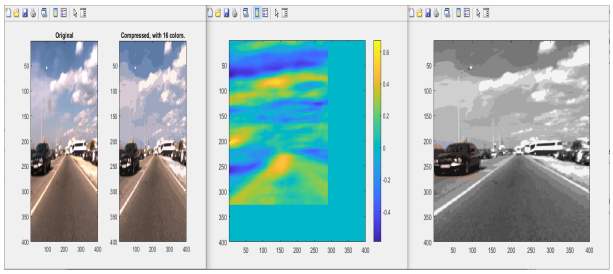

Fig. 29: Template Matching with Correlation Map of a Frame Image from the First Dataset with 20 iterations

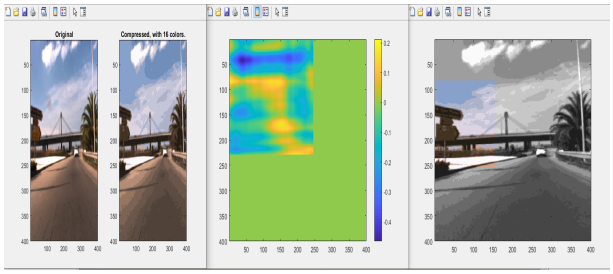

Fig. 30: Template Matching with Correlation Map of a Frame Image from the Second Dataset with 20 iterations

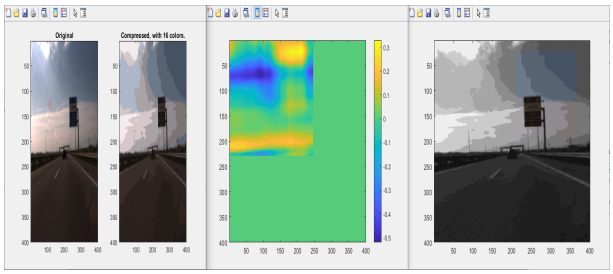

Fig. 31: Template Matching with Correlation Map of a Frame Image from the Third Dataset with 20 iterations

4.4 Train, Test and Accuracy Results

It's possible to determine the `accuracy` results of this algorithm by first handling the data by opening the dataset from the `.csv` file and split the data randomly into `train` and `test` datasets(a standard ratio of 67/33 was used to split the data into train and test datasets). Next we calculate the distance between the two data instances by using the `Euclidean` distance measure, in order to predict the data. This is needed for the next step which is to locate the k most similar data instances in the training

dataset for a given member of the test dataset and in turn make a prediction. Once we have located the most simlar neighbors for a test instance, the next task is to devise a predicted response based on those neighbors by allowing each neighbor to vote for their class atribute and take the majority vote as the prediction. After that the only thing left to do is to evaluate the accuracy of the predictions and an easy way to do this is to calculate a ratio of the total correct predictions out of all predictions made, called the `classification` accuracy. The `accuracy` results for each dataset can be seen in the figures down below.

