Instituto Superior Técnico

Mestrado Integrado em Engenharia Mecânica

Computational Vision

Work number 1 – Problem 16

Afonso Valador, 87142

José Trigueiro, 87225

Professors: José Azinheira and Alexandra Moutinho

2020 – 2021

20th of November 2020

TÉCNICO LISBOA

Contents

1. Introduction 3
2. Analysis of the dataset 3
3. Thresholding algorithm 4
   3.1. Colour space selection and threshold parameters 4
   3.2. Post processing: mathematical morphology and region analysis 6
   3.3. Validation, testing and discussion of the results 8
4. Segmentation with k-means Clustering algorithm 9
   4.1. Initial clustering 9
   4.2. Analysis of the blue cones cluster 12
   4.2.1. Cone detector 13
   4.3. Analysis of the yellow cones cluster 14
   4.4. Validation, testing and discussion of the results 16
5. Post processing with Hough Transform 18
   5.1. Edge detection 18
6. Optical Flow 20
7. SURF Method 21
8. Program Description and Tutorial 21
   8.1. Option B – Segmentation based on HSV threshold only 22
   8.2. Option C – Segmentation by HSV threshold and filtering by mathematical morphology 23
   8.3. Option D – Segmentation by threshold on HSV and Hough transform 23
   8.4. Option E – Segmentation by threshold on HSV, mathematical morphology and Hough transform 24
   8.5. Option G – K-means segmentation method 25
   8.6. Option H – Matching SURF features 26
9. Conclusion 26
10. References 27

1. Introduction

The proposed work consists in the identification of coloured traffic cones in a Formula Student competition racetrack, in order to give information to the car's autonomous driving algorithm about the route it should take, as well as warning it when it is veering off-course.

Still images from the work handout were supplemented with videos available on YouTube from various Formula Student teams to test the developed solutions.

For the segmentation of the image, both thresholding in the HSV colour space and an algorithm based on super pixel clustering were used. A post processing technique based on the Hough Transform of straight lines, taking advantage of the triangular shape of the cones was also developed.

2. Analysis of the dataset

From the images in the dataset and particularly the competition videos [2], [3] and [4], it was concluded that the course was always defined in the same way: blue cones on the inside of the course (left side) and yellow cones on the outside (right side) as seen in Figure 1.

Figure 1- Sample image of the dataset, with blue and yellow cones shown

This means that the first step of the problem will be developing a procedure that reliably identifies blue and yellow objects. This approach presents some challenges, since the choice of colour for the cones will potentially generate some false positives, since the sky may be included in a thresholding algorithm that detects blue and dry grass or other background objects may be detected in one that detects yellow. Another possible difficulty may be separating any yellow or blue features in the car or the track from cones automatically.

3. Thresholding algorithm

3.1. Colour space selection and threshold parameters

By default, MATLAB imports images from files in RGB format. This colour space was shown to be inadequate for this problem, since information about the colour of the images is present in all 3 dimensions of this space, making thresholding difficult. Using only a grayscale image also was considered unreliable, since information about colour, the most defining characteristic of the cones, is lost in the conversion.

This led to the approach of using the HSV colour space. This colour space has three dimensions: Hue, measuring the dominant wavelength in the perceived mixture of light waves, Saturation, measuring the purity of the colour (amount of white light mixed in the colour) and Value, which is similar to intensity and describes how dark the colour is. As shown in Figure 2, the Hue will be around 1/6 for yellow and 2/3 for blue. These values are coherent with MATLAB, since it measures the HSV colour channel values in a scale from 0 to 1, so they can be used in a first trial as thresholding candidates.

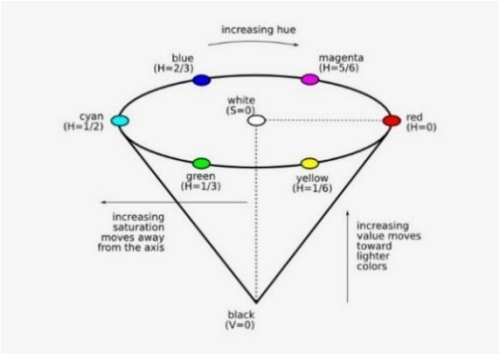

Figure 2 – HSV colour space model

Dividing the sample images in their HSV components, an example of the result obtained is shown in Figure 3.

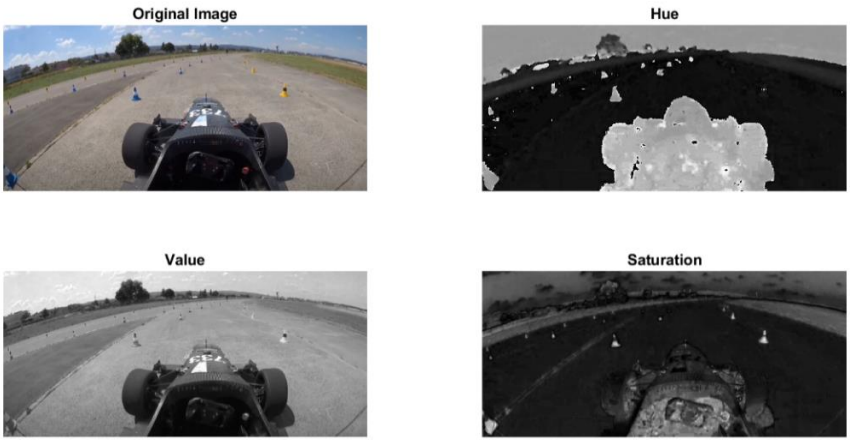

Figure 3 - HSV decomposition of a sample image

Some features in Figure 3 help in developing a threshold that will separate the cones:

- Since the cones are monochromatic painted in pure, bright colours, their saturation is higher than most objects in the image background. However, some elements of the car also have high saturation.
- Value differs in yellow and white cones: in yellow it is higher than most objects other than the sky and white paint, while in blue it has a comparable value to the background, meaning removing pixels with low value will not yield satisfactory results for blue identification.
- As expected, yellow takes a low value in the hue plot, meaning it cannot be easily identified even by human eye. However, since blue has a high value it is identified in the image, along with the whole visible sky and some background objects.

This analysis gives some insight into the parameters needed for testing. First an auto threshold using Otsu's method did not yield an adequate result for any of the colour channels. The best results were obtained after auto-thresholding in the hue channel, shown in Figure 4. This method however kept too much noise in the image, so a manual threshold was used instead.

Figure 4 – Result of Otsu's method auto-threshold in the hue channel

Using an educated guess from interpretation of the sample images and the colour space combined with some fine adjustment, it was decided that taking thresholds on all channels of the colour space would be necessary to get an accurate identification. In the case of the hue and value, a maximum and minimum were considered as limits for thresholding since a colour is identified by an interval and not only by a simple threshold. The values were used for the colours are shown in Table 1.

Colour	Hue Threshold		Saturation Threshold (minimum)	Value Threshold	
	Minimum	Maximum		Minimum	Maximum
Yellow	0.08	0.17	0.6	0.1	1
Blue	0.52	0.72	0.6	0.1	0.6

Table 1 – Thresholds for yellow and blue

While this process detects some cones and their colour, it also detects some noise which will need to be removed if it is to be used in practice in an autonomous vehicle, since false positives may generate a false model of the route. Using the previous sample image as an example, a segmentation of the image by extraction of connected components (*bwlabel* function) was performed and is shown in Figure 5. Note that the two closest yellow cones are detected as expected, but 16 other blobs are present, ranging from image noise that was picked up in the thresholding to unconnected parts of the cone that are counted as different blobs.

