Laboratory Report II
NCC-based segmentation and Harris corner detection

DIBRIS, Robotics Engineering, 2021
University of Genoa

Bauyrzhan Zhakanov and Jabrail Chumakov

22 November 2021

Abstract

This work is devoted to the segmentation and identification of features in different images by using the Normalized cross-correlation (NCC) and Harris corner detection methods. These techniques were used in order to identify certain regions in the image and highlight them. In this experiment, it was revealed that the selected windows and their size in the images affect the computation time and accuracy of detection in case using the NCC technique. Six different pictures taken from the small video fragment were compared with each other, where the main object of comparison was the red and black cars. Furthermore, it was experimentally proved that removing objects that are less than a certain number of pixels in the image can reduce the number of detected corners when using the Harris corner detection method.

1 Introduction

As a rule, an image is consists of individual pixels. Considering a black and white image (BW), it can be seen that it consists of white and black pixels, which increase from 0 to 1, where 1 is the brightness of the pixel with the maximum value. In addition, each pixel has its own specific position on the image matrix.

In the first part of the experiment, the NCC method was used on a pre-selected patch of the image with the red and black cars. 6 images were imported and compared with the patch to find a similar fragment with a car in all images. Following that, the aim was to place a red rectangular box around the center of the window. A similar procedure was done with the black car, but with different patch sizes, which affected the script execution time and the accuracy of the detection.

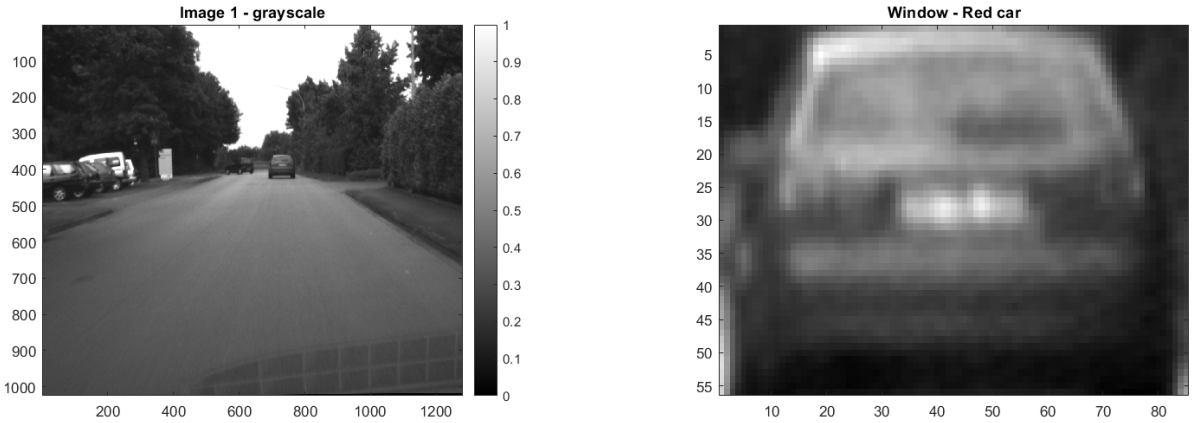
In the second part of the experiment, the Harris corner detection technique was used, which reveals the corners on the image. The accuracy of detection can be adjusted inside the script by reducing the pixel size, which should be taken into account when analyzing the image.

The entire scripts were written by using MATLAB, and the images were displayed through the figure, subplot, and imagesc() functions.

2 NCC-based segmentation

2.1 Red car

In this experiment, there were 6 RGB images, which were imported into MATLAB and converted to the grayscale images as in Figure 1 (a), since otherwise, the NCC-based segmentation would not work.



(a) Grayscale image 1

(b) Window around red car

Figure 1: Grayscale image 1 and cropped area with car

A small area around the red car was chosen as a window (Figure 1(b), which will later be compared with other images to find similarities. By using the normxcorr2 function, the segmented image can be obtained. It is worth noting that after applying NCC, the image size increases slightly and the output image becomes blurry, which is illustrated in Figure 2.

