University of Porto - Faculty of Engineering

Master in Informatics and Computing Engineering

Project Report

Homographies for Computer Vision

Applications and Use Cases of Homographies

Miguel Amorim
up201907756@edu.fe.up.pt
Rita Mendes
up201907877@edu.fe.up.pt
Sérgio Estêvão
up201905680@edu.fe.up.pt

April 2023

Contents

List of Figures

1 Introduction

According to the authors of [1], a homography is a projective transformation that relates the positions of points in two images of the same planar surface taken by a camera. Specifically, let x and x' be the coordinates of a point in two images, related by a homography H. Then, x and x' are related by the equation x' = Hx, where H is a 3x3 matrix of real numbers. In other words, the image of a point x in the first image is given by the point Hx in the second image.

Homographies are widely used in the field of Computer Vision, particularly in applications such as image registration, object recognition, camera calibration, and 3D reconstruction. They provide a powerful tool for relating points in different images or views of a scene, allowing for accurate alignment and correspondence.

The main aim of this report is to illustrate some uses of homographies in the field of Computer Vision, as well as test the performance of different algorithms related to feature detection, description, and matching.

We will cover the application of homographies in object localization, by finding a given object in a larger scene, distance measurement of points on an image, recognition of planar markers, and in creating a bird's eye view of a street lane.

All the applied Computer Vision algorithms used are derived from the Python binding of the OpenCV library [2].

To close this report, there will be a section for conclusions where we will analyze the obtained results. This will also include a short discussion of how the concepts applied for each optimization can be used in other situations where Computer Vision is necessary.

2 Applications

In this section, we will describe our approaches and results to four different tasks that involve the use of homographies in Computer Vision related problems.

2.1 Object Localization

In the Object Localization task, the main objective is to identify an object in a given scene, being the object partially possibly or non-planar. To do this, we use feature matching. Feature matching is the process of identifying keypoints, or distinctive features, in both images and then matching them up. These keypoints could be edges, corners, or other distinct shapes.

In this task, we also utilized different algorithms to evaluate the performance of each and the impact they have on the final solution.

2.1.1 Approach

The Object Localization task is divided into three phases: feature detection and description, feature matching, and identifying the object in the image.

In this section, we will describe what we have done in each of those phases.

1. **Feature Detection and Description**: to match both images, we needed to first find their local feature, for this we tested three different algorithms: SIFT, an algorithm that generates

unique descriptors that are invariant to scale, orientation, and illumination changes; ORB, which combines two algorithms, FAST corner detection algorithm with the BRIEF descriptor and uses an oriented approach to compute the descriptors and can handle scale and rotation changes; and BRISK, an algorithm that uses a pyramid-based scale space approach and generates a binary descriptor that is both compact and efficient.

2. **Feature Matching**: after finding the features, these needed to be matched to try to identify the object or part of it in the scene, for this we used two different algorithms, Brute Force, that tries to match every feature in both images and FLANN, that has a more efficient approach since it uses a k-d tree for fast nearest neighbor search.
3. **Outlier removal**: Next, we filter the matches and improve the results and remove outliers, in this task we tested Lowe's ratio, RANSAC, and a percentage of the matches with the shortest distance.
4. **Homography and Identification**: finally, after obtaining the final matches, the homography is calculated and a polygon is drawn in the scene surrounding the found object.

2.1.2 Results

We executed the code with three different algorithm combinations, SIFT + FLANN, the most frequently used algorithms for this task which we used as a control group, ORB + Brute Force, and BRISK + Brute Force.

The choice of the brute force feature matching algorithm in the ORB and BRISK algorithms combinations was based on the data size being used. Since the images did not have a significant size, performance was not taken into consideration, and since exact matches are crucial in this task, brute force search was more appropriate.

We tested these algorithms with three distinct scenarios, one where the object was partially occluded, one where the object shared similar features with other objects in the scene, and finally a scene where the object was non-planar.

In the *figures 1* to *3* it is possible to ascertain that the SIFT + FLANN algorithms were able to identify every object in all the tests, ORB + Brute Force and BRISK + Brute Force were only able to identify the partially occluded object. This may be due to the fact that SIFT has been shown to provide more distinctive features compared to the ORB and BRISK, particularly in images with significant changes in scale, rotation, and illumination, this can be credited to SIFT's use of a difference-of-Gaussian (DoG) approach, which enables the detection of keypoints at different scales and facilitates the capture of scale-invariant features that are robust to variations in scale and rotation. Since ORB and BRISK may produce less distinctive features but offer faster computation times compared to SIFT, they would be more suitable for applications that require real-time processing, such as robotics or mobile devices, where computational speed is a critical factor.

Regarding the RANSAC, in the *figure 4* we can notice that disregarding the use of this algorithm to remove mismatches had a significant impact on the results. All three combinations of algorithms were not able to detect the partially occluded object, this comes to confirm that although RANSAC can be computationally expensive, it is an effective and reliable approach to removing outliers and mismatched features.

Finally, we tested the control group, SIFT + FLANN, with different values for the Lowe's ratio.

In *figure 5* we can see the influence that the values have on the number of matches produced by the program, even though in all cases the SIFT + FLANN was able to identify the object, with a lower Lowe's ratio only matches that are indeed related to the object are kept, which confirms that this mismatch filter algorithm is also efficient and has a considerable impact on the output of the program.

2.2 Distance Measurement

In the Distance Measurement exercise, the main objective is to develop an application that allows the measurement of the positions of points and distances on a plane with a single camera.

2.2.1 Approach

The Distance Measurement task is divided into four phases: user interaction with the application, feature matching, computing the homography, and calculating the distance between points.

1. **User Interaction**: We show the scene to the user and wait for him to interact. We wait for 6 clicks from the user: the first two for the points that we want to measure the distance and four more to create the homography. The user can see the coordinates of the points that he is clicking on in real-time. After that, the user must click the *Enter* key to resume the rest of the process.
2. **Feature Matching**: Then, we use SIFT and FLANN to get the matches between the scene and the object. Here, we also check the user's input if he chose characteristic points or use the near to them with a maximum limit of 30 pixels. If the program can't find any characteristic point it prints to the user "No matches found".
3. **Homography**: We make two homographies, first the one with the four points received from the user and another one with the objective of making it more visually appealing to the user, where we used almost all the matches that we detect earlier.
4. **Calculate distance**: Finally, with the matches, we get the coordinates from the points on the frontal image, and with a function we computed earlier we can calculate the distance between the projection points. With the pixel density of the image, we can convert the distance to mm (millimeters).

2.2.2 Results

With this approach, we were able to achieve good results on projection and matching points as can be seen in *figures 5*. However, we noticed that there would be space to improve. Due to our dependence on user interaction to define the points, most of the time we work with near points and not exactly the ones that had been clicked on by the user. We also tried without using RANSAC to filter mismatches features and we noticed a significant decrease in the quality of the results, as seen in *figures 6*.

2.3 Marker Recognition

This section explores the application of homographies to the recognition of augmented reality markers in a picture. The proposed task involves developing an application capable of finding two different augmented reality markers in a picture and drawing a rectangle around each one in different colors.