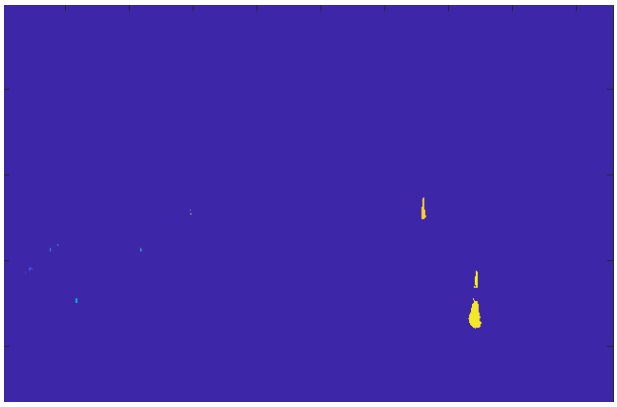

Figure 5 – Connected component labelling of a sample image

In the next chapter some techniques are shown that minimize false positives and allow for a better cone identification algorithm.

3.2. Post processing: mathematical morphology and region analysis

To get a correct and unique detection of the cones in an image, first it is important to remove isolated noise pixels that are not eliminated through the thresholding process. A way of dealing with this problem is by applying mathematical morphology, more specifically an **opening**, a process consisting of an erosion, a process that will shrink (in the case of features larger than the uses structuring element – STREL) or remove (in the case of smaller features than the STREL) followed by a dilation, that will approximately restore to their previous size and shape features that were not eliminated by the erosion. An adequate balance between noise removal and preservation of information about identified cones was found by using a square STREL with a width of 3 pixels.

Since the cones are convex shapes, any holes in their corresponding blob are a result of noise or removal by the thresholding process. This means that a **closing** process is useful as well since it fills any missing pixels inside the cone blob and smoothens its border. The result of the segmentation from Figure 5 after opening and closing is shown in Figure 6.

Figure 6 – Segmented image after opening and closing

Note that, while all the noise was removed from the image (all blobs correspond to detected cones), the process will still yield four blobs while only three yellow cones are visible in the image and close enough to be detected by the algorithm. This is caused by the upper and lower part of the cone being separated by a stripe, which is not captured by the thresholding, meaning upper and lower part of the cone are detected as different objects.

This problem calls for the use of an additional tool: region analysis of the image using the *regionprops* function that will calculate the properties of the different regions of a segmented region. In this case, two properties are necessary: the centroid coordinates and the area. It was decided that to better approximate the position of the cone, the smallest (by area) regions belonging to the same cone would be eliminated. To find blobs belonging to the same cone, a minimum distance in pixels was defined under which blobs were considered related. The Euclidean distance was used, with a minimum value of 50 pixels being set. This finally yields a segmentation that is appropriate to use in a real image to detect cones, shown in Figure 7.

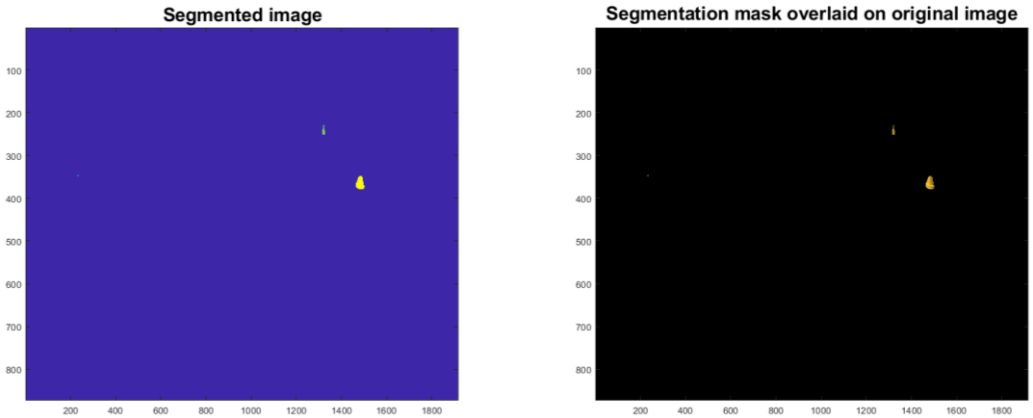

Figure 7 - Result of segmentation and mask overlaid on original image

3.3. Validation, testing and discussion of the results

Images were fed into the previous algorithm for validation of its performance and testing of accuracy.

For the image used in previous explanations, the result is shown in Figure 8. The nearest cones were detected and only cones far away were missed by the algorithm. No false positives were detected. The colour of the box indicates the colour of the detected cone.

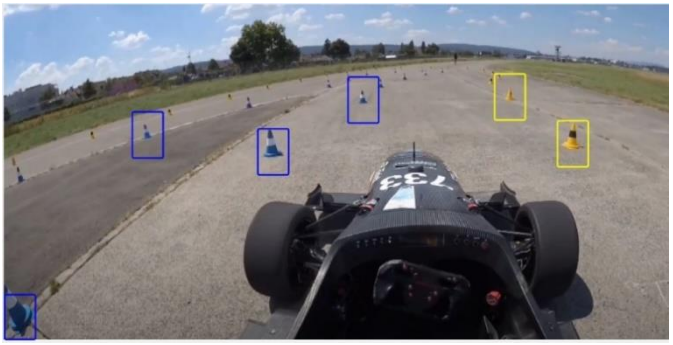

Figure 8 – Results of cone identification in example image

The algorithm performs well for most other images where other blue or yellow objects are not in the frame, even though it can only identify cones when they are close. This performance would be better if the videos were not run through YouTube's compression algorithms and if they were in a higher resolution (they were only published in a split-screen view with other information, lowering the resolution of the image). Examples of correct identification are shown in Figure 9.

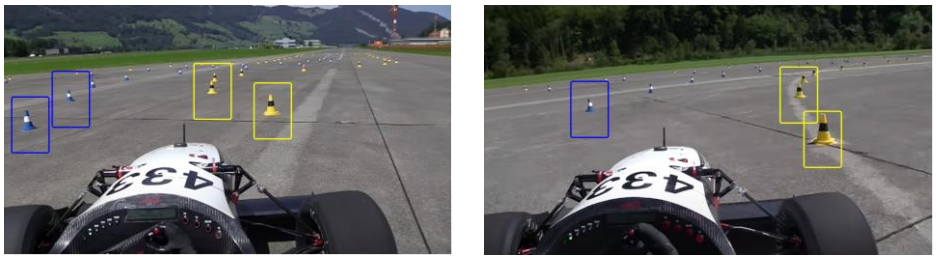

Figure 9 - Cone detection in other sample images

However, like any other computer vision application, the success of this method is context dependent, especially affected by lighting conditions, camera and video storage quality, objects surrounding the racetrack and the design of the car itself. An example of failure is shown in Figure 10. On the top left, a background object has a similar colour to the cones,' so it is mistakenly identified as one. Features in the car are also coloured blue and yellow, so they are also detected. Finally, the brightness of the sunlight means a yellow cone that should be in range of detection is missed.

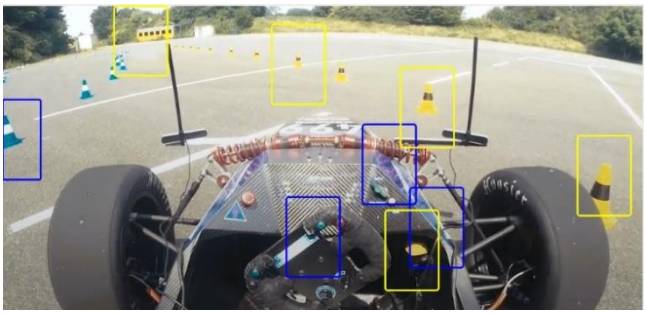

Figure 10 – Sample image showing multiple failure points of the algorithm

4. Segmentation with K-means Clustering algorithm

This method has some similarities with the previously shown algorithm. It is a colour-based segmentation method that not only relies on the Lab colour space (to perform the clustering using k-means), but also in the HSV colour space, to optimize the final image and make sure that only the cones are highlighted.

