U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Computer Vision

Project 1

Group 11

Frederico Lopes (201904580)
Marta Mariz (201907020)
Miguel Freitas(201906159)

Problem 1

Objective:

The objective of this problem was to develop an application that localizes an object in a scene, even when the object is partially occluded. This part of the report describes the implementation and evaluation of two feature point detectors, namely SIFT and ORB, for detecting and matching the keypoints in the images. Different parameters for matching the points were tested and also the influence of using the RANSAC method. The project was tested with different objects and scenes, including non-planar objects.

Methodology:

The images were first converted to grayscale to simplify the processing. Then, the SIFT and ORB algorithms were used to detect keypoints in the images. The descriptors of the keypoints were matched using brute force, and the incorrect matches were deleted using the RANSAC method and Lowe's ratio test.

The implementation of the project was done in Python, using the OpenCV library. The evaluation of the project was done using different objects and scenes, including non-planar objects. The evaluation criteria were the accuracy and speed of the application.

Discussion:

The implementation of the project showed that the SIFT algorithm was more reliable than the ORB algorithm for localizing objects in scenes. The SIFT algorithm was able to detect more keypoints and match them more accurately than the ORB algorithm.

The use of the RANSAC method and Lowe's ratio test was also important for improving the accuracy of the application. The RANSAC method helped to eliminate incorrect matches, while Lowe's ratio test helped to remove ambiguous matches, even though it left some incorrect matches.

The evaluation of the project showed that the application was able to detect and localize non-planar objects with high accuracy, as long as there were enough keypoints available for matching. However, the application was limited by the number of keypoints that could be detected in the images.

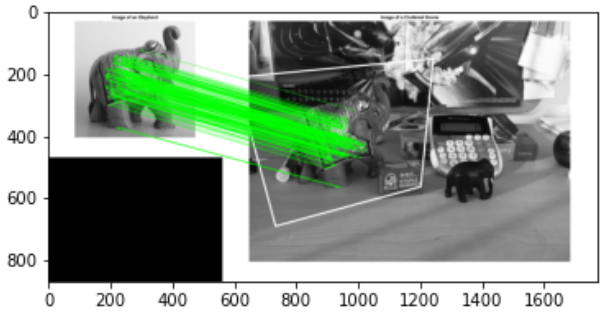


Fig 1 - SIFT and RANSAC

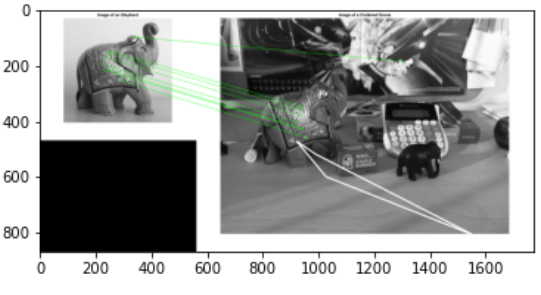


Fig 2 - ORB and RANSAC

Problem 2

Objective:

The objective of this problem was to develop an application that can measure distances on a plane using a single camera, using points whose coordinates were well known in order to be able to calculate the real distance between two points on that plane. A setup with 4 known points was compared to one with 10 known points with some wrong correspondences established.

Methodology:

Firstly, we had to create our own images of a plane: one with 4 points marked on it and another with 10 points, in order to be able to demonstrate the effects of the RANSAC method later on.
The app starts out by reading the image of the plane and asking the user to select the points on the image and inputting their real plane coordinates - the origin (0,0) is the top left corner.
With the minimum 4 points required for homography calculation, we do just that and we warp the image to prove that we indeed obtain a frontal view of the plane.

Now that we have a reliable way of converting from image coordinate to plane coordinates, we ask the user to select two points on the image and we then display those points' coordinates and the euclidean distance between them.
For demonstration purposes, we tried the same experiment but now with 10 known points.
We introduced a number of wrong correspondences between the image and plane coordinates in order to test the effect this would have on the warped perspective of the image.

The implementation of the project was done in Python, using the OpenCV library. The evaluation criteria were the accuracy and speed of the application.

Discussion:

In this exercise, we have explored the use of the Random Sample Consensus (RANSAC) algorithm for attenuating the effects of wrong correspondences in homography calculation.