2.3.1 Approach

The application we developed consists of four main phases: detecting common points in the image, calculating the homography, transforming the templates of the markers to be in the same plane of the image, and performing template matching.

In this section, we will describe what we have done in each of those phases.

1. **Common points**: To find the four matching points of the images that we need to calculate the homography we can take different approaches, from manually selecting them to using feature or corner detection algorithms, such as SIFT or Harris corner detection, respectively.
2. **Homography**: The next step is to find the perspective transformation between two planes. As we are calculating the homographies with the minimum number of matches allowed, RANSAC won't be as relevant as in other problems.
3. **Warping**: With the homography, we can then warp the markers' images to be in the same plane as the corresponding marker in the scene. Then, we crop the images to contain only the marker's content and apply thresholding to improve the next phase's performance. The results of this phase can be seen in *figure 7*.
4. **Template Matching**: The final step is to apply template matching to find which portions of the scene match the markers' templates. In this step, we had to try scaling the image with different factors and determine which fitted best. Finally, we calculated the coordinates of the rectangles' points and drew the polygons surrounding them.

2.3.2 Results

Based on various tests using different approaches to find the matching points of the images, we found that what yielded the best results was manually selecting the corners of the markers since the algorithmic methods were struggling to correctly detect at least four correct matches between the template and the scene. With that in mind, we decided to get the template markers' corners from their geometries and prompt the user to select the corresponding corners in the image. One limitation of our implementation is that the user must select the markers' corners in the same order they are loaded into the program and clockwise, starting from the upper left.

With our methodology, we were able to achieve correct detection of the markers as seen in *figure 8*. Even with satisfactory results, we consider that there is further room for improvement, namely in the first phases. Choosing better images could yield better results with automatic corner detectors, which would remove the need for user input. Having more matches would also improve the quality of the homography since it would make sense to use Lowe's ratio and RANSAC to improve the quality of the matches.

2.4 Bird's Eye View

In the Bird's Eye View exercise, the main objective is to develop an application that processes an image of a lane, as seen by a car driver, generating a bird's eye view of the lane.

A Bird's Eye View is an elevated view of an object or location from a very steep viewing angle, creating a perspective as if the observer were a bird in flight looking downwards. Bird's-eye views can be aerial photographs, but also drawings, and are often used in the making of blueprints, floor plans, and maps.

2.4.1 Approach

This application has 9 main steps:

1. **Select White Yellow**: We create white and yellow masks and we use them on the images
2. **Convert to gray**: Convert the result of the earlier step to grayscale images, easy to work with
3. **Apply Smoothing**: Apply Gaussian Filter to add smoothness to the images, making easier the next task
4. **Detect Edges**: Detecting edges with the Canny Algorithm
5. **Select Region**: Select the region of interest and filter it with function cv2.fillPolly.
6. **Detect Lines**: Detect the lines on the region of interest with the HoughLinesP algorithm
7. **Lane Lines**: Detect the lane and the lane lines with functions like average slope intercept that computes the averaged multiple lines detected for a lane line or make line points that convert a line represented in slope and intercept into pixel points
8. **Create Dotted Line**: Here, we stop using lines and start working with points, giving the idea of dotted lines.
9. **Change perspective**: Finally, we change the perspective of the images to get the top-down view, which was the final goal here. This can be seen in *figure 19*.

2.4.2 Results

With this approach, we were able to achieve noteworthy results in the detection and were able to bird's eye view of a street lane. We want to highlight that our application works with any image size since our transformation in the final step is done taking into consideration the height and width of the image. All intermediate steps and images obtained through our methodology can be seen in *figure 9* to *figure 19*.

One approach that we took into consideration was to find the vanishing point of the image, where the street lane lines would culminate, and find the horizon line, however, we concluded that the first approach that we took, which is described in the previous section, was able to return sufficiently good results.

3 Conclusions

Through this project, we were able to study the uses of homographies in the field of computer vision, as well as understand how many algorithms used for feature detection, description, and matching operate under images with different characteristics.

On one hand, we observed the results and performance of different algorithms regarding the Object Localization task and were able to conclude that, although SIFT and FLANN outperformed the algorithm combinations using ORB and BRISK, these last two still have a valuable application in other use cases. On the other, we notice the impact that the removal of feature mismatches, in this case using RANSAC, has in many of the applications.

Furthermore, we were able to comprehend the various applications of homographies to solve real-world problems, namely the detection of objects and markers to aid in augmented reality, the measurement of distances on a plane with a single camera, and the synthesis of a bird's eye view of a plane, that could be applied in autonomous driving mechanisms.

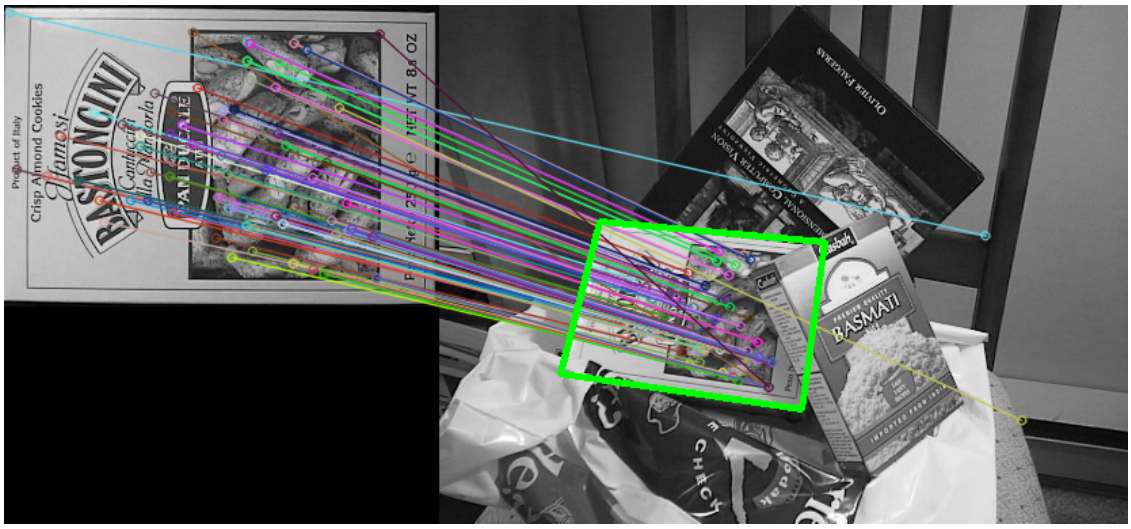
In the end, we were content with our results, however, there was still a margin for improvement as mentioned in the report.

References

[1] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second ed., 2004.

[2] Itseez, *The OpenCV Reference Manual*, 2.4.9.0 ed., April 2014.