4.1. Initial clustering

In a first approach to this semi-automated algorithm [5], the whole image was considered for clustering. This means that, for example, all the details in the car structure will be consider in the segmentation, leading to major problems. Since the camera is fixed in this structure, it is possible to simply crop out the geometry of the car and analyse the remaining image, as seen in Figure 11.

Figure 11 - Geometry of the car about to be cropped out

After this operation[1], the image is ready to be subjected to clustering. To process this action, first the colour space should be converted to Lab. By taking out the channels “a” and “b” from this colour space it is possible to, finally, apply the segmentation through K-means to the image subject to analysis.

[1] Note that is extremely important to only use the images provided or equivalent, because if the car structure, the camera position or even the image resolution changes, the crop does not result as intended.

After a couple of trials with the number of clusters that is needed for the segmentation, 6 turned out to produce the best results, on average, i.e. separating all the main components, but not excessively, which could led to further difficulties.

The result of this command is a matrix with only one channel and where the values of each pixel corresponds to the cluster to which that pixel belongs. Finally, a grayscale image is shown in Figure 12 for a rough visualization of each cluster.

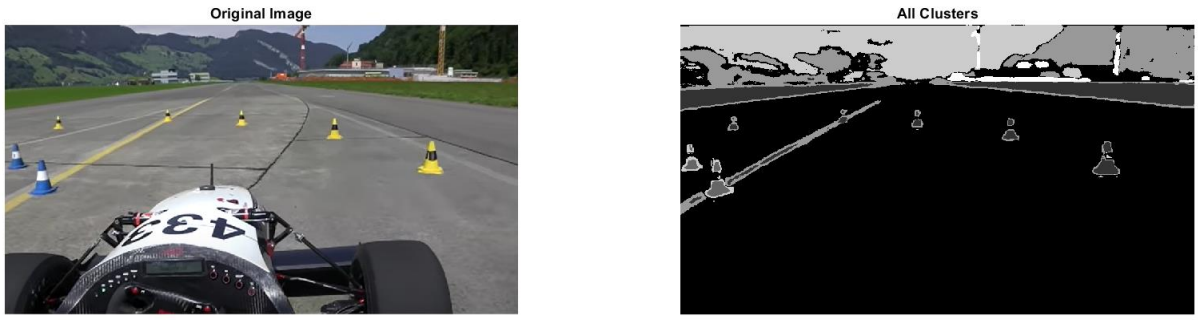

Figure 12 - Sample image (left); All the 6 clusters in a grayscale colour space (right)

To better see all of this clusters in theirs respective colors, a simple colour separation algorithm was carried through all the clusters, which lead to the following results:

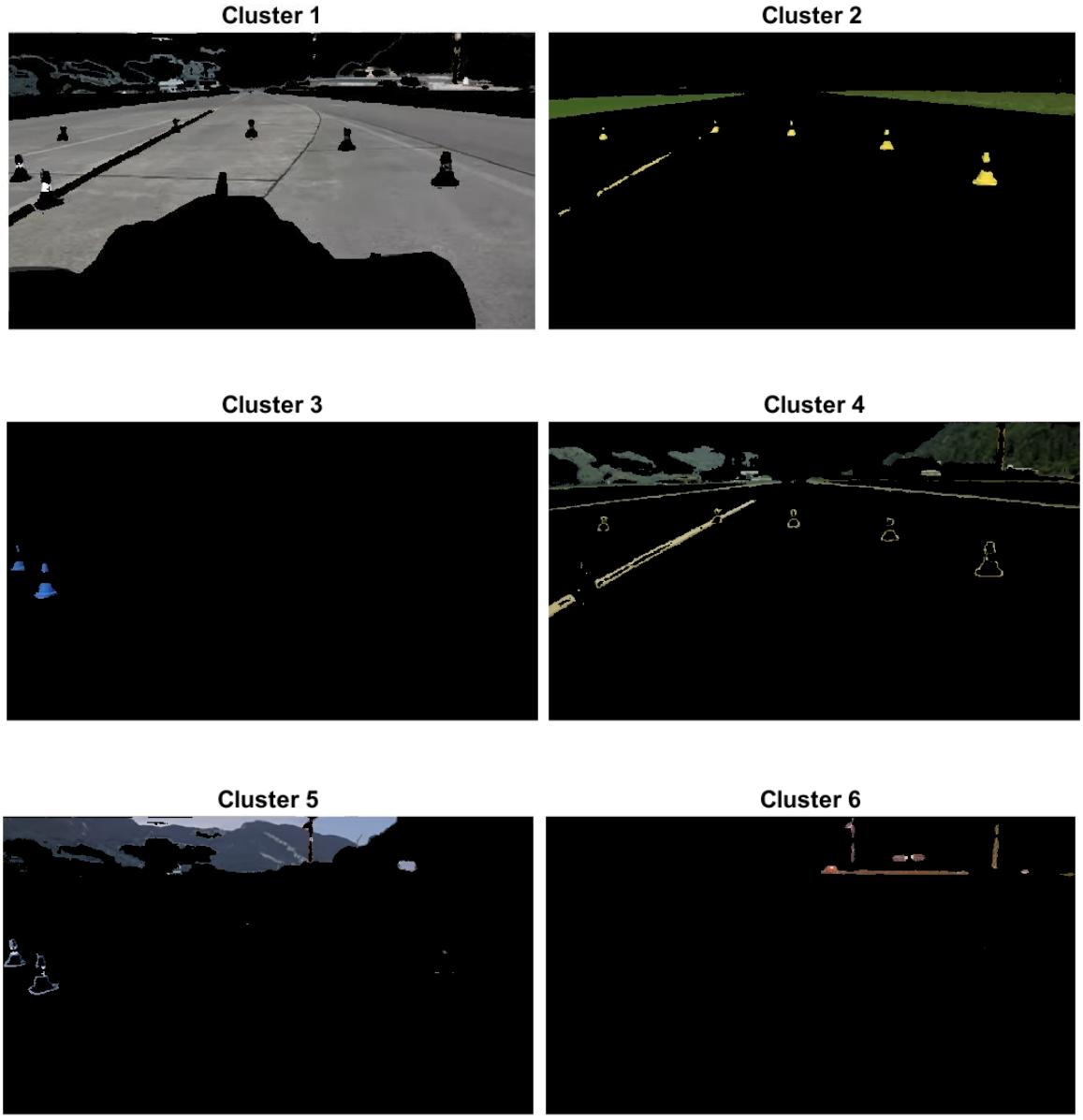

Figure 13 All the cluster of the sample image separated, in RGB colour space

A quick look at the figure 13 makes it possible to notice that:

- The cluster 1 has the structure of the car cropped out, as it was intended. Also, this cluster always has the lowest number of black pixels, since the road occupies most of the image.
- Visually it very clear that the clusters with the cones are the clusters number 2 and 3, yet the algorithm needs to automatically detect those indexes, so a procedure to solve this problem needs to be implemented.
- Cluster number 4 and number 5 seem to share the same colours of clusters 2 and 3, respectively, which can lead to additional issues.

- Cluster number 2 not only contains more features than the yellow cones, but also has colours other than yellow, since a significant fraction of it corresponds to the green areas of the original image.

The above points are going to be very important in the analysis carried forward in the next sub-sections.

4.2. Analysis of the blue cones cluster

The first part of this approach was identical in both cases and, at the same time, approximately what was descripted in chapter 3. Again, using an educated guess, along with some adjustments, a threshold was chosen for all the 3 channels of the HSV colour space sample images. Since the sample images used in this chapter were different, it was necessary to find different values for these thresholds, although they shouldn't change too much from the previously found ones. Table 2 shows the referred values.