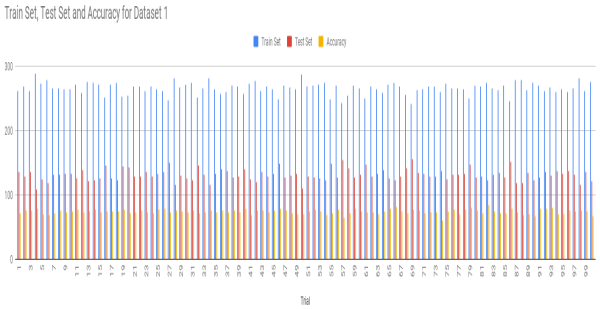

Fig. 32: Train Set, Test Set and Accuracy for Dataset 1

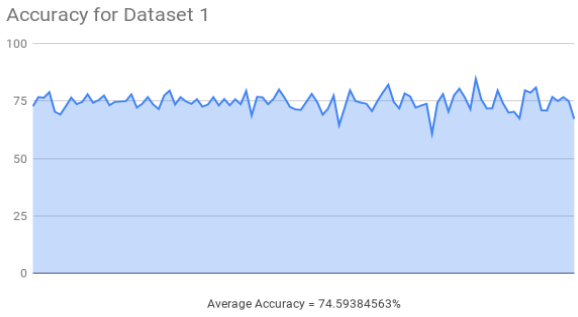

Fig. 33: Training Accuracy for Dataset 1

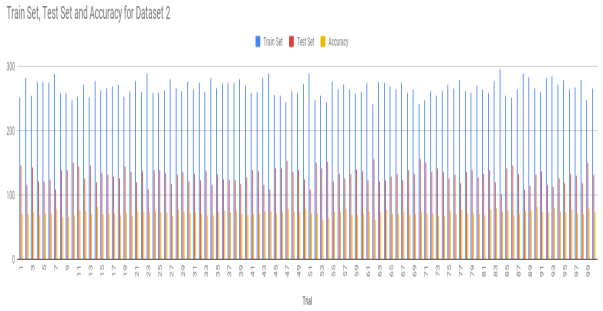

Fig. 34: Train Set, Test Set and Accuracy for Dataset 2

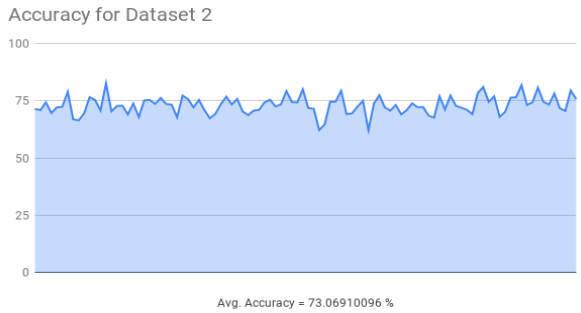

Fig. 35: Training Accuracy for Dataset 2

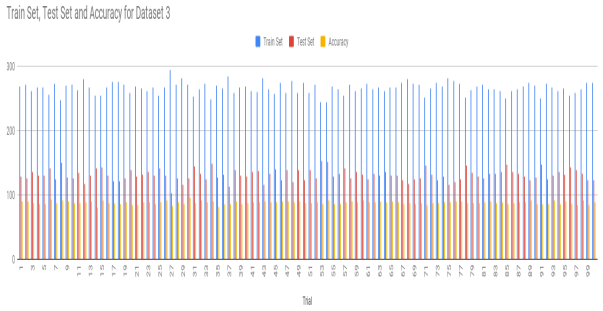

Fig. 36: Train Set, Test Set and Accuracy for Dataset 3

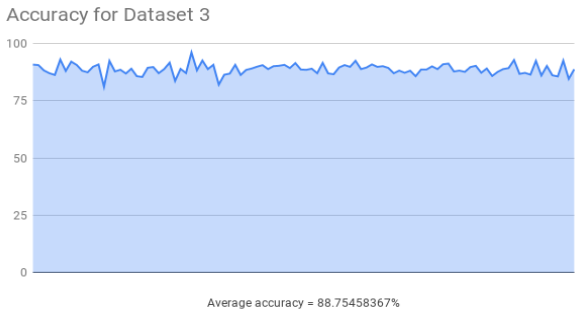

Fig. 37: Training Accuracy for Dataset 3

5 Conclusions

Throughout the development of this project, it was possible to learn how a multitude of techniques work and match them together in a greater scale than merely lessons and tutorials can provide, with a greater motivation backing it up. It was all planned with a target use in mind, but the features developed can be used in other contexts. Having this said, the goals set for this project were achieved. The algorithm can correctly create the correlation maps between the images in order to construct the "`interest zone`" that can be used to look for the object. There were of course some drawbacks to this procedure, the main cause being the image quality of some of the frames of the datasets used. It is also of course important to go over the whole social aspect of working in a team, even if only a two-man one, soft-skills become imperative, cooperation and understanding is crucial and something to always be worked on granted the opportunity, which this project has.

Appendix A
Acknowledgments

The authors of this report would like to thank the professor Petia Georgieva for her time, patience, and insight.

References

[1] "Template Matching Algorithm" [Online].
Available: https://www.docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html
https://www.imageeprocessing.com/2011/06/template-matching-in-matlab.html

[2] "ATLAS Project" [Online].
Available: http://lars.mec.ua.pt
http://lars.mec.ua.pt//atlascar_live.html

[3] "Machine Learning Algorithms" [Online].
Available:https://www.sas.com/id_id/insights/articles/analytics/machine-learning-for-beginners-and-beyond.html?utm_source=bing&utm_medium=cpc&utm_campaign=ai-ml-us&utm_content=GMS-101514&keyword=machine+learning&matchtype=be&publisher=bing&msclkid=2a4501f5ba9c1d2082555d491208c9f8&utm_source=bing&utm_medium=cpc&utm_campaign=AI+%26+Machine+Learning&utm_term=machine+learning&utm_content=AI+%26+Machine+Learning_Machine+Learning

[4] "How to compare machine learning algorithms" [Online.]
Available:https://machinelearningmastery.com/compare-machine-learning-algorithms-python-scikit-learn/

[5] "How to work with K-Means Algorithm" [Online.]
Available:https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
https://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/