Our experiments have shown that RANSAC can effectively improve the accuracy of homography estimation, successfully safeguarding against the presence of wrong correspondences.
The results of our experiments demonstrate that RANSAC is a powerful tool for robust homography estimation. By iteratively selecting a subset of correspondences and fitting a homography model to them, RANSAC is able to identify and remove outliers and wrong correspondences, resulting in a more accurate estimate of the homography matrix.
One potential limitation of using RANSAC is its computational complexity, which can be significant for large datasets. However, several optimizations and extensions of the algorithm have been proposed to address this issue, such as the use of parallel computing or hierarchical RANSAC. In our particular case, the most points we tested were 10, so this wasn't a real concern, however, it is a known drawback of the algorithm.

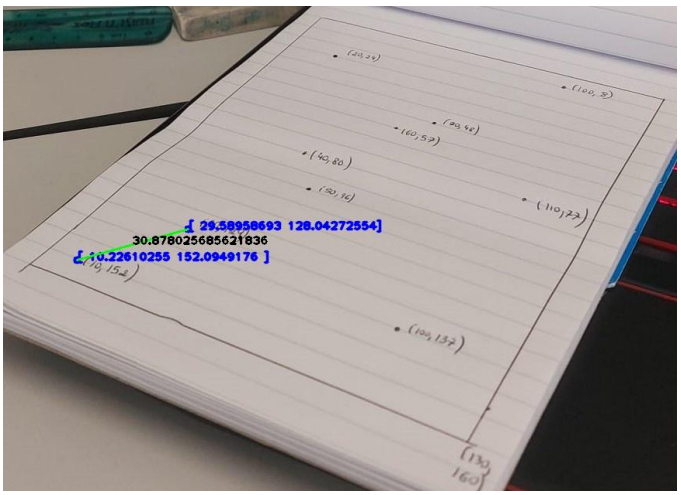


Fig 3 - Distance between 2 points in the plane

Fig 4 - Warped Perspective with 4 wrong correspondences (no RANSAC)

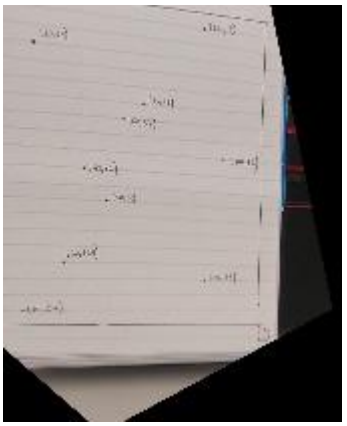


Fig 5 - Warped Perspective with 4 wrong correspondences (RANSAC)

Problem 3

Objective:

The goal is to develop an application that recognizes reference markers, in an image taken from a perspective view. Given a perspective image and two markers the application needs to be able to draw a red rectangle around the first marker and a green around the second one.

Methodology:

Firstly the user needs to select the corners for the first marker. The corners can be provided in whatever order, there is a function that will reorder them to get the desired result.

Then we calculate the homography with the points from the first maker and the selected points from the perspective image. With this homography we perform warp perspective, to guarantee that the image is not cropped we sum an offset to the destination sizes. Afterwards we perform template matching with TM_CCOEFF_NORMED and draw a red rectangle around this marker in the warped image.

Next it's necessary to warp the perspective of the image again to be able to identify the second marker. We repeat the process above, ask the user to select the marker 2 corners in the image, calculate the homography and warp perspective.

Finally we calculate and apply the inverse of the homographies we performed to get to the original image now with the markers identified.

Discussion:

We did not apply a corner detection algorithm, we simply ask the user to manually select the corners. We tried different algorithms and they were not selecting the right corners.