Colour	Hue Threshold		Saturation Threshold (minimum)	Value Threshold (minimum)
	Minimum	Maximum		
Blue	0,57	0,70	0,4	0,1

Table 2 - Threshold for blue

After finding these values, a mask can be created in all the 6 clusters to filter out the values of Hue, Saturation and Colour that don't belong in the defined range. Then, it is possible to convert this image into a black and white one and count all the white pixels (with value 1, that correspond the blue pixels inside the defined threshold range in table 2) of all clusters.

Figure 14 – Black and White images of the blue clusters (3 – left and 5 – right)

Figure 14 shows the two clusters with the greatest number of white pixels. In this specific case, it is visually clear that the image from the left represents the blue cone. However, two questions arise from this method: (1) "How to detect if there is any blue cone just by the number of white pixels?"; and (2) "Is it possible to distinguish the two images above just by the number of pixels?". The answer, in this specific case, is yes, but it would be an error if one

decided to detect what image has the blue cones int this way. If the sky on the cluster 5 changes it's HSV components just a little, that change can compromise one's conclusion about the number of blue pixels of interest.

At this point there is a need to answer the described questions to conclude about the existence of the blue cones, and, if any, in what cluster they appear.

4.2.1. Cone detector

To answer the first question a simple procedure is made. First, the cluster with the road (in this case, cluster number 1), is discarded from our list of options that can be the cluster with the blue cones. It was observed that this cluster, in some sample images from the dataset, had a huge number of residual blue pixels in the range of the threshold defined. To prevent the algorithm from wrongly choosing this cluster, it counts all the black pixels, in all clusters and the assigns this cluster to the minimum number of black pixels. Thus, since for all the cases this cluster has always the road present, it's feasible to conclude that it won't be included in our options anymore.

Finally, at most two relevant clusters remain that should be subject to this detection (in this case, once again, cluster 3 and 5). The only difference between them is, in general, the sky. To solve this problem only 3/5 of the height was considered to the pixel counting, as shown in figure 15.

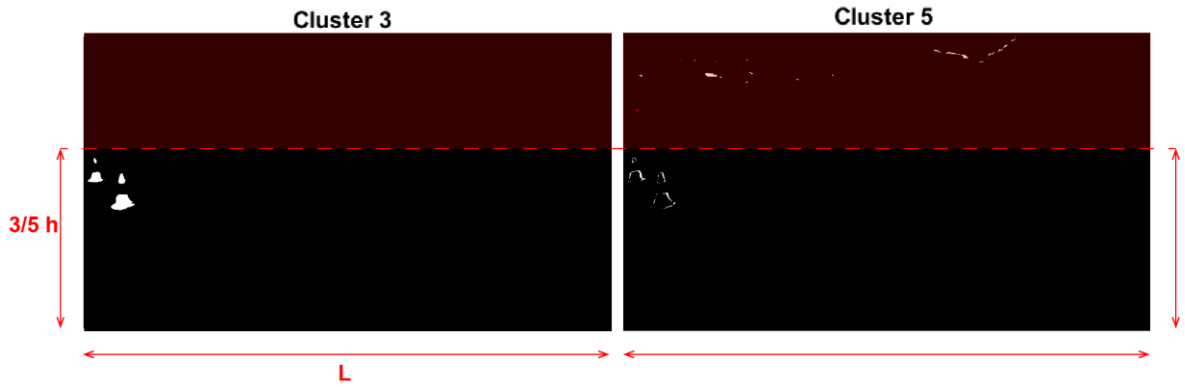

Figure 15 - Image to be cropped from the detection (in red) from cluster 3 (left) and 5 (right)

If doubts remained, this procedure makes it clear that cluster 3 has more white pixels and, therefore, the greatest number of blue pixels in the selected HSV range.

The main goal can now be achieved: selecting the cluster with maximum number of white (blue) pixels gives us the index of the cluster with the blue cones. To detect if there's any cone in that region, a tolerance of 50 pixels between features was imposed, with features closer than that being eliminated.

4.3. Analysis of the yellow cones cluster

An identical and simpler procedure was carried through the analysis of the yellow cluster. A first glance at figure 16, may wrongly lead to the conclusion that segmenting this image even further will be difficult, since it has mainly 3 parts: (1) the green areas, (2) the yellow line on left side and (3) the yellow cones.

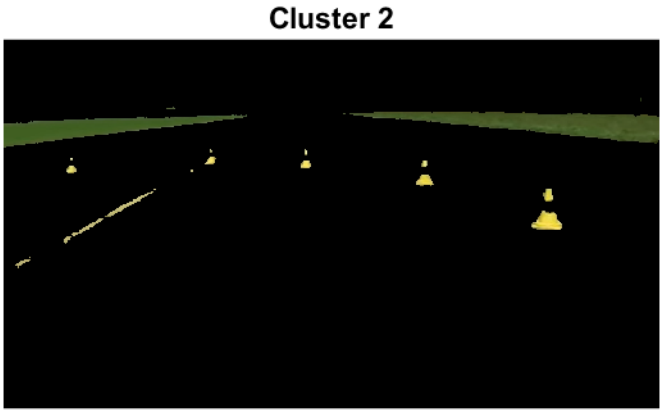

Figure 16 - Cluster 2 of the sample image

The remaining part of this analysis is nearly the one descripted in 4.2. A manual threshold for the yellow colour of the pins was chosen to filter out all the remaining colours. At first, some obstacles were found with these procedures, because the values chosen didn't seem to separate the colours correctly. However, with an exhaustive search, the values in Table 3 turned out to produce adequate results, across the multiple sample images.

Colour	Hue Threshold		Saturation Threshold (minimum)	Value Threshold (minimum)
	Minimum	Maximum		
Yellow	0,11	0,16	0,56	0,32

Table 3 - Threshold for yellow

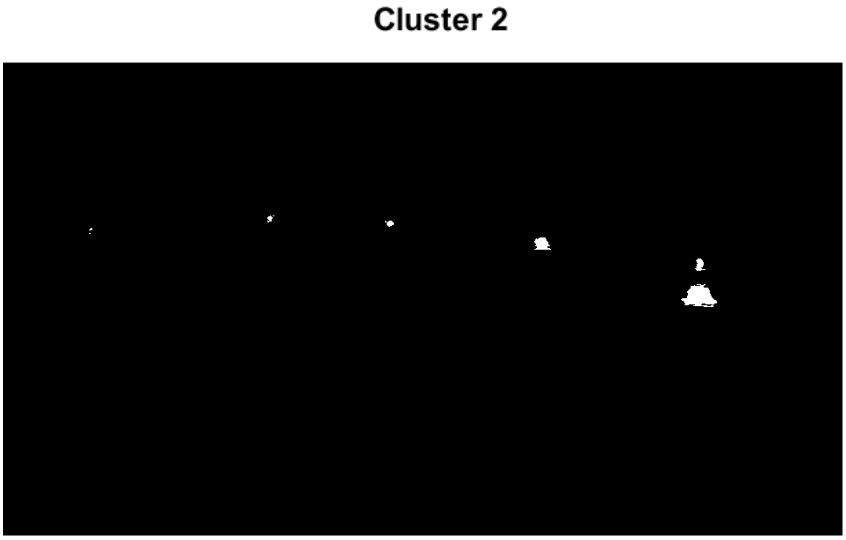

Figure 17 - Black and White image of the yellow cluster (2)

Once again, the main goal can now be achieved: selecting the cluster with maximum number of white (yellow) pixels gives us the index of the cluster with the yellow cones. To detect if there's any cone in that region, a tolerance of 100[2] pixels was imposed, eliminating blobs under that distance. This was proven to be sufficient to find the most relevant regions of interest. The result, in a black background is illustrated in Figure 18.