Appendix

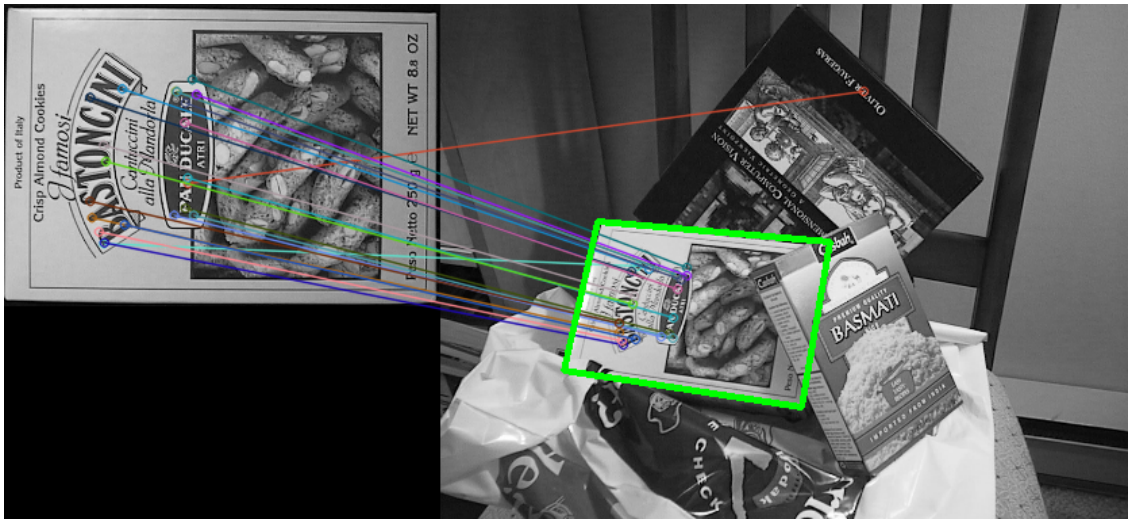


(a) Detection of a partially occluded object

(b) Detection of a non-characteristic object

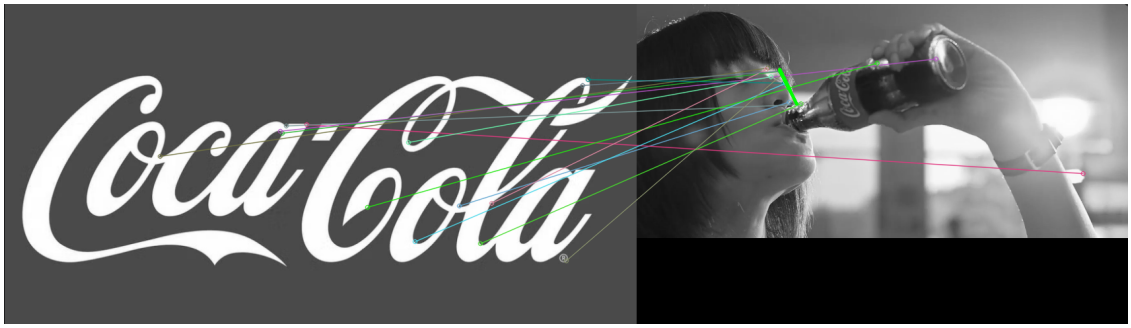
(c) Detection of a non-planar object

Figure 1: Results using SIFT, and FLANN, and RANSAC for Object Localization



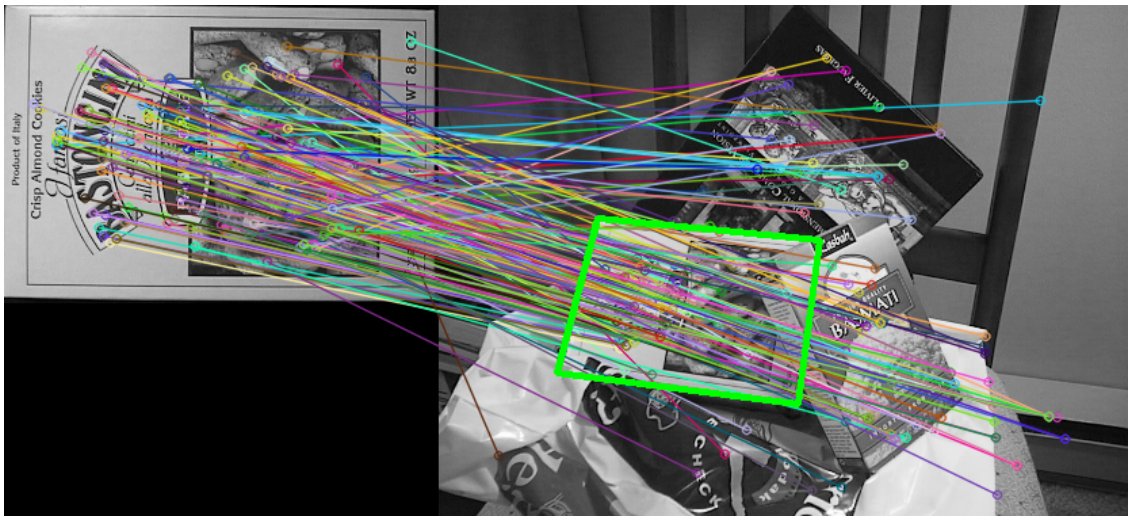
(a) Detection of a partially occluded object

(b) Detection of a non-characteristic object

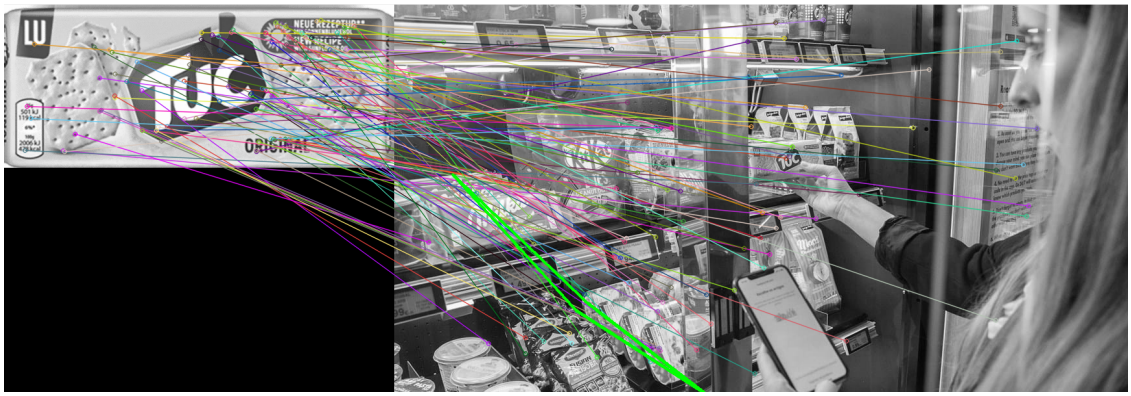


(c) Detection of a non-planar object

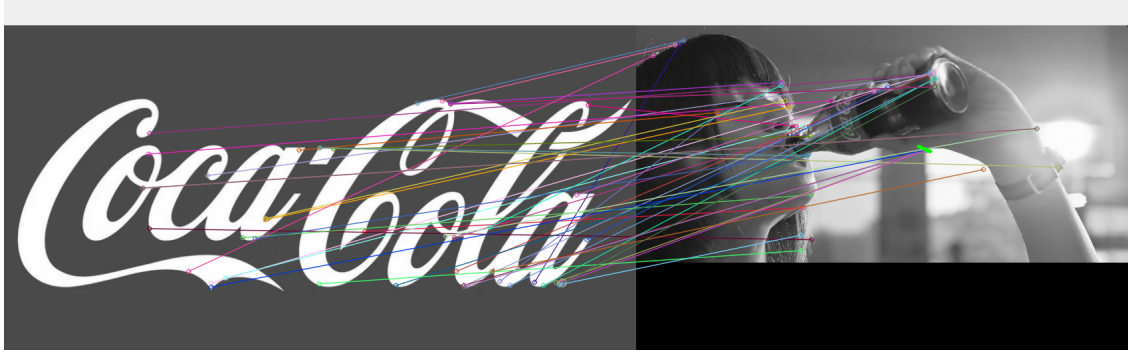
Figure 2: Results using ORB, Brute Force with hamming distance, and RANSAC for Object Localization



