

Machine Learning Project

Object Detection for ATLASCAR Project

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 "zone of interest" 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 ??, the

core of the tool is exposed, discussing some of the problems found and solutions developed in order to obtain the results showcased. Afterwards, in section 3, 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 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 "zone of interest" 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 bounding 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 and visualise and determine the "zone of interest" of a given object in the scene. Not only that this 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 were developed based on other datasets provided by other people that have worked in the ATLAS-CAR project.



Fig. 1: ATLAS-CAR2

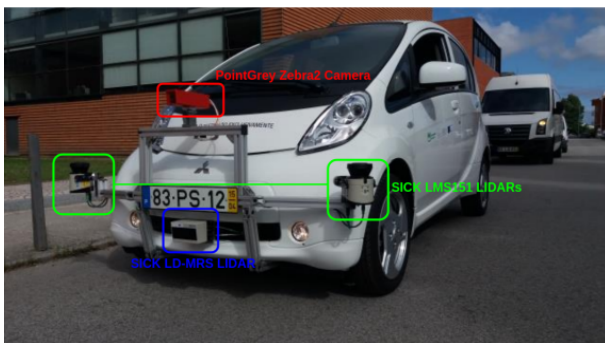


Fig. 2: ATLAS-CAR2 Sensors

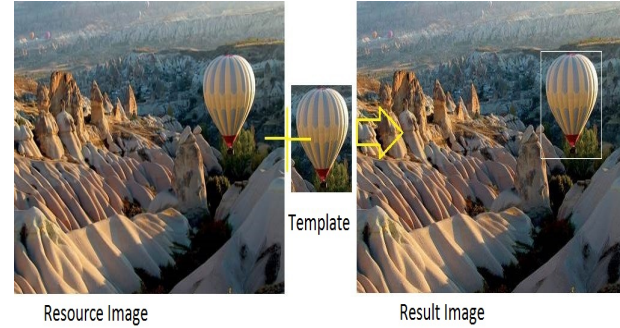


Fig. 3: Template Matching Example

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 ATLAS-CAR 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 ATLAS-CAR system. From these experiments, 3 rosbag files were created, one from each experiment with the ATLAS-CAR.

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

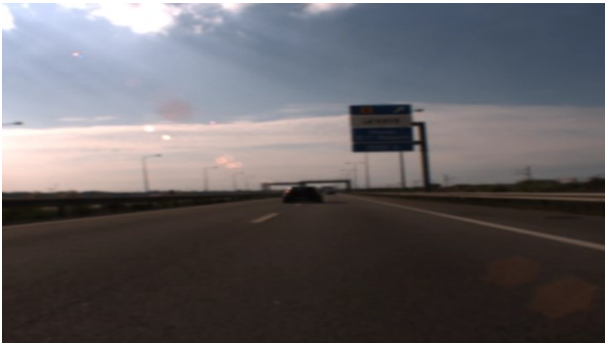


Fig. 6: Frame example from third dataset

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:

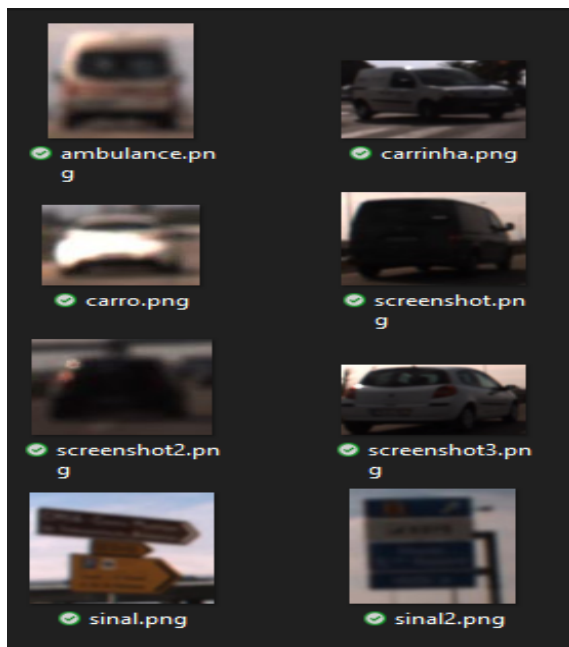


Fig. 7: Templates used in Template Matching

4 MACHINE LEARNING

In order to determine the "zone of interest" 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 the most suitable in this context.