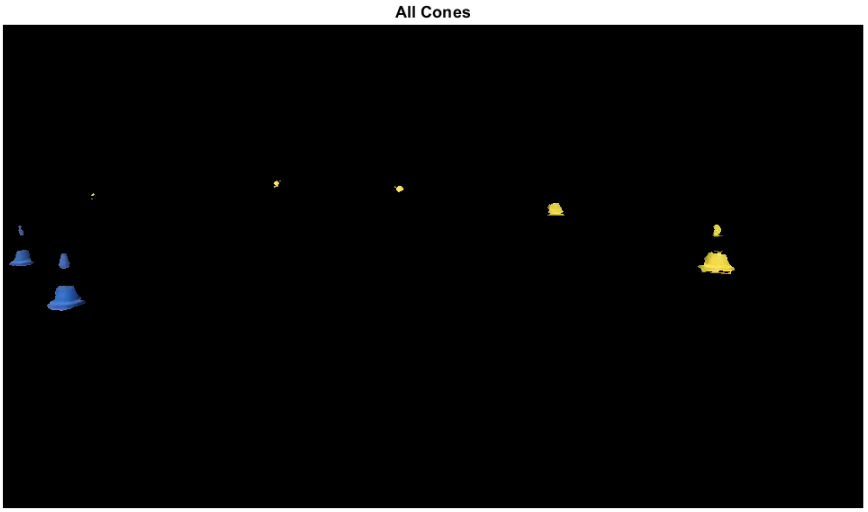

Figure 18 - Blue and Yellow cones highlighted

[2] This value differs from the one found in the blue pixels because after analysing multiple images, the value for the number of yellow pixels trespassed the tolerance imposed for the blue pixels (50) but was always lower than 100.

4.4. Validation, testing and discussion of the results

With the procedure completed, the algorithm was tested in the database given, verifying the robustness of this method. The image that was used as an example in this chapter is shown with the cones identified in figure 19.

It is now time to test the database given and conclude about the robustness of this method. The image that was carried through this chapter is now shown with cones identified in Figure 19.

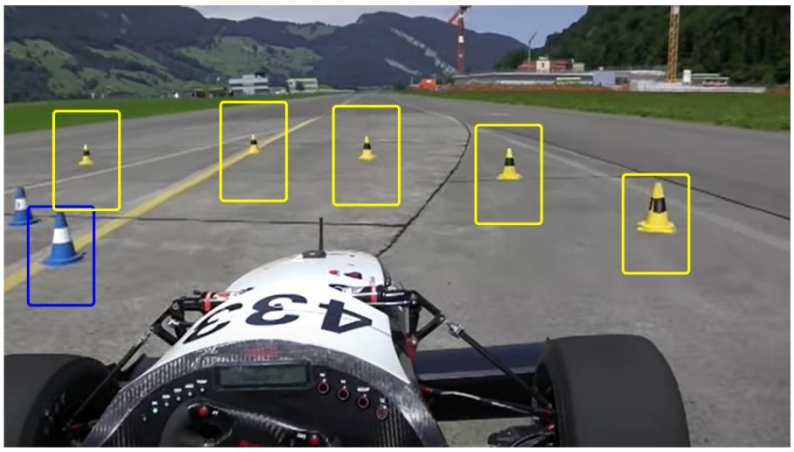

Figure 19 - Results of cone identification in the example image

Although all the yellow cones were successfully identified, the left blue cone in the border was not detected, even though it clearly appeared in figure 18. If a comparison between these two figures (18 and 19) was made, one can conclude that the blue cone that failed to be identified had a detected area much larger that most of the yellow cones. Yet it failed to be identified, probably because it was too close to the other blue cone.

Extending this result to the other sample images given in the dataset, the results in Figure 20 were obtained.

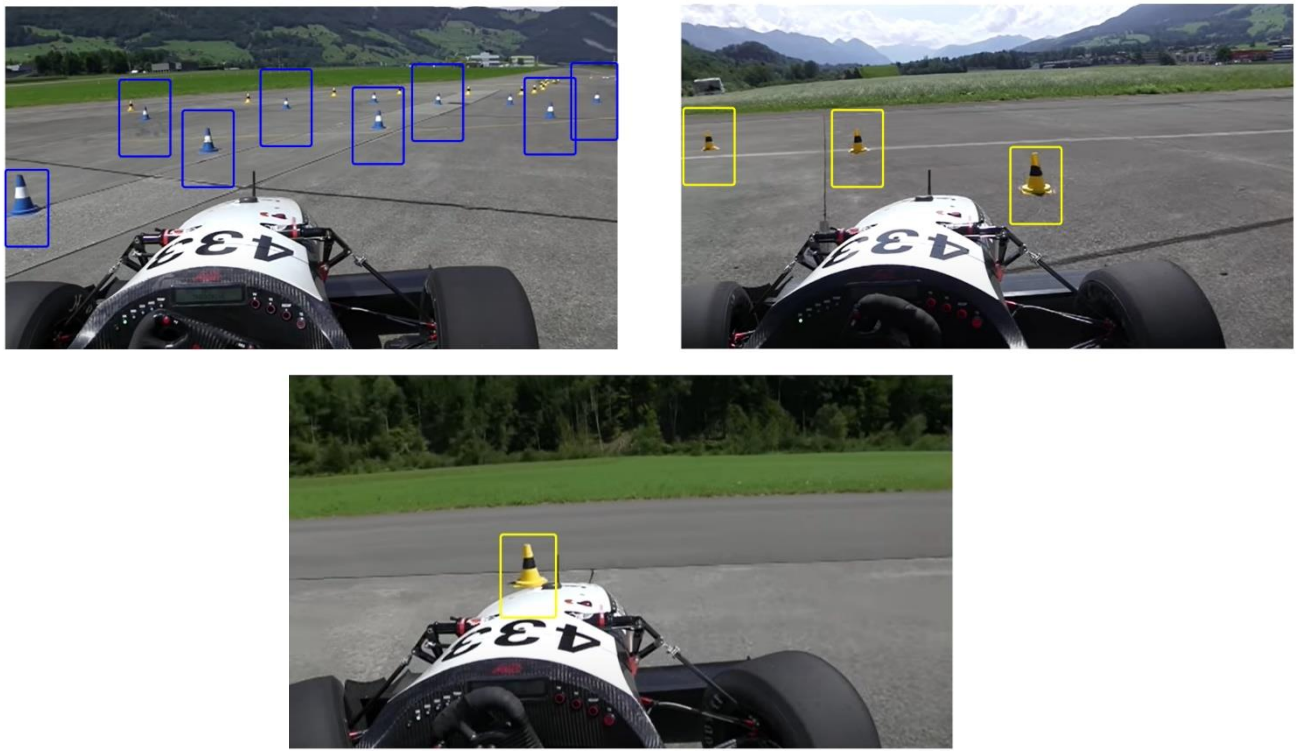

Figure 20 - Cone detection in other sample images

In the top left sample image, almost all the blue cones were detected, even the ones that were at a considerable distance from the camera. Thus, the range up to which this method can identify cones seem higher than in the previously discussed methods. Besides this, when the cones are from a single colour, blue or yellow, the detection appears to be working as intended in new images.

These results are very accurate, but, as was said in chapter 3, they are extremely context dependent. The lighting and weather conditions seemed to make a great impact in the performance of this method. One of the advantages is the fact that the region of the car was completely taken out from the analysis, which is equivalent to say that the car wasn't in the frame, so none of its features will be mistakenly detected as a cone. This procedure is easy to perform and could be implemented in the previous algorithms to achieve better results.