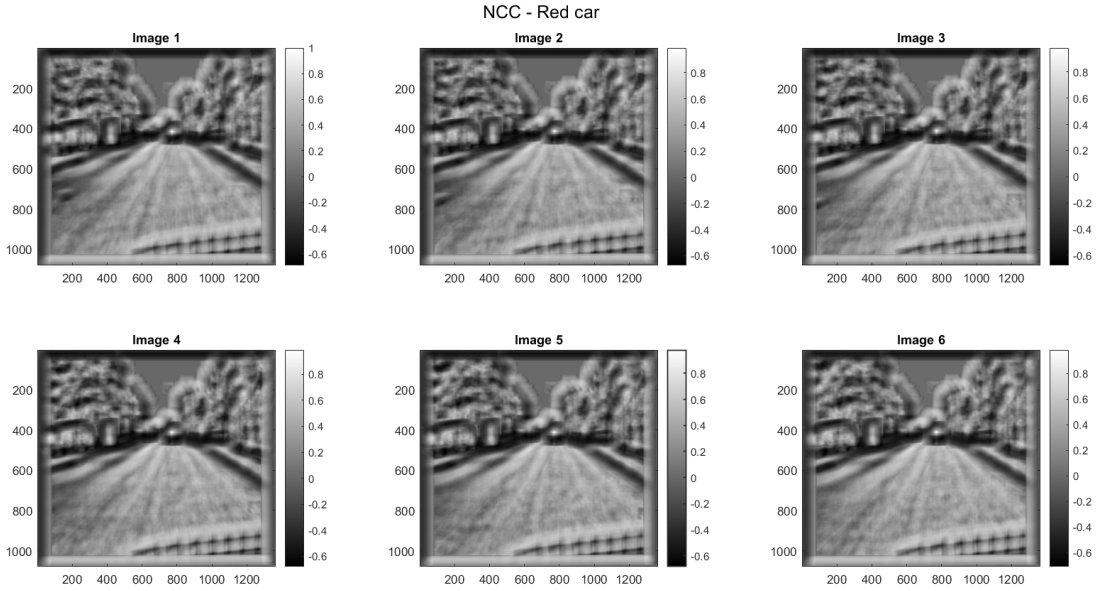


Figure 2: NCC segmentation of all images with red car

In the case of provided images, their size increased from 1024 x 1280 to 1079 x 1370. Furthermore, it can be seen that in the red car area, or in other words, the previously selected window,

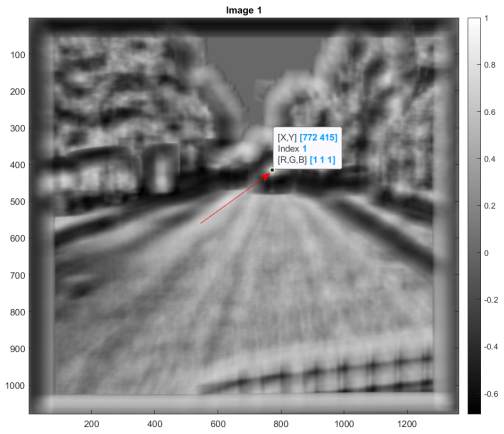


Figure 3: Position of the maximum of the score map - Red car

a white dot has appeared. This means that the NCC has correctly selected the patch and the brightest pixel in the image is located at the window's center which can be seen from Figure 3.

It allows the user to convert the NCC image to BW and obtain only a selected blob around the red car. As it was previously mentioned, the position of the white pixel is the center of the window that has been previously selected. In this regard, it is possible to trace the movement of the red car by placing a red box around it, by having the x and y position value of the white pixel from the NCC image, which is illustrated in Figure 4.