(a) Detection of a partially occluded object

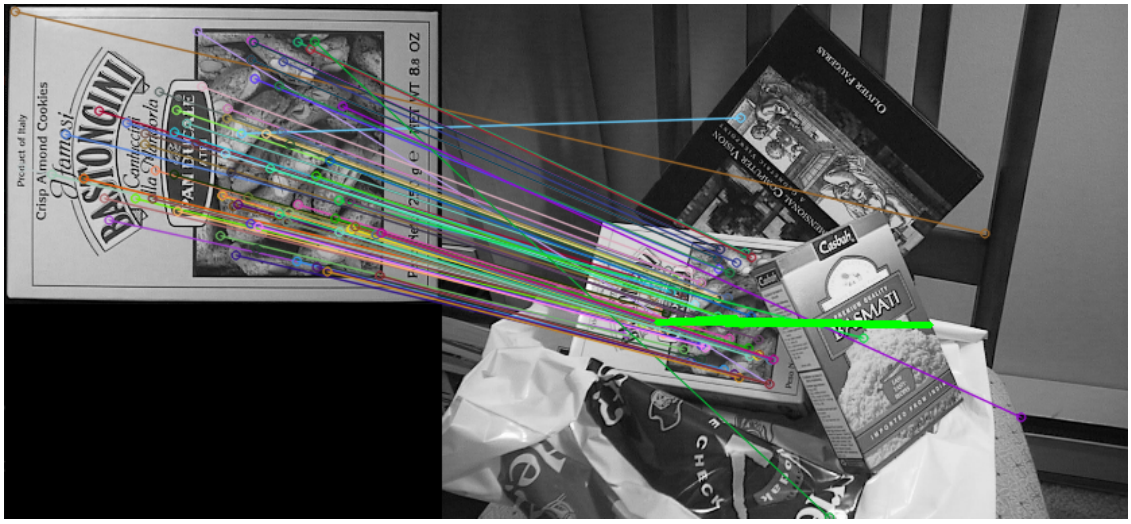


(b) Detection of a non-characteristic object

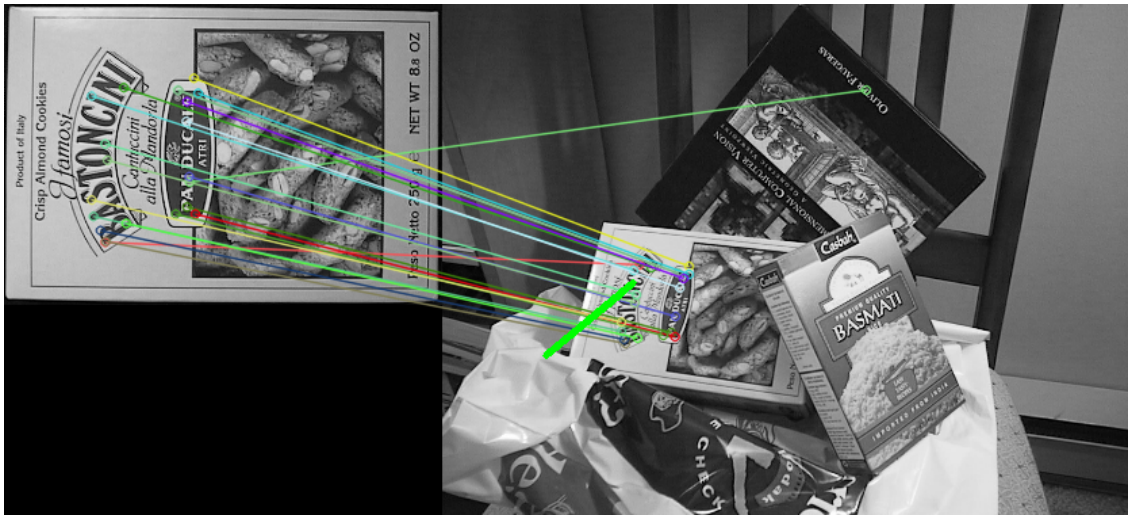


(c) Detection of a non-planar object

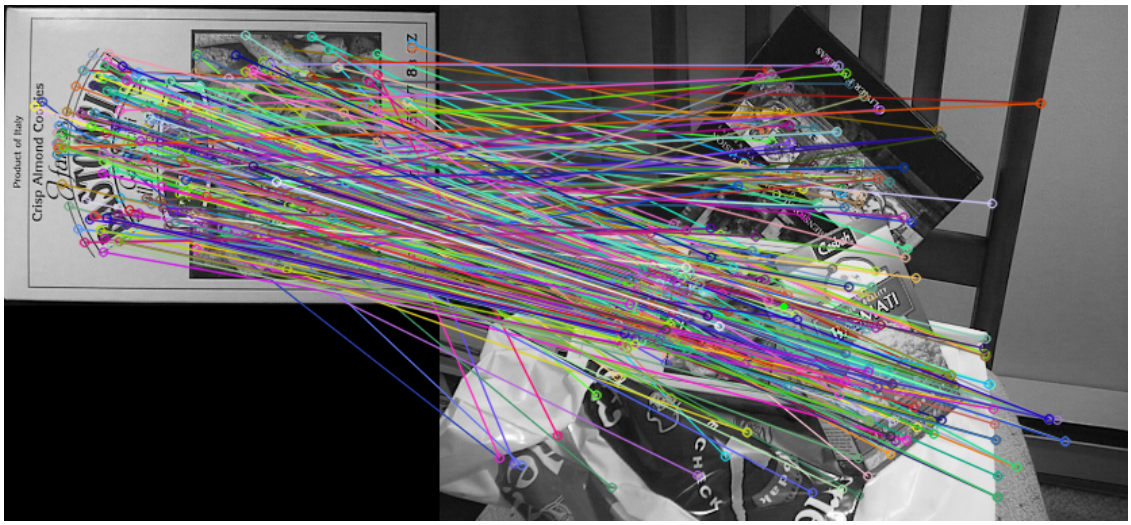
Figure 3: Results using BRISK, Brute Force with hamming distance, and RANSAC for Object Localization