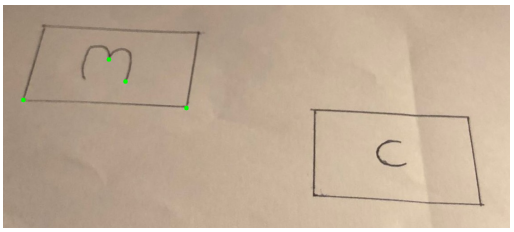


Fig 6 - Results of FAST corner detection for the first half of the image

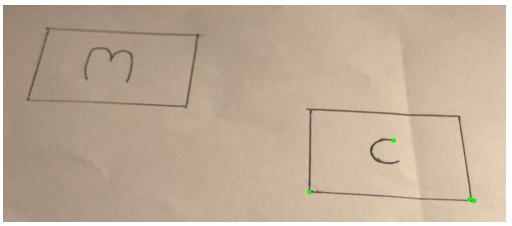


Fig 7 - Results of FAST corner detection for second half of the image

Warp perspective crops the image to only fit the content identified in the corners provided to the calculation of the homography. Because we wanted to recreate the entire image but with the markers detected this turned out to be a problem. To face it, we create a bounding box and apply an adjustment matrix to make sure that the entire warped image is visible and fits within the output image, while maintaining the correct perspective.

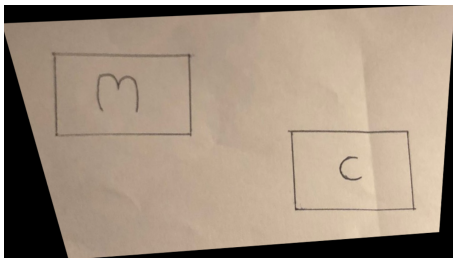


Fig 8- Result after the first warp perspective

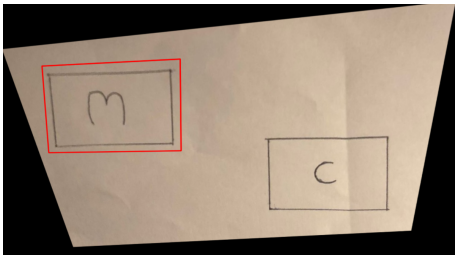


Fig 9 - Result after the second warp perspective

In terms of results in the next images there is the image provided and the markers identified.

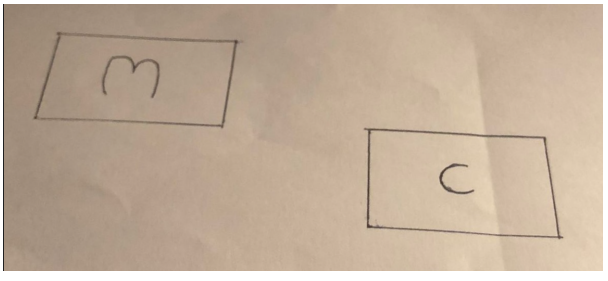


Fig 10 - Initial image

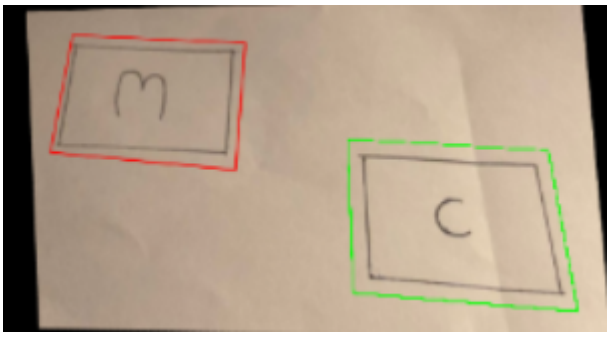


Fig 11- Results

We were able to identify the right markers in the image.

We also tried identifying the markers when they are on two different 3D planes. We were able to identify them but with the changes of size we could not get a proper final image with the inverse warp perspective homography applied.

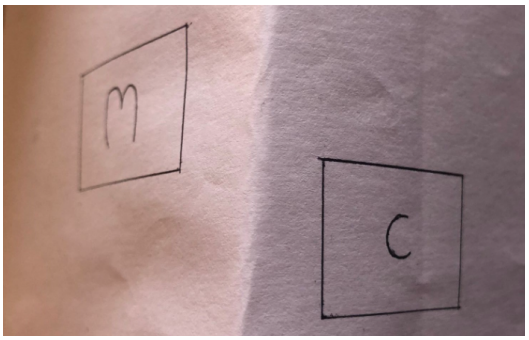


Fig 12 - Original second image

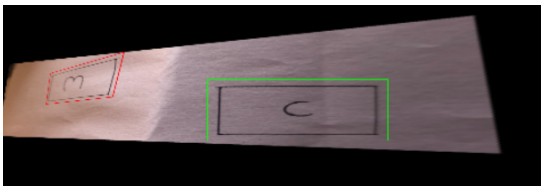


Fig 13 - Result

Problem 4

Objective:

The objective of this problem was do 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.The following text will the describe the implementation of this application, which process involves various steps, including image calibration, edge detection, line detection, and perspective transformation.