4.1 Knn-Means Algorithm

The k-nearest neighbors algorithm is a Unsupervised Machine Learning algorithm which is used where 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 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 algorithm 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.

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.

```

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

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

```

Fig. 8: Knn-Means Algorithm

First we load the frames from the dataset and extract each RGB dimensions 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 $m \times 3$ 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 reorder 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 Lecture 07 of the Machine Learning course, for set number of iterations. For the purposes 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 process 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.

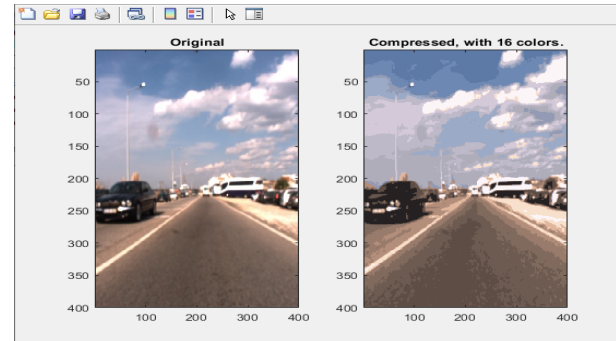


Fig. 9: Frame from first dataset compressed with 16 colors

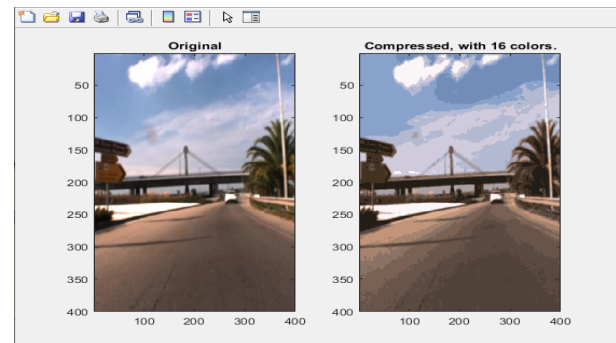


Fig. 10: Frame from second dataset compressed with 16 colors

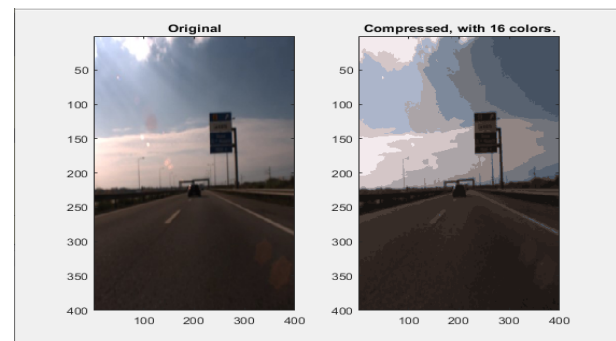


Fig. 11: Frame from third dataset compressed with 16 colors

After the compression process, we can proceed to the Template Matching process 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 through 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 12.

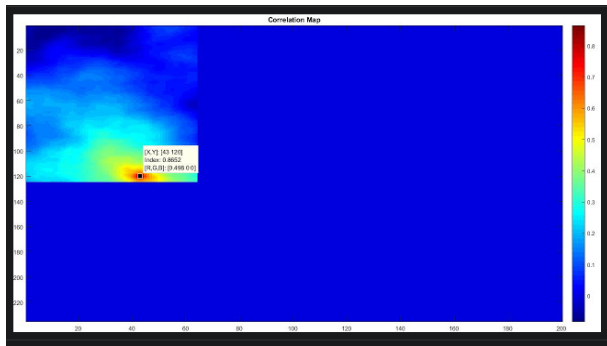


Fig. 12: Correlation Map Example

This correlation map defines the "zone of interest" 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 13 and in figure 14.