(a) Results using SIFT, FLANN without RANSAC

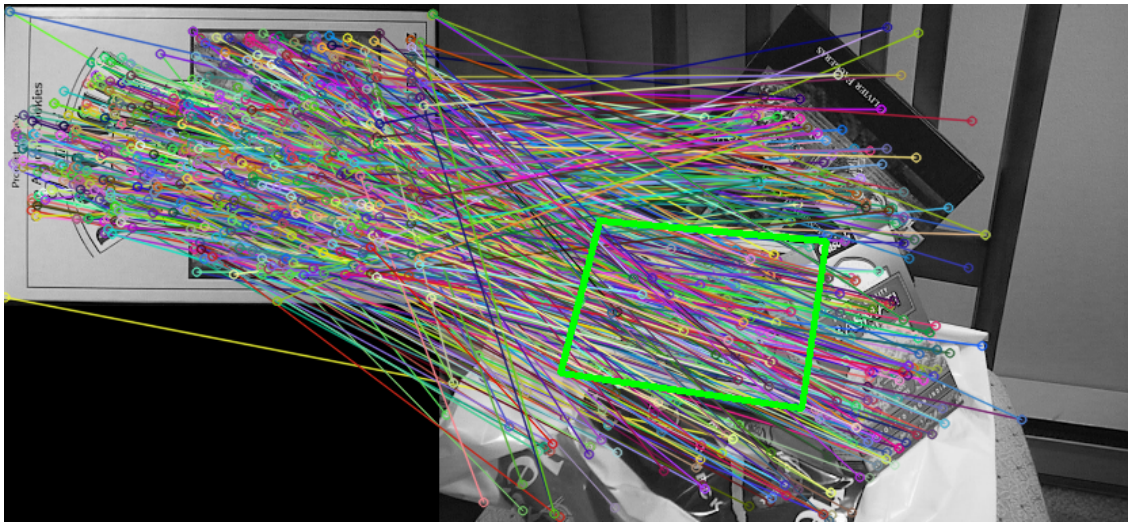


(b) Results using ORB, Brute Force with hamming distance without RANSAC

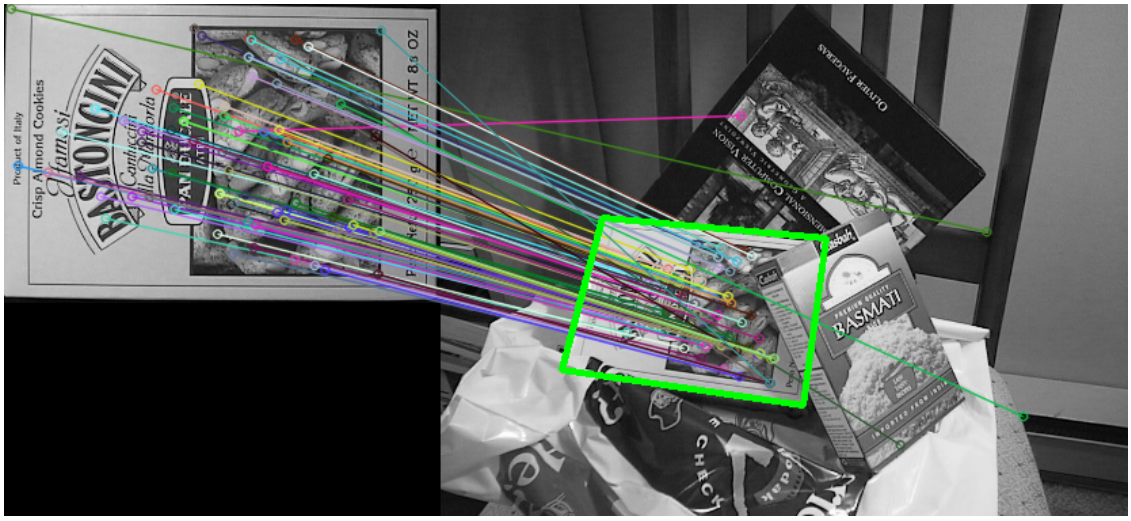


(c) Results using BRISK, Brute Force with hamming distance without RANSAC

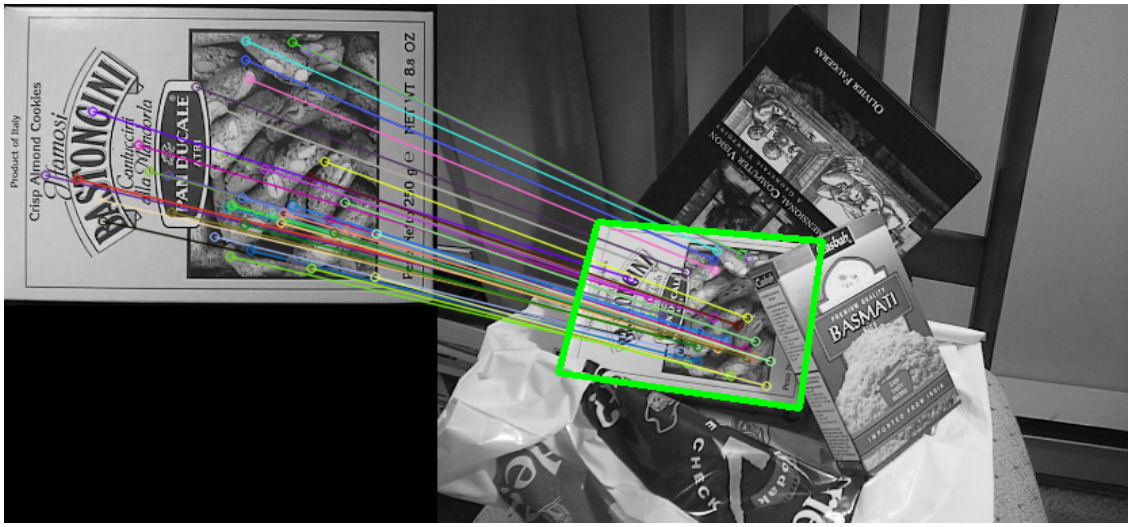
Figure 4: Impact of not using RANSAC in Object Localization