This report discusses the methodology used to develop the application and explores the implications of simplification assumptions made in the image processing steps. All the images used were obtained from the following online course: [Become a Self-Driving Car Engineer | Udacity]

Methodology:

The first step in developing the application was to calibrate the camera using images of a chessboard. The calibration process involved having photos of the chessboard in different angles, finding the corners of the chessboard and using these values to calculate the camera's parameters which were used to undistort the images of the road.

Next, the image was converted to grayscale, and Gaussian blur was applied to reduce the noise. The edges were then detected using the Canny detector. After that, a Hough transform was applied to the detected edges to detect lines. The detected lines were then averaged to find the two lines corresponding to the traffic lane.
Using the two lines, the points corresponding to the beginning and end of each line were found. These points were then used to calculate the perspective matrix, which was used to transform the image into a bird's eye view.

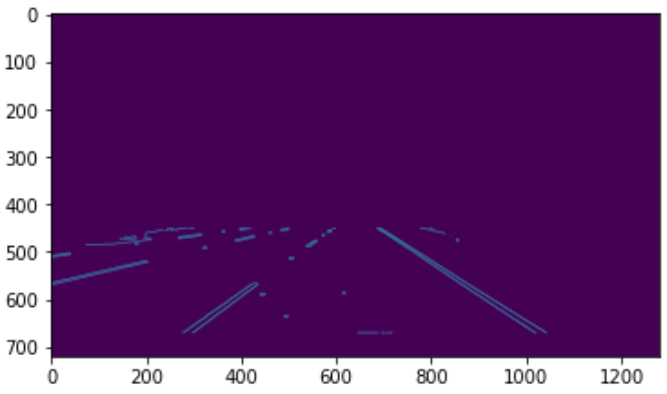


Fig 3 - Canny edge detection

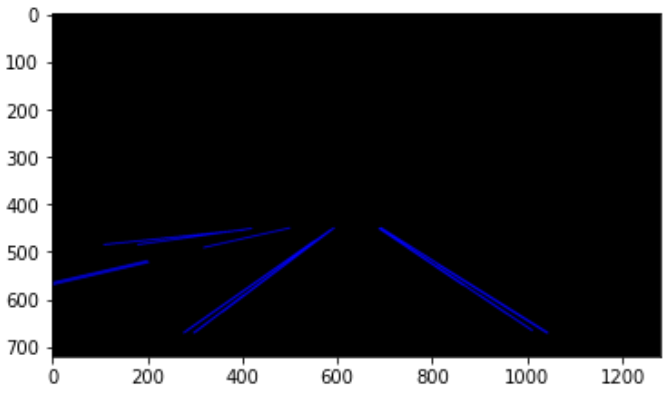


Fig 4 - Hough transform

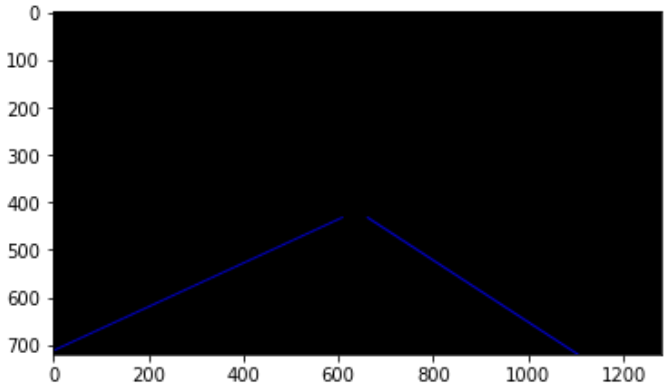


Fig 5 - Line averaging