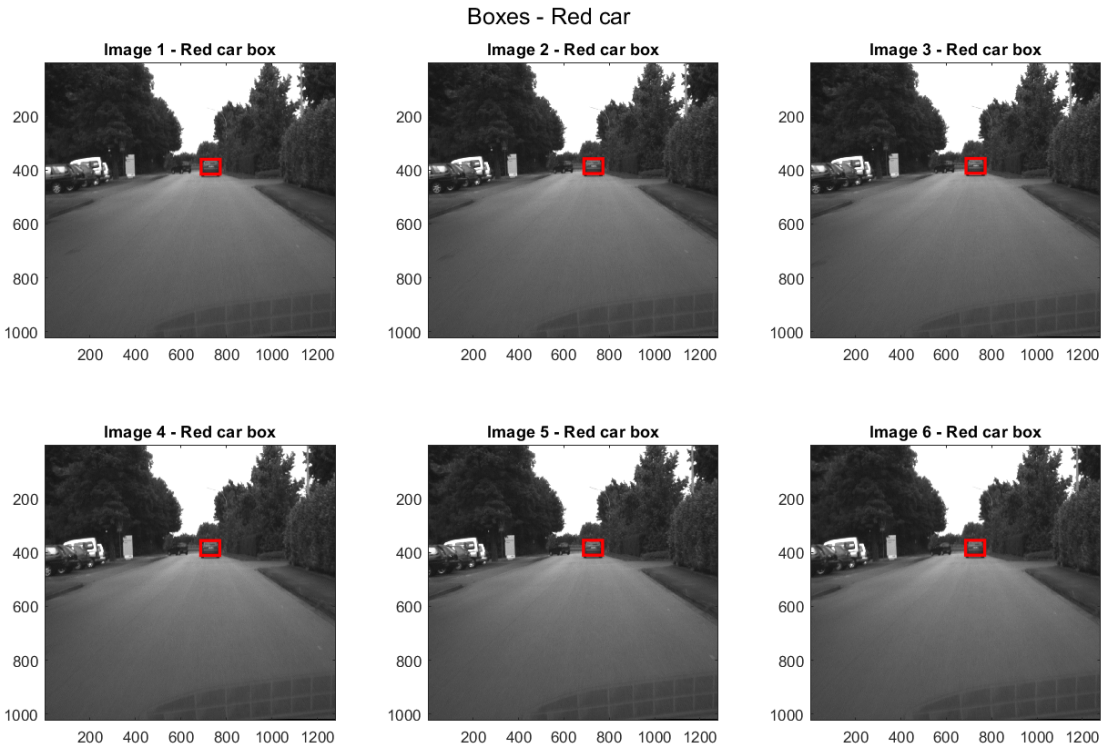


Figure 4: Boxes around red car window

2.2 Dark car

By using the same approach, it required to repeat the experiment by selecting the area around the black car. From Figure 5, it can be seen that the image quality is much worse since the black car is farther away than the red one.

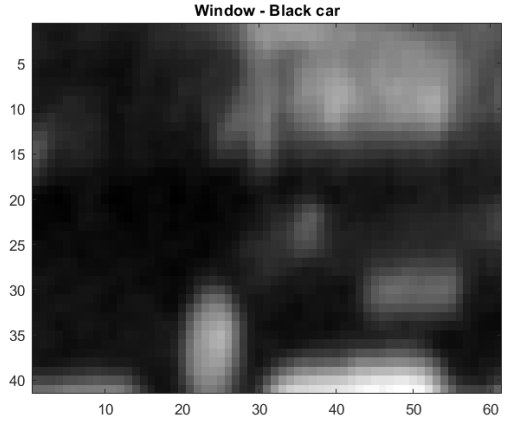


Figure 5: Window around black car

In addition, the pixels around the red car are sharper, allowing for better analysis of that area. Based on the results of the NCC, it can be seen that despite the poor image quality in that region, the NCC correctly identified the patch (Figure 6).