(a) Results with a Lowe's ratio of 1



(b) Results with a Lowe's ratio of 0.75



(c) Results with a Lowe's ratio of 0.5

Figure 5: Impact of different values of Lowe's ratio in Object Localization

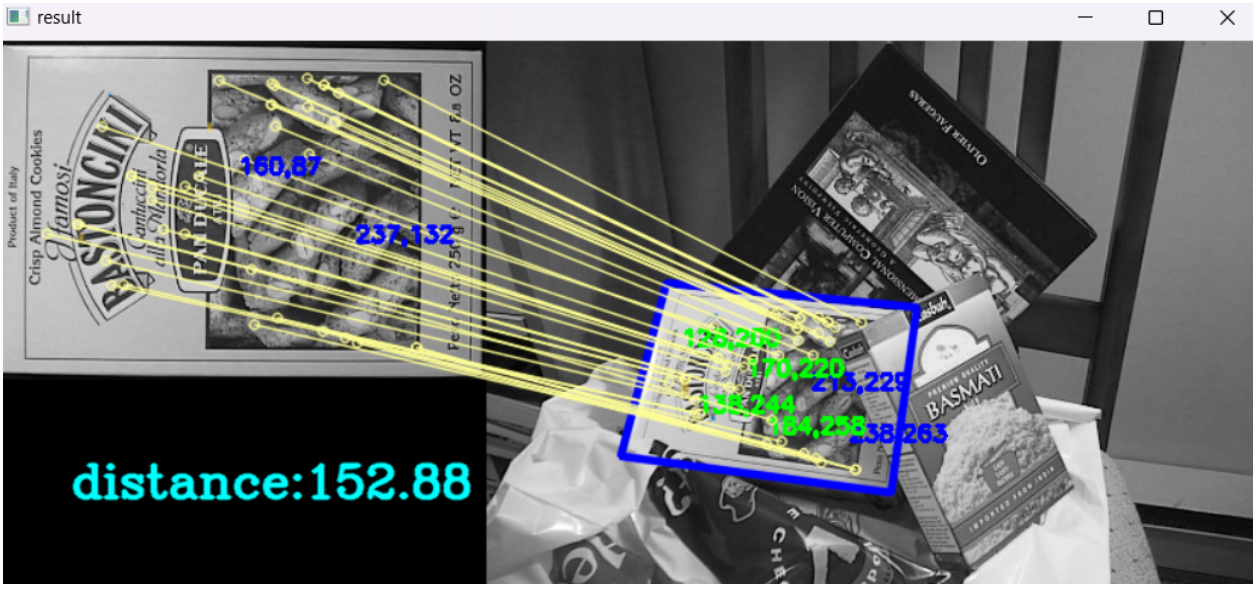


Figure 6: Result of projection and getting the distance with RANSAC

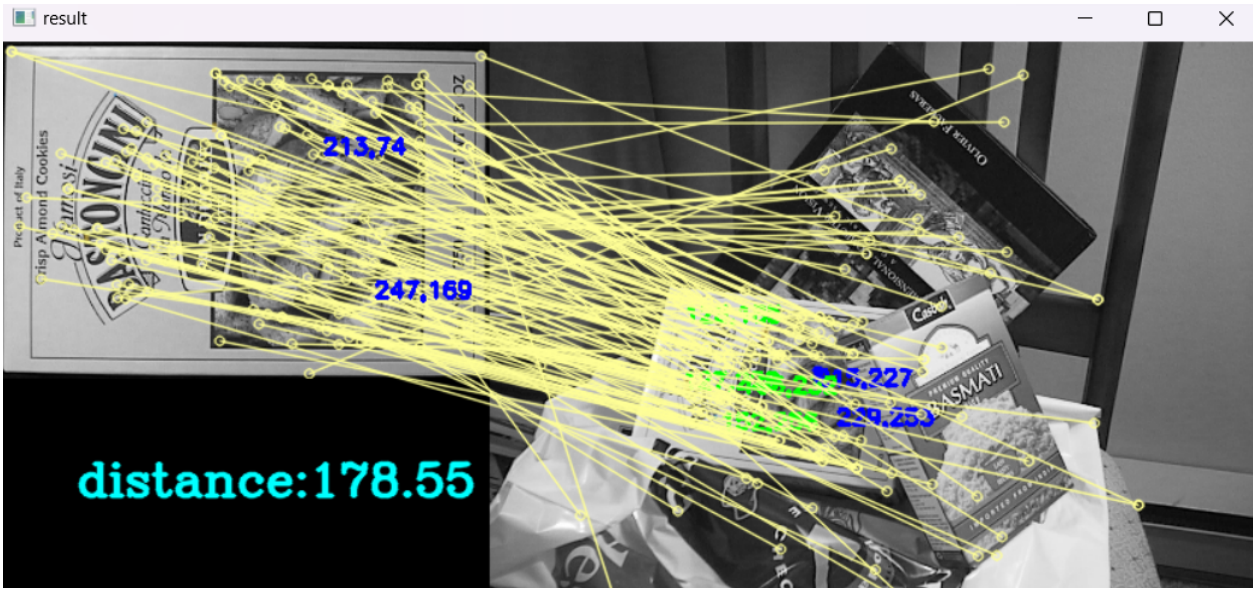
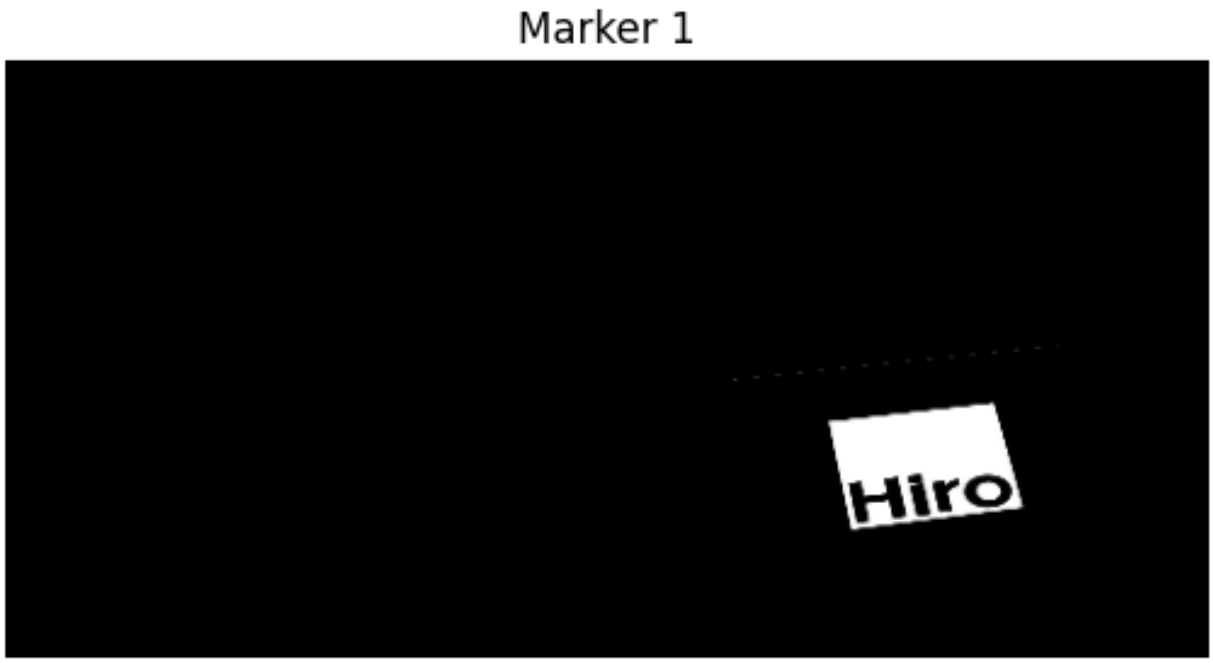
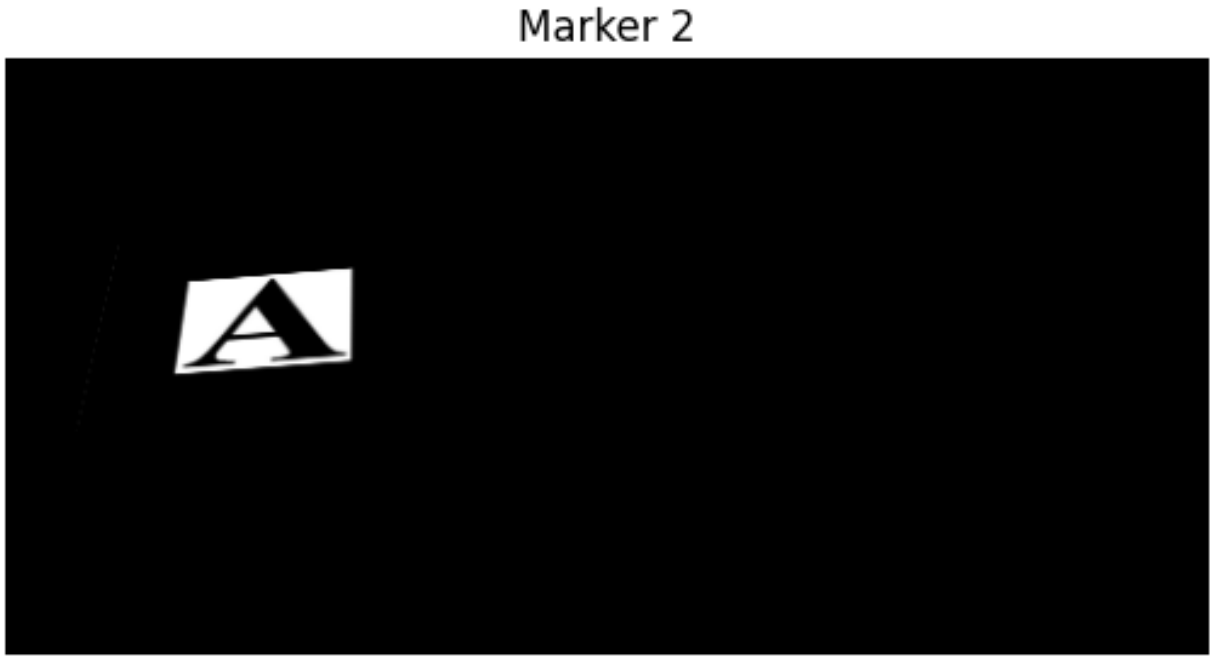