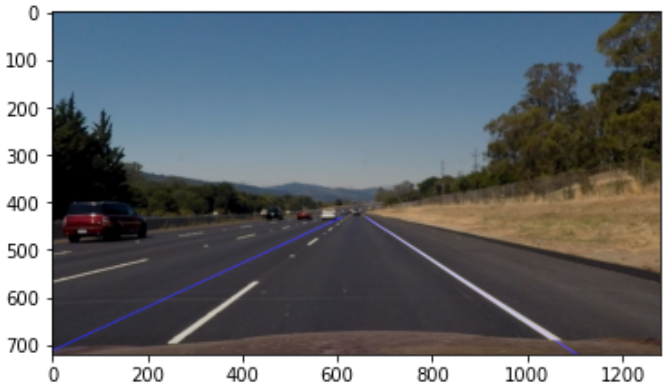


Fig 6 - Line over image

Fig 7 - Bird's eye image

Discussion:

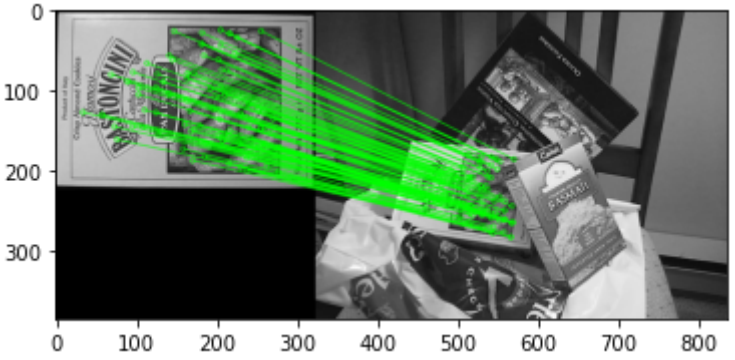
One of the simplification assumptions made in this application is that the road is that the lane is always in the same region on the image. This simplification was made to ignore the edges detected above the horizon since the camera was fixed to the car altough in some occasions such as hills this may not work. Another assumption is that the lane markings are straight and continuous. This assumption will not hold in real-world scenarios, where the lane markings may be curved or broken, leading to inaccuracies in line detection.

One issue present in this application is that when detecting the lines from the edges, some of the detected edges do not correspond to the road lines and this leads to an incorrect calculation of the average line. Another potential issue is that it relies on the quality of the camera calibration. If the calibration is not accurate, it can lead to errors in the perspective transformation, resulting in an incorrect bird's eye view.

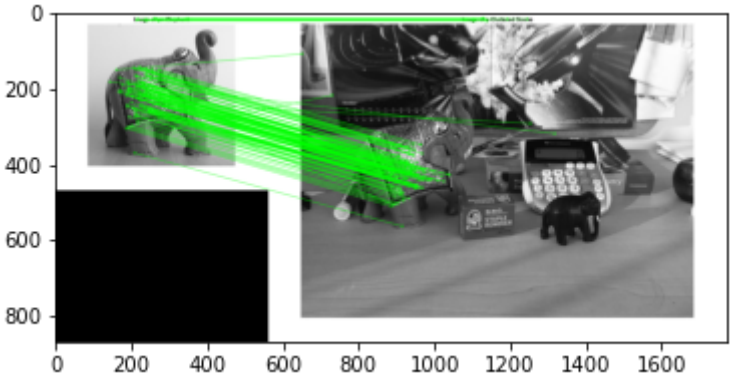
To address these issues, further research can be conducted to explore alternative approaches for perspective transformation and line detection that can account for the uneven and sloping surfaces utilizing more advanced techniques, such as deep learning-based methods.