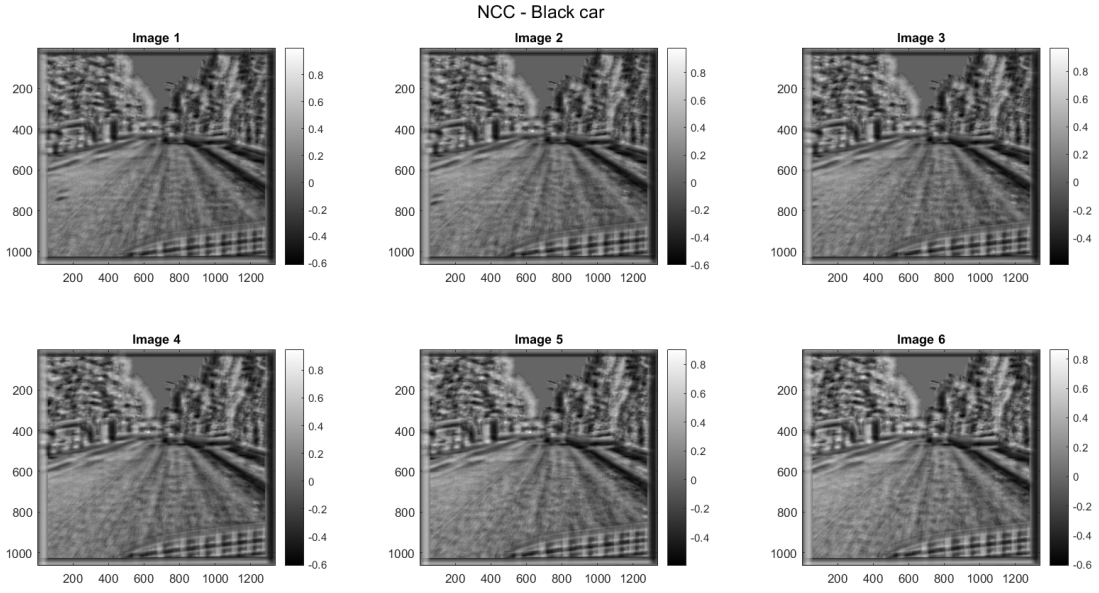


Figure 6: NCC segmentation of all images with black car

From the Figure 7 the location of the brightest pixel in a black car image can be obtained.

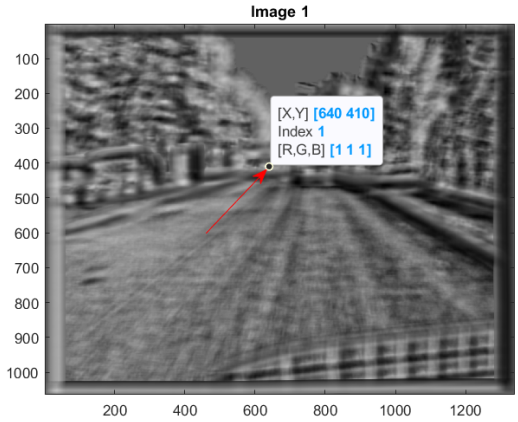


Figure 7: Position of the maximum of the score map - Black car

In addition, the movement of the black car can be tracked by placing a red rectangle around its area based on the NCC results which are illustrated in Figure 8.

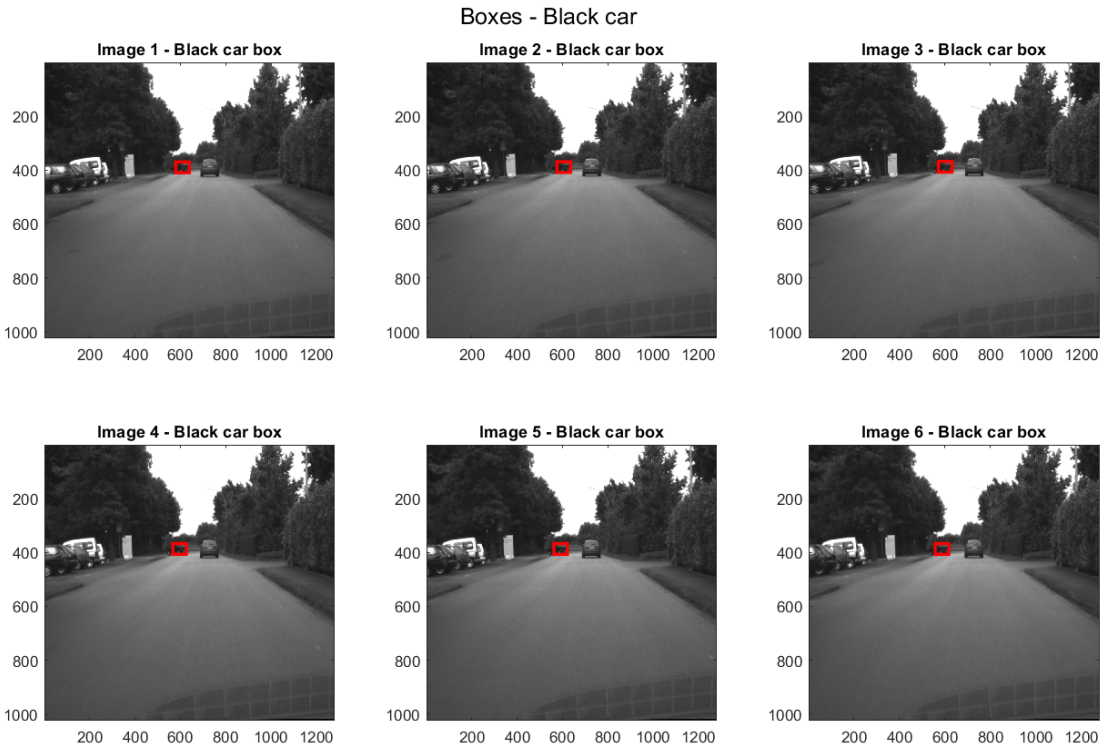
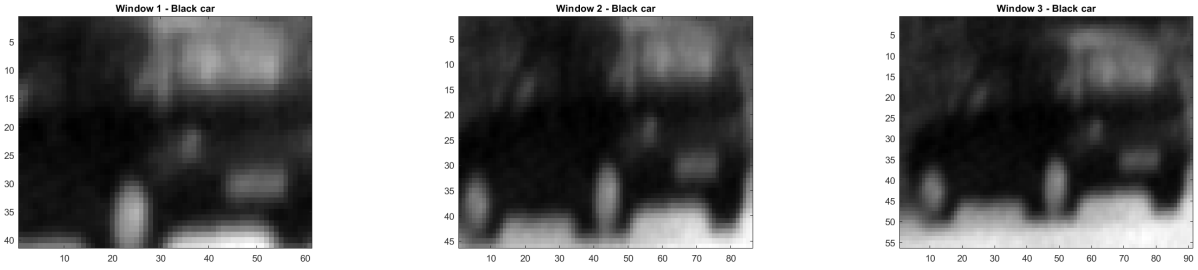