Figure 7: Result of projection and getting the distance without RANSAC



(a) "Hiro" marker



(b) "A" marker

Figure 8: Markers transformed into the scene's plane using the estimated homography

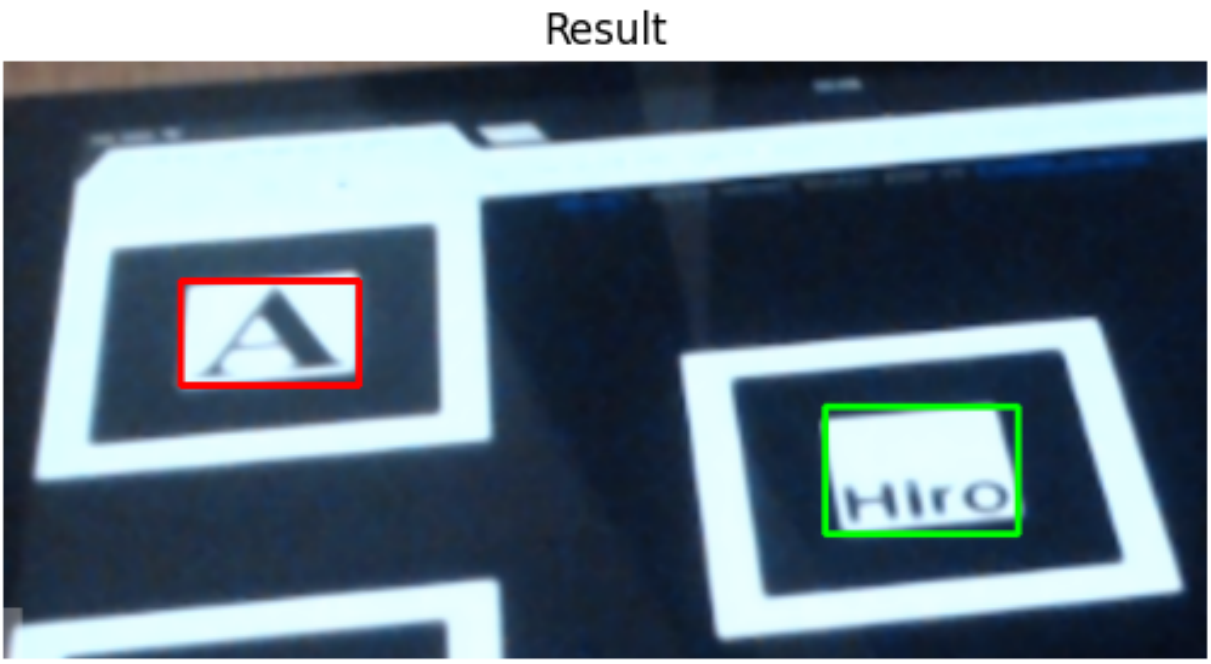


Figure 9: Both markers detected and outlined by rectangles of the corresponding colors

