Computer Vision: Lab5

Mamoru Ota Ana Luísa Campos e Sampaio Melo

1 NCC-based Segmentation