Figure 8: Boxes around black car window

2.3 Dark car with 3 different window size

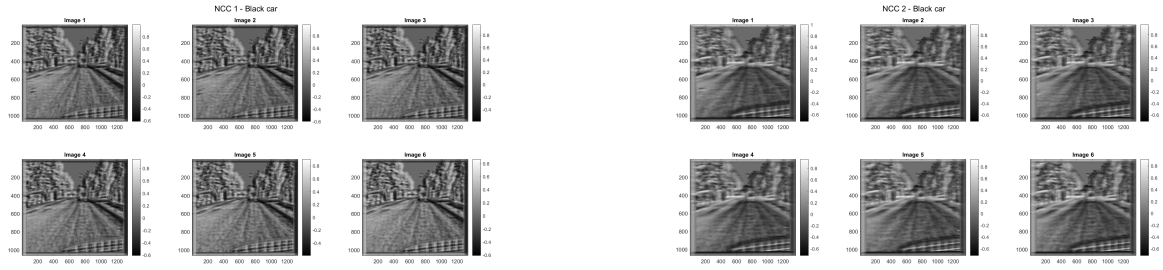
The same experiment was done with the area of the black car, but with different window sizes. Three different windows were considered, from small to large (Figure 9).



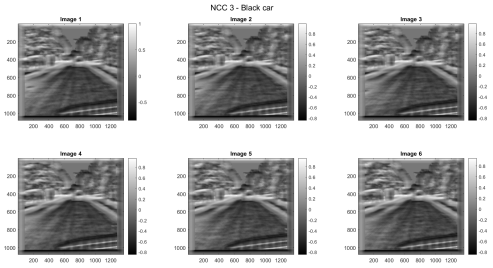
(a) Window 1 - black car (b) Window 2 - black car (c) Window 3 - black car

Figure 9: Different windows for black car

During the experiment, it was revealed by using 'tic' and 'toc' that the time required to run the script took about 1.622 s, 1.756 s, and 2.48 s, respectively. In other words, the larger the area was considered, the longer it took for MATLAB to run the script. Furthermore, it was also revealed that the larger the window size was, the brighter the pixels after applying NCC were. This can be seen in Figure 10.