Figure 10: Example of images used on exercise 4

Figure 11: After using the yellow and white masks

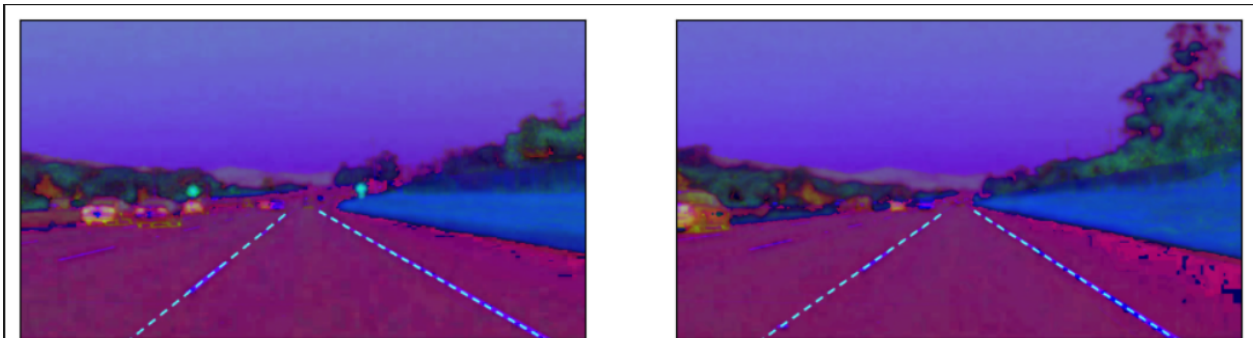


Figure 12: Result of converting to hsv

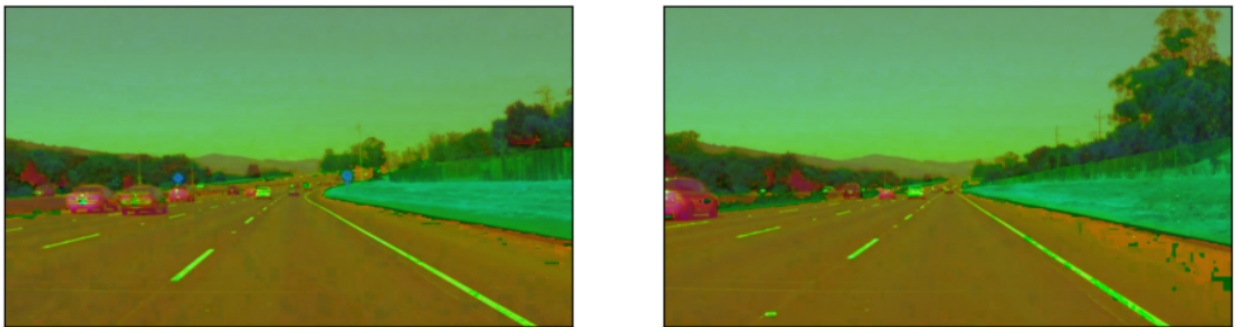


Figure 13: Result of converting to hsl

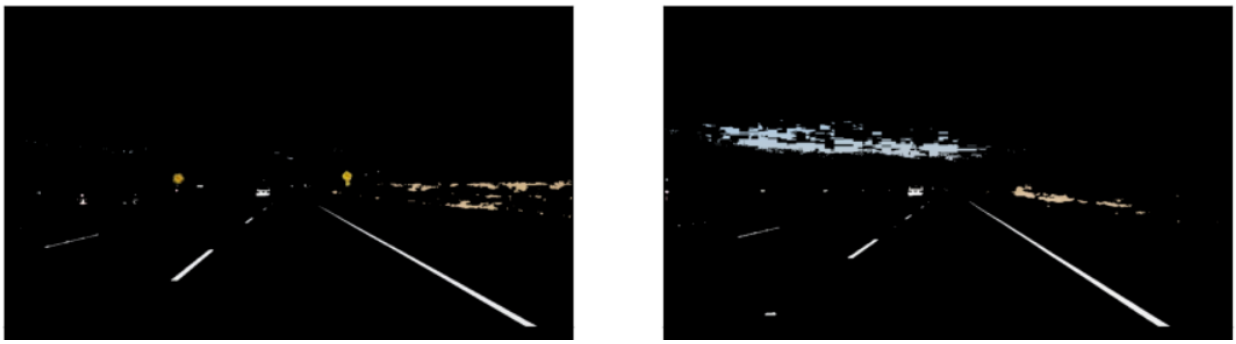


Figure 14: Result of applying the yellow and white maks after converting to hsl

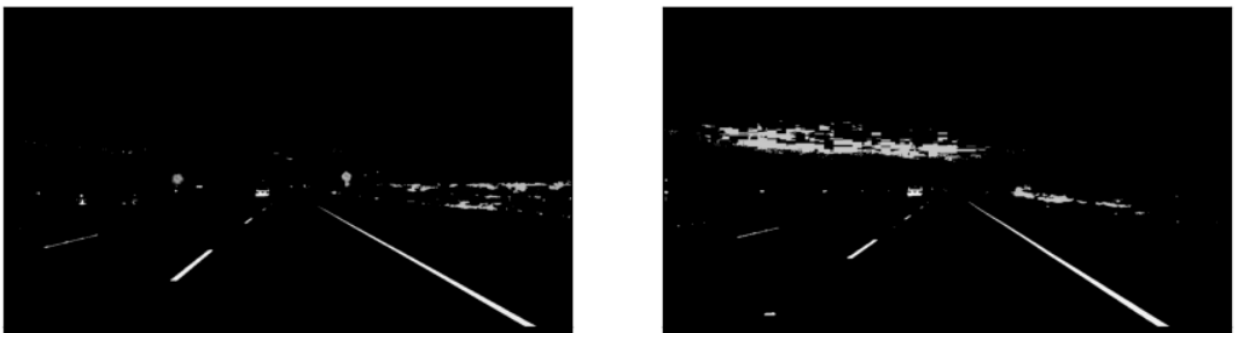


Figure 15: Result of converting the previous step to grayscale

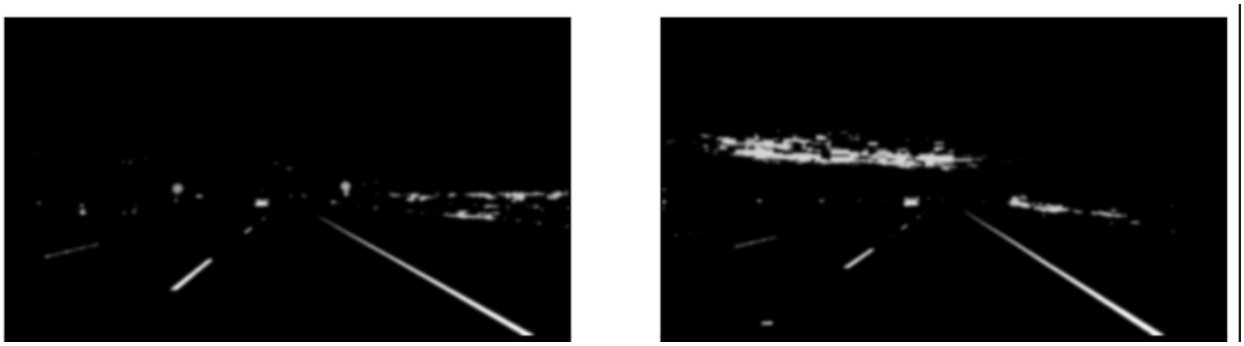


Figure 16: Result of applying the Gaussian filter

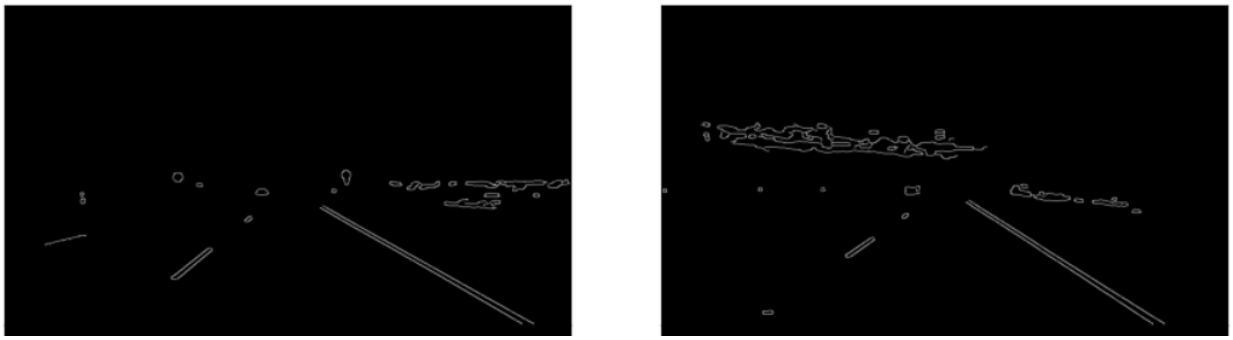


Figure 17: Result of Canny's edge detection

Figure 18: Region of interest

Figure 19: Result of drawing lines on the original images

Figure 20: Result of changing the perspective on the dotted images