Annexes

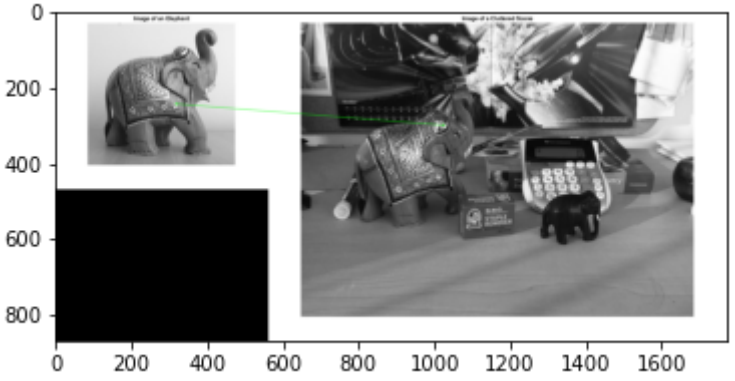
Problem 1



SIFT and RANSAC

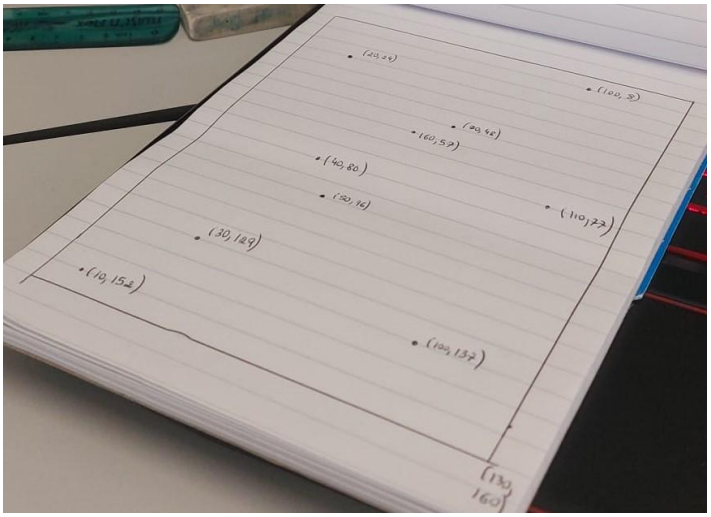


SIFT and Lowe’s test

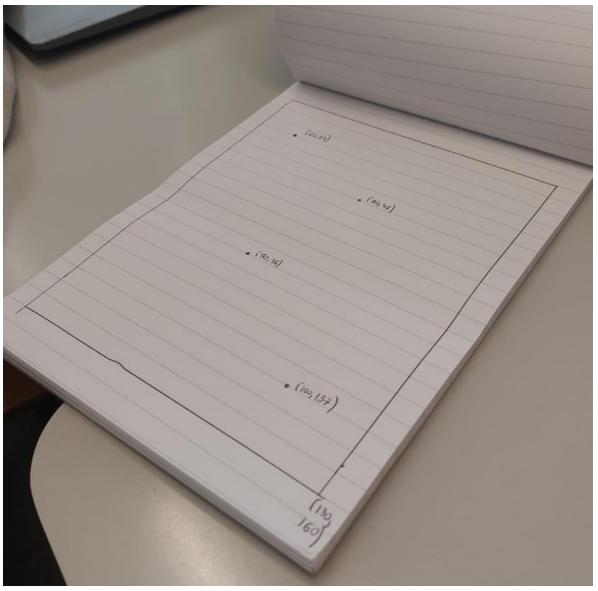


SIFT and Lowe’s test

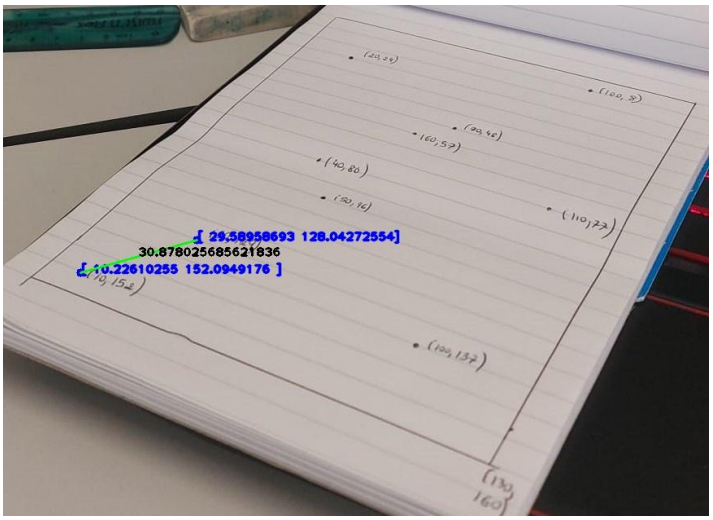
Problem 2



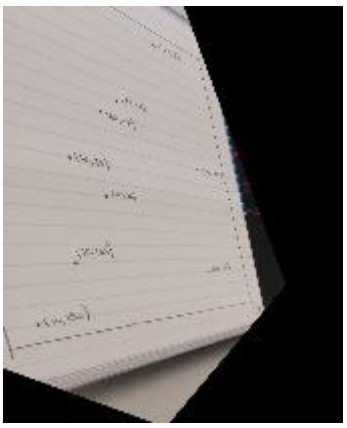
Initial 10 point plane image



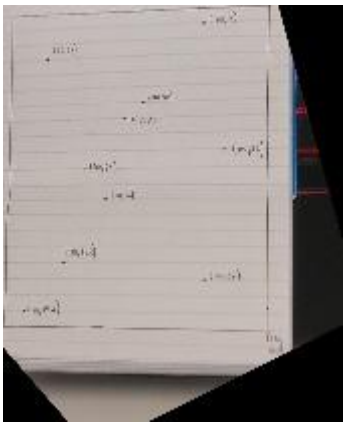
Initial 4 point plane image



Distance measured between two points

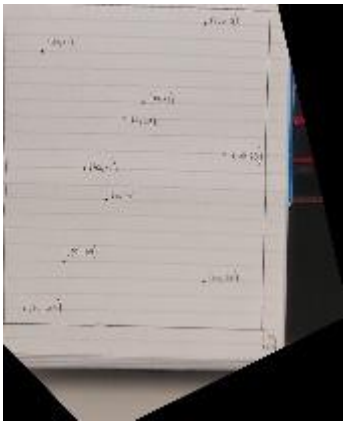