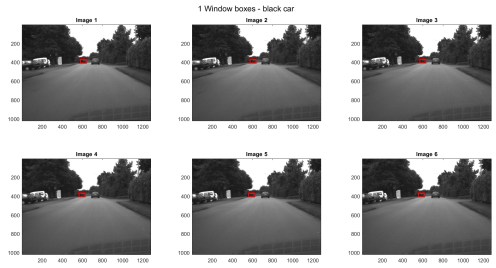
(a) Window 1 - NCC (b) Window 2 - NCC



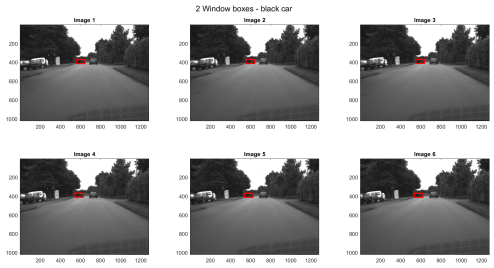
(c) Window 3 - NCC

Figure 10: NCC with different windows for black car

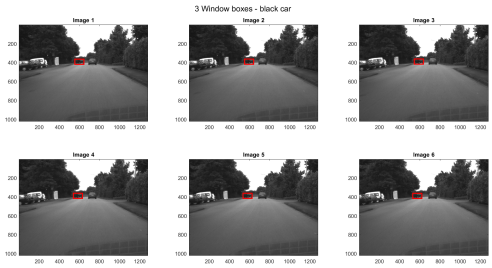
In addition, it can be seen that the image is tracked equally well on all 3 windows, which can be seen in Figure 11. However, the best results were achieved when the blob was more distinguishable, as in window 1. Thus, it can be concluded that the size of the window affects the time of script execution, as well as the accuracy of the detection.



(a) Window 1 - Box



(b) Window 2 - Box



(c) Window 3 - Box

Figure 11: Boxes on different windows for black car

2.4 Comparing results with Lab 4

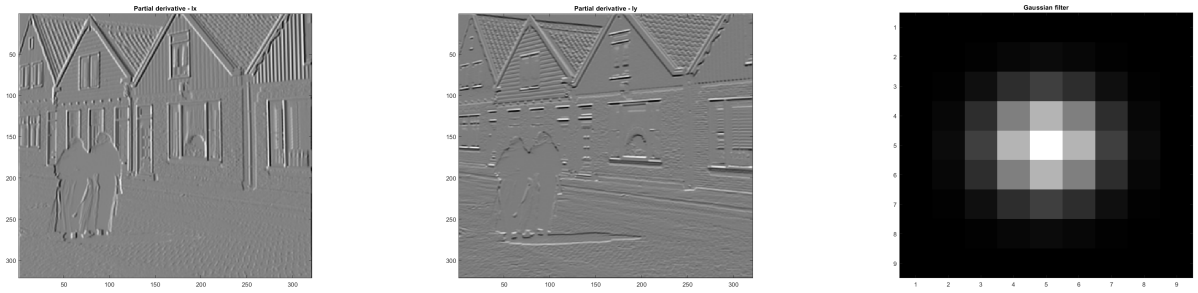
Comparing the results of this experiment with laboratory work 4, it can be noted that this method is more effective since it immediately gives the location of the blobs. Additionally, this technique is more time-efficient and the results are much more accurate. The fact is that the method in Lab 4 gives a number of spread white pixels that need to be corrected and aligned with your target object later, while due to the NCC, this can be done much easier and faster.

3 Harris corner detection

In this experiment, the main task was to find the corners by using the Harris corner detection technique. Initially, there was the grayscale image, as shown in Figure 12.

Figure 12: Original image

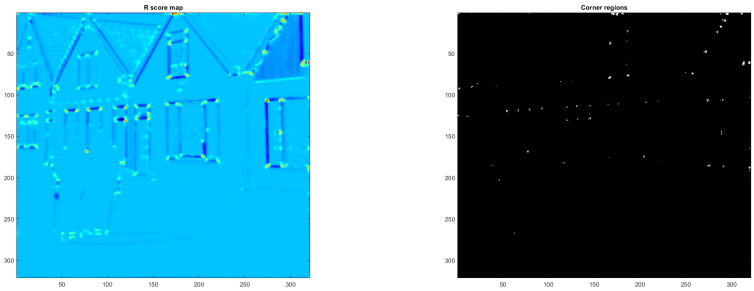
After that, after computing the x and y derivatives of the image, the partial derivatives of the image were found, in addition to the applied Gaussian filter, which can be seen in Figure 13.



(a) Ix (b) Iy (c) Gaussian filter

Figure 13: Different windows for black car

Next, the R-score map and corner regions were obtained as shown in Figure 14.



(a) R-score map (b) Corner regions

Figure 14: R-score map and corner regions

Finally, it can be seen in Figure 15 that there is an image with centroids in the detected corners. Furthermore, it is worth noting that users can weed out pixels below a certain size, which will make it possible to get rid of unnecessary corners.

Figure 15: Detected corners