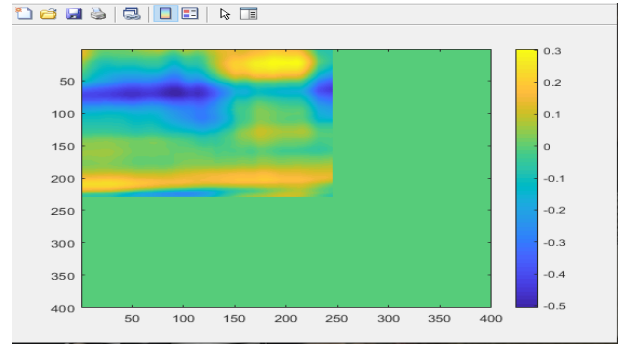


Fig. 13: Correlation Map between target and template images

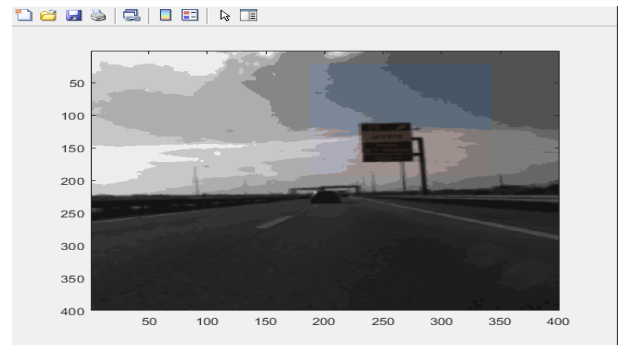


Fig. 14: Template Matching Spatial Domain Result

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

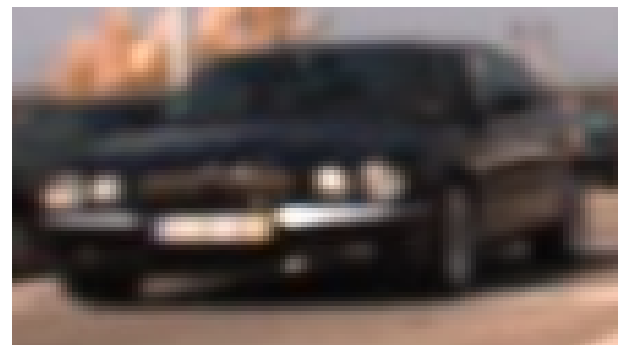


Fig. 15: Template Object used for the First Dataset

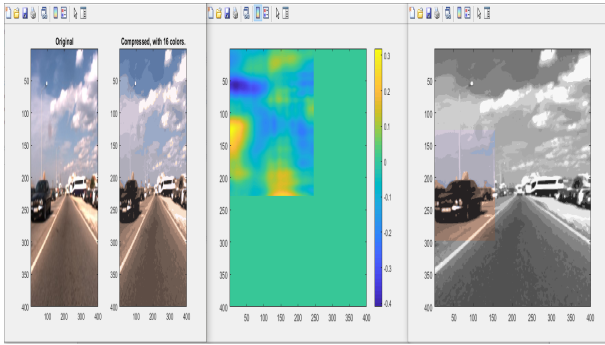


Fig. 16: Template Matching with Correlation Map of a Frame Image from the First Dataset



Fig. 19: Template Object used for the First Dataset



Fig. 17: Template Object used for the Second Dataset

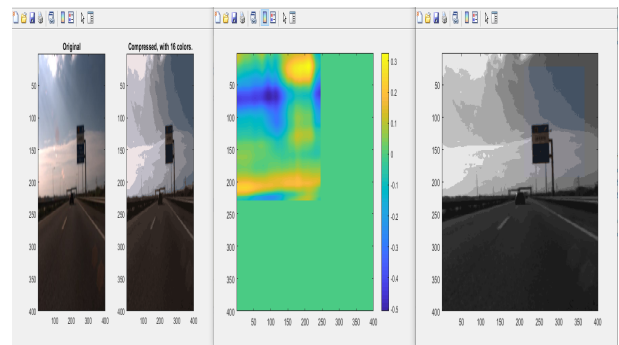


Fig. 20: Template Matching with Correlation Map of a Frame Image from the Third Dataset