Warped perspective with 2 wrong correspondences (no RANSAC)



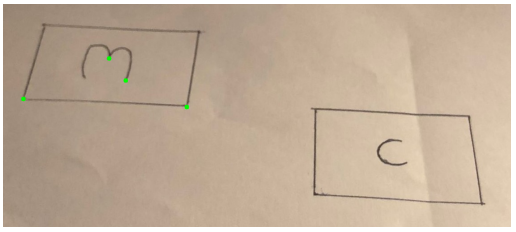
Warped perspective with 2 wrong correspondences (RANSAC)

Warped perspective with 4 wrong correspondences (no RANSAC)

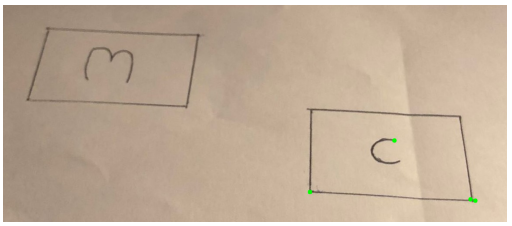


Warped perspective with 4 wrong correspondences (RANSAC)

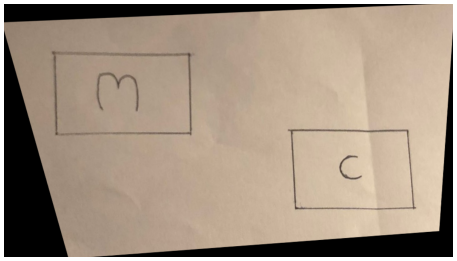
Problem 3



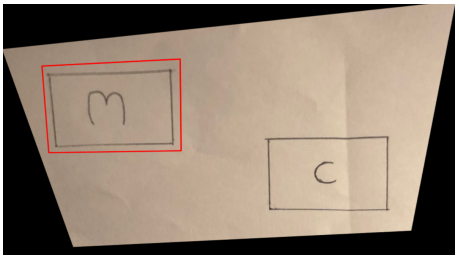
Results of FAST corner detection for for second half of the image