However, this method can be time consuming and difficult to implement in real life. Its calibration to light and weather conditions, should be as precise as possible and is done manually. Besides this, the resolution of the image must be specific, otherwise it is necessary to change the crop region of the car. Although this is an easy procedure when considering the calibration before a competition, it could prevent the application to new sample images, from

other cars or cameras. To have higher versatility in the crop region of the car, an improvement that could be achieved is defining the crop region with a more general shape image 11, at the cost of a smaller region for cone detection.

5. Post processing with Hough Transform

The Hough transform is a transformation of coordinates from the image space to a Hough space characterized by a parametrization of a curve. A cone is approximately the shape of a triangle, so by search in the region of interest around a blue or yellow blob for two lines that are both long enough and form an acute angle between them, it is possible to detect if a blob corresponds to a triangular object, which would be a strong candidate for a detected cone. This means that the Standard Hough Transform, mapping the points to the (ρ, θ) space, given by the polar parametrization of a straight line may be used as shown in Figure 21.

$$\rho = xcos(\theta) + ysin(\theta)$$

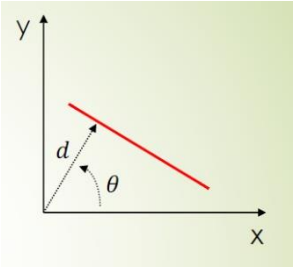

Figure 21 – Parametrization used in the Hough Transform

In this case, it was chosen to analyse the Hough transform from -35º to 35º, avoiding this way any strong horizontal lines like the horizon or lines in the road, and to analyse the Saturation channel of the image, since it was shown in Figure 3 that cones usually have higher Saturation than the surroundings.

The transform works as a voting scheme, so points belonging to a certain line will contribute to increase the values associated with the parameters of that line, meaning that the more points exist in a line, the more evidence there is for the presence of that line. The *houghpeaks* function allows for detection of the two most likely lines present in the image, and by putting limits on the angles and magnitudes of those lines a classification can be made if it represents a cone.

5.1. Edge detection

For the Hough Transform to be applied, the edges defining the sides of the cones must be defined in a binary mask. For this it is necessary to apply edge detection methods to each region of interest around the cones. MATLAB provides several methods for edge detection, which are shown in Figure 22 with their default parameters. Sobel, Prewitt, and Roberts apply

direct linear filtering and then look for the maximum of the gradient, while the LoG method looks for zero crossing after filtering with a Laplacian of Gaussian. In the end, the Canny method was chosen, since having tuneable lower and upper thresholds as well as adjustable variance of the Gaussian filter helps with filtering out the noise while keeping the information about the often-weak edges of the cones.

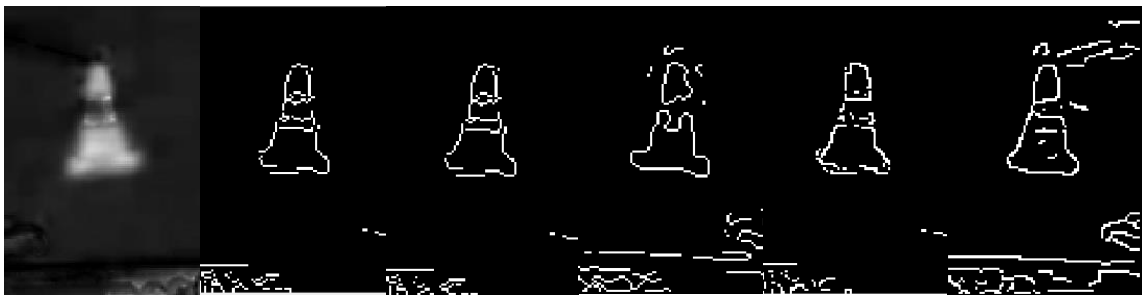

Figure 22 – Edge detections methods. Left to right: original, Sobel, Prewitt, log, Roberts and Canny

5.2. Validation, testing and discussion of the results

Figure 23 shows the lines corresponding to the Hough peaks in an accurate detection of a cone and a false positive (a yellow button on the car). Note that since the most strongly detected lines all have a negative angle, so this image is automatically rejected.

Figure 23 – Peaks of the Hough transform in a cone and a false positive

Applying this process to the image shown in Figure 10, given as an example of failure of the thresholding algorithm, the results are shown in Figure 24. The rate of false positives decreased, showing only one of the previous cases. Two of the cones were missed, however, one because it is in a group, so its edges are merged, and the other because its left edge is outside the image frame.

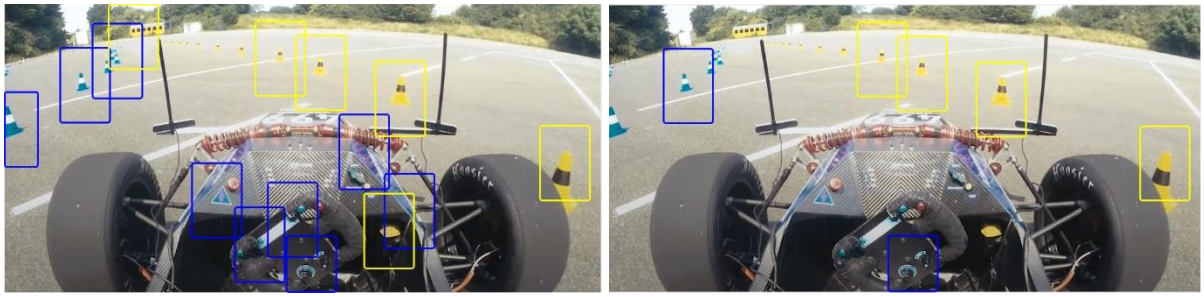

Figure 24 – Detected cones before and after processing using the Hough transform

6. Optical Flow

Optical flow is a way of estimating the apparent velocities of objects between frames of a video. MATLAB presents several algorithms that estimate the optical flow – Farneback, Horn-Schunk, Lucas-Kanade and Lukas-Kanade derivative of Gaussian. From these, Farneback and Lucas-Kanade were too sensitive to noise to identify any features in the image, while Lukas-Kanade DoG filtered out the cones' movement. Horn-Schunk manages to detect and track the movement of the cones. However, since the video is recorded from a moving camera in the car, it also detects the relative motion of the background as movement. While it is difficult to isolate the cones from the background due to this, optical flow may still be considered a useful tool for finding features to track in a video, if it is used alongside a classification system that takes the regions with highest optical flow and evaluates if they correspond or not to a cone. Figure 25 shows an example of this, where the movement of the cones relative to the car is detected but also the relative movement of the background and the movement of the car.

Figure 25 – Optical flow estimation in a video of a lap through the racetrack

7. SURF Method

Some approaches were tried by extracting features from known pictures of cones with the same shape and colours as the ones in the images. However, this algorithm was found to be too sensitive to changes in background and orientation of the cones, possibly due to the cone being a very simple shape that doesn't have enough descriptors for proper identification. For most combinations of cones and images no matches were found, and only one match was found in the whole set between an isolated cone and a cone in an image that wasn't itself, and only when setting all thresholds to the maximum value the function would allow, making it very sensitive to false positives. This result is shown in Figure 26. In this case outliers couldn't be removed because the number of match features was too low to distinguish them.