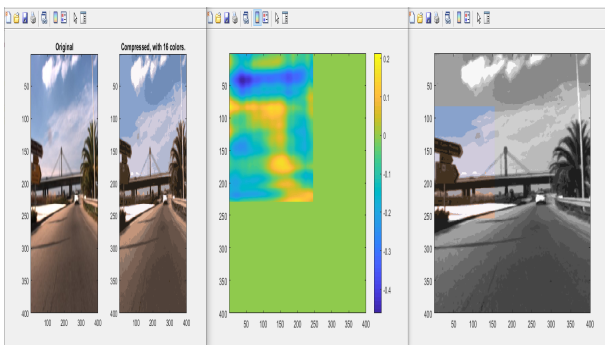


Fig. 18: Template Matching with Correlation Map of a Frame Image from the Second Dataset

4.3 Other Results

4.4 Training Model

4.5 Final Results

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 the project were achieved.

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] Docs.opencv.org, "Camera Calibration and 3D Reconstruction" [Online].
Available: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html
- [2] Docs.opencv.org, "Image Inpainting" [Online].
Available: <https://docs.opencv.org/2.4/modules/photo/doc/inpainting.html>
<https://www.nvidia.com/research/inpainting/>
<https://en.wikipedia.org/wiki/Inpainting>
- [3] Docs.opencv.org, "Image Inpainting Masks" [Online].
Available: https://docs.opencv.org/3.4/df/d3d/tutorial_py_inpainting.html
https://www.researchgate.net/post/How_to_design_mask_of_an_image_for_inpaintings2
<http://answers.opencv.org/question/205615/how-to-get-a-mask-of-an-image-so-that-i-can-use-it-in-the-inpainting>
https://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_Image_reconstruction_Inpainting_Interpolation.php
- [4] Docs.opencv.org, "Simple Image Inpainting" [Online].
Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_photo/py_inpainting/py_inpainting.html
<https://stackoverflow.com/questions/10340537/image-inpainting-cvrectangle>
<https://stackoverflow.com/questions/53838667/how-to-get-a-mask-of-an-image-so-that-i-can-use-it-in-the-inpainting>
<https://dsp.stackexchange.com/questions/36222/what-is-the-difference-between-image-inpainting-and-image-denoising>
- [5] Docs.opencv.org, "PCD (Point Cloud Data) 3D Format Images" [Online].
Available: http://pointclouds.org/documentation/tutorials/pcd_file_format.php
<https://stackoverflow.com/questions/47103273/pcd-file-cannot-view-color-information-correctly>
<https://abhishhek4273.com/2014/07/01/opencv-and-pclfiltering-a-pcd-file/>
https://stackoverflow.com/questions/45783815/view-multiple-point-clouds-in-same-window-using-pcl-in-c?fbclid=IwAR1hIPdh1MTeLM9bDk2URmgiXhdF7zSn14SYnW9_35P-CzaJGJfbTKaFfH8
<https://stackoverflow.com/questions/3796452/how-do-i-display-draw-a-ply-object-in-opengl>
<https://stackoverflow.com/questions/30764222/how-to-read-ply-file-using-pcl>
<https://stackoverflow.com/questions/51350493/conversion-of-ply-format-to-pcd-format>
http://docs.pointclouds.org/trunk/structpcl_1_1_r_g_b.html
- [6] Docs.opencv.org, "ICP Alignment with Point Clouds" [Online].
Available: http://docs.pointclouds.org/1.0.1/classpcl_1_1_iterative_closest_point.html#_details
https://docs.opencv.org/ref/master/d9/d25/group__surface__matching.html
https://www.mrpt.org/Iterative_Closest_Point_%28ICP%29_and_other_matching_algorithms