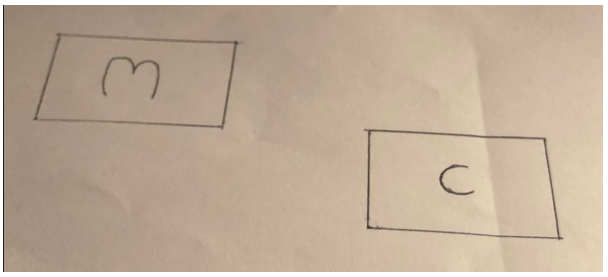
Results of FAST corner detection the first half of the image



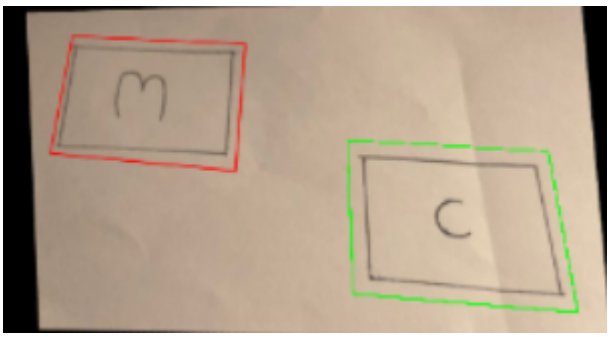
Result after the first warp perspective



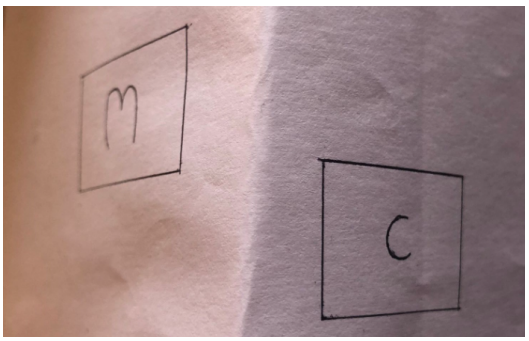
Result after the second warp perspective



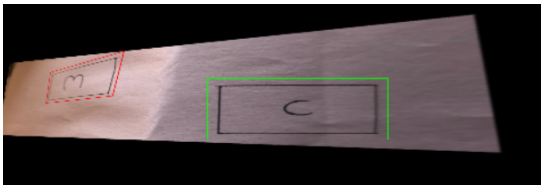
Initial testing image



Final results results



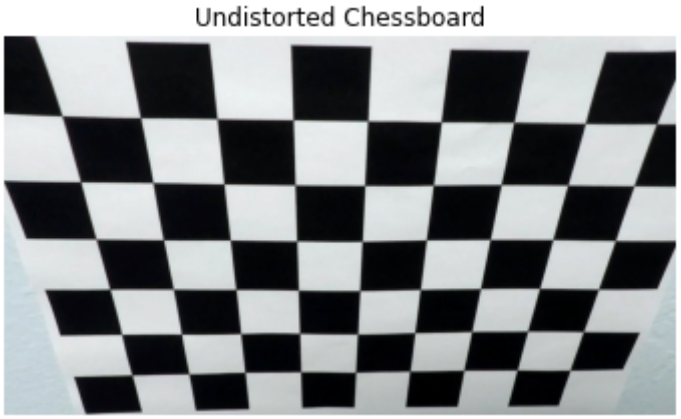
Original second image with markers in different 3D planes



Result without inverse homography

Problem 4

Distorted ChessBoard

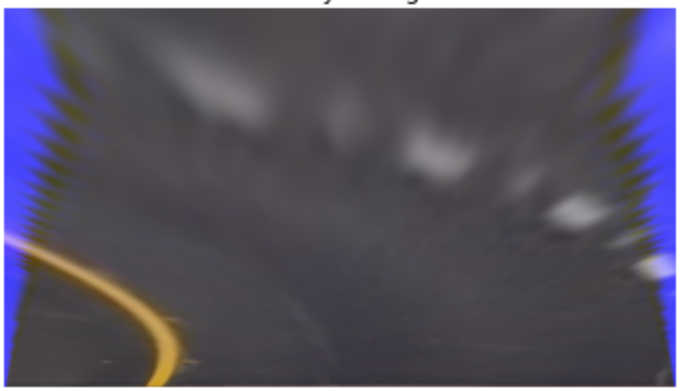


Undistorted Chessboard

Distorted Road

Undistorted Road

Curved Road



Curved Road after Transformation