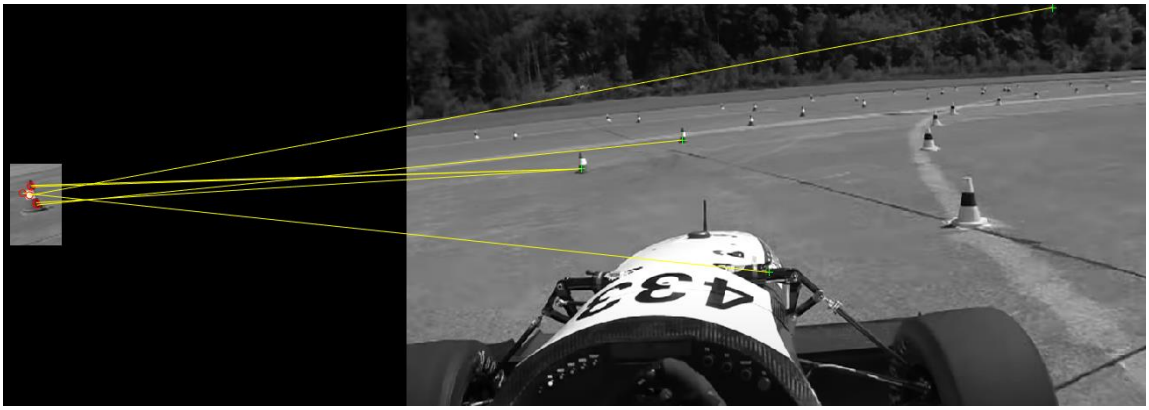

Figure 26 – Matching using SURF

8. Program Description and Tutorial

The algorithms described above were written in MATLAB 2020b. A set of test images taken from the provided YouTube videos were taken and stored in a folder called images, sent in the .zip file along with the code. The images have the correct name and image selection is done inside the program so no further action should be required.

The script that should be called first is in file "**main_script.m**". Here is implemented a command line menu that allows for selection of the various algorithms for image segmentation and processing. All algorithms except Optical Flow (Option F - which comes with its own demonstration video) and SURF (Option H - which comes with an example) need an image to be loaded first (Option A in the menu). A description of the chosen algorithm, including Options A and F is shown in Figure 27. The more complex options of the algorithm (B to E, G and H) are described further in the following chapter.

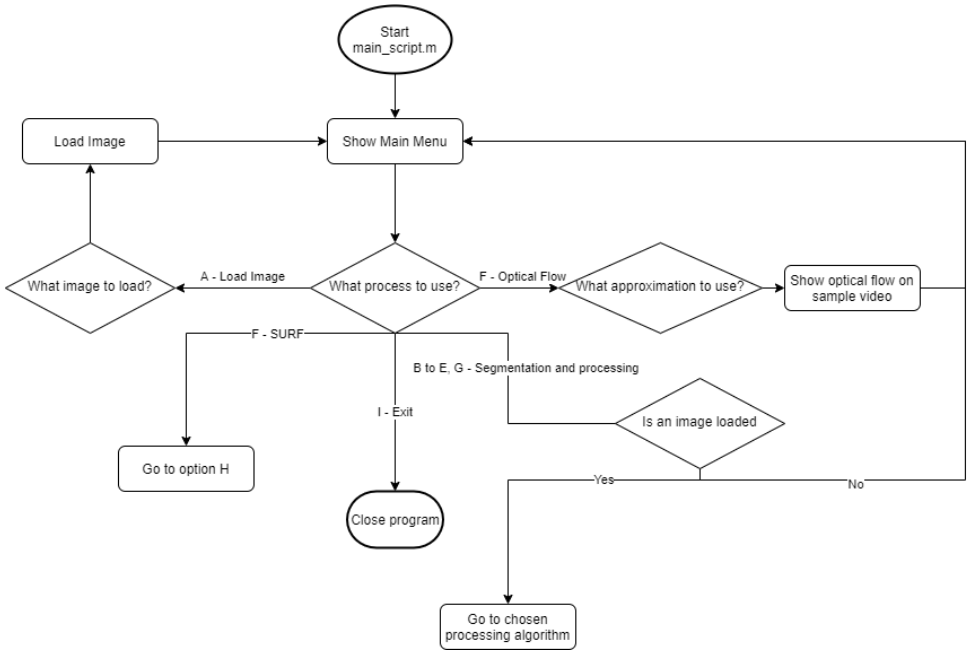

Figure 27 – Flowchart of the main script of the program

8.1. Option B – Segmentation based on HSV threshold only

This option is in the code for demonstration purposes only, since it almost always gives bad results, with high fraction of false positives. The algorithm consists in segmenting the options according to the thresholds in Table 1 with no further filtering or classification validation. The corresponding flowchart is shown in Figure 28.

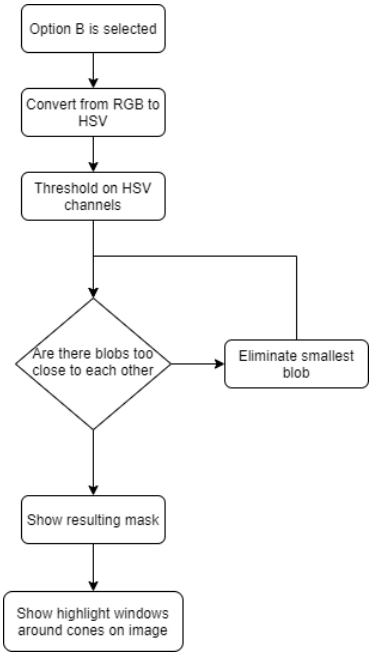

Figure 28 – Flowchart of Option B

8.2. Option C – Segmentation by HSV threshold and filtering by mathematical morphology

This option follows similar steps to option B but adds an extra step of filtering by opening and closing. The respective flowchart is shown in Figure 29.

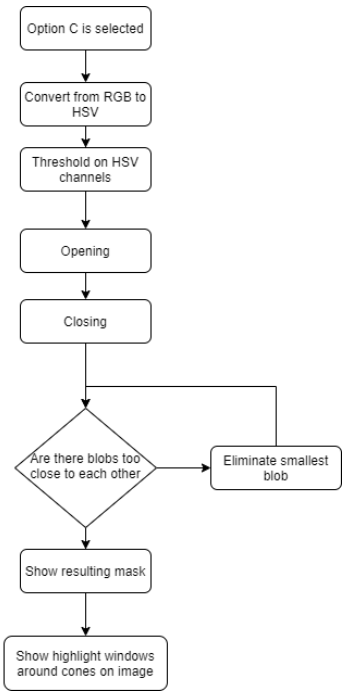

Figure 29 – Flowchart of Option C

8.3. Option D – Segmentation by threshold on HSV and Hough transform

This option also starts by a segmentation of the image using the HSV colour space but the classification of blobs into cone or not a cone is done via the Hough Transform. First the flowchart from Figure 28 is followed to obtain the regions of interest around possible cones. Then the Canny detector is used to detect edges in the region of interest, followed by the application of the Hough Transform and the classification through the properties of the peaks, as described in chapter 5 or in the flowchart in Figure 30.

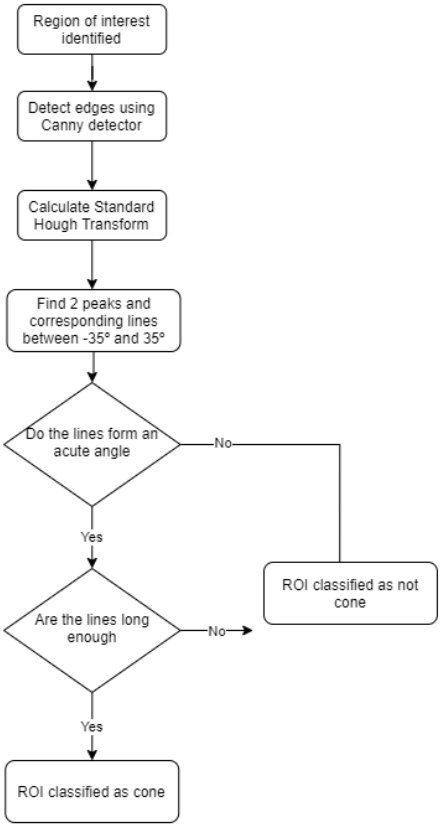

Figure 30 – Flowchart of Option D

8.4. Option E – Segmentation by threshold on HSV, mathematical morphology and Hough transform

This option consists only in the combination of options C and D, so in consists in following the flowchart in Figure 29 followed by the one in Figure 30. It is a more exclusive but also robust to false positives way to filter out the regions of interest obtained by thresholding.

8.5. Option G – K-means segmentation method

Option G corresponds to the method descripted in chapter 4. The following flowchart is also a sequence of actions, based on the same chapter.

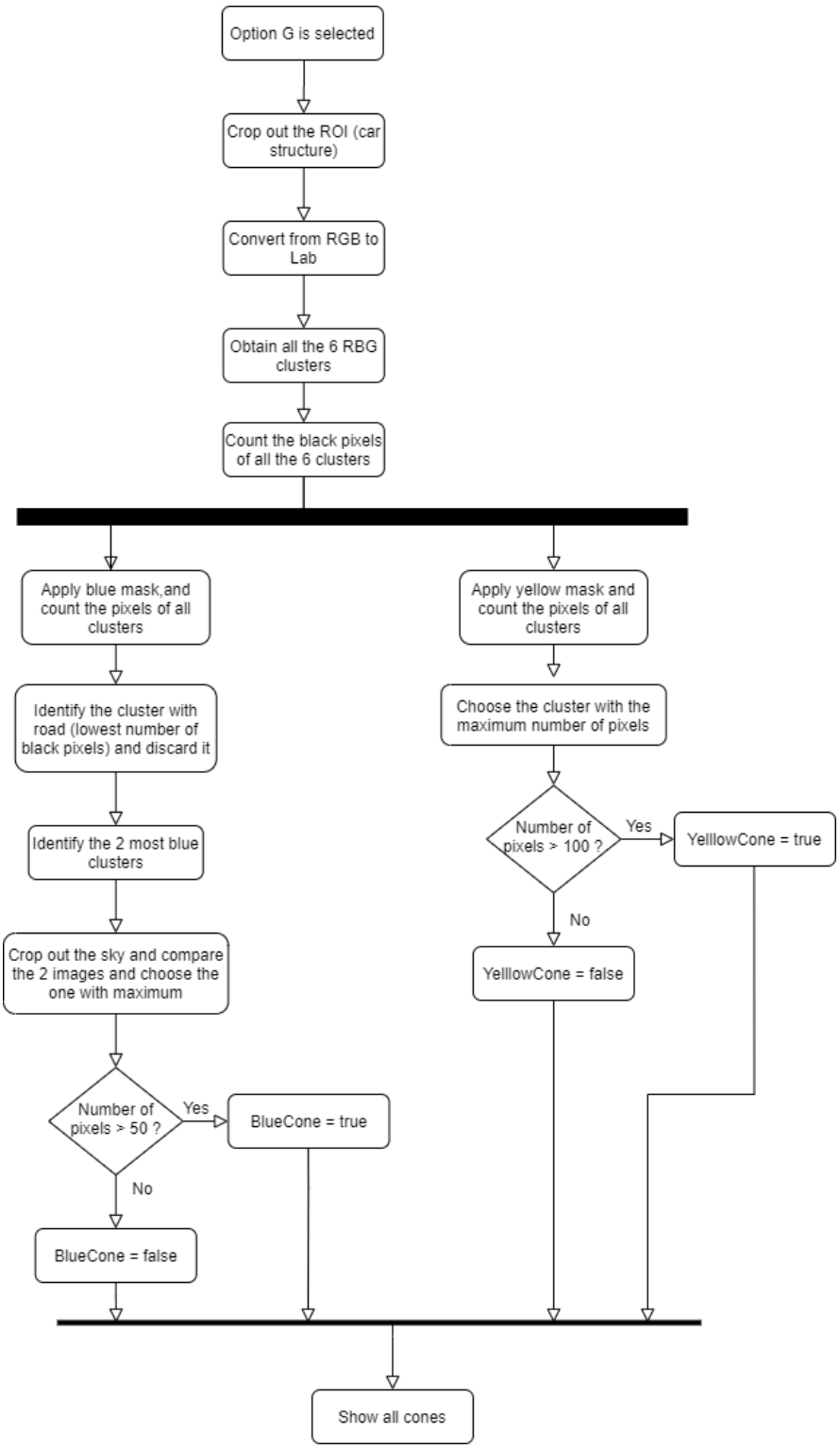

Figure 31 - Flowchart of option G

8.6. Option H – Matching SURF features

This option gives an example of matching SURF features between an example cone and cones in an image. Since it mostly yielded bad results only an example is implemented, but the image can be change by altering the name of variable “Image” in *SURFdetector.m.* The corresponding flowchart is shown in Figure 32.

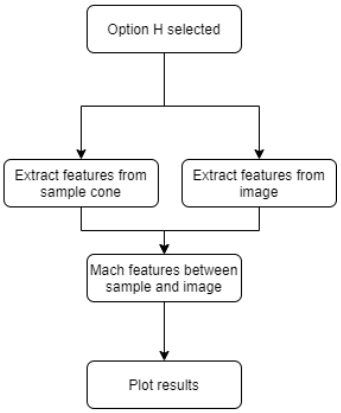

Figure 32 – Flowchart of option H

9. Conclusion

Current autonomous vehicles for Formula Student competitions use advanced machine learning algorithms that are outside the scope of this course. One example of this type of implementation is in [1], where the identification is done by a combination of the processing of LIDAR data and a Support Vector Machine classifier to identify colour. As such, it is expected that the studied techniques will not yield the robustness to changing environment conditions and overall accuracy that is seen in real applications of this type. Nevertheless, in most situations the algorithm managed to detect at least one cone of each colour, even if it had false positives. Using either the Hough transform, or the mathematical morphology alone leads to false positives in most images while using both in sequence is too aggressive of a filtering technique and leads to many cones being missed. A better tuning, possibly adjusted on the go as the car moves, of the parameters of each technique would lead to a better classification.

As a suggestion of possible improvement, with the exception of the K-means clustering based segmentation, where the method would not work without cropping out the car, all algorithms were made with the possibility in mind that the car could change, so it wasn’t cropped out manually. Performance could then be further improved by knowing the shape of the front of the car and removing it from the image beforehand. Other avenue of future work would be a more robust algorithm involving semantic segmentation using Deep Learning and CNNs.

10. References

[1] H. Tian, J. Ni, and J. Hu, “Autonomous Driving System Design for Formula Student Driverless Racecar,” in *IEEE Intelligent Vehicles Symposium, Proceedings*, Oct. 2018, vol. 2018-June, pp. 874–879, doi: 10.1109/IVS.2018.8500471.

[2] “(7) Formula Student Driverless Split View 2017 - YouTube.” https://www.youtube.com/watch?v=4ah5aZ09i6g (accessed Nov. 20, 2020).

[3] “(7) Autonomous Racing: AMZ Driverless with flüela - YouTube.” https://www.youtube.com/watch?v=FbKLE7uar9Y (accessed Nov. 20, 2020).

[4] “(7) KIT19D Dynamics | Formula Student Spain - YouTube.” https://www.youtube.com/watch?v=9_wI5vHNGTA (accessed Nov. 20, 2020).

[5] “Color-Based Segmentation Using K-Means Clustering - MATLAB & Simulink Example.” https://www.mathworks.com/help/images/color-based-segmentation-using-k-means-clustering.html?fbclid=IwAR0vHaqzHXyT15m6IdM2NqMB5H5uYYKHoF90ihpmFvl7Kw8nL_UFuc32P5I

https://www.mathworks.com/help/vision/ug/semantic-segmentation-using-deep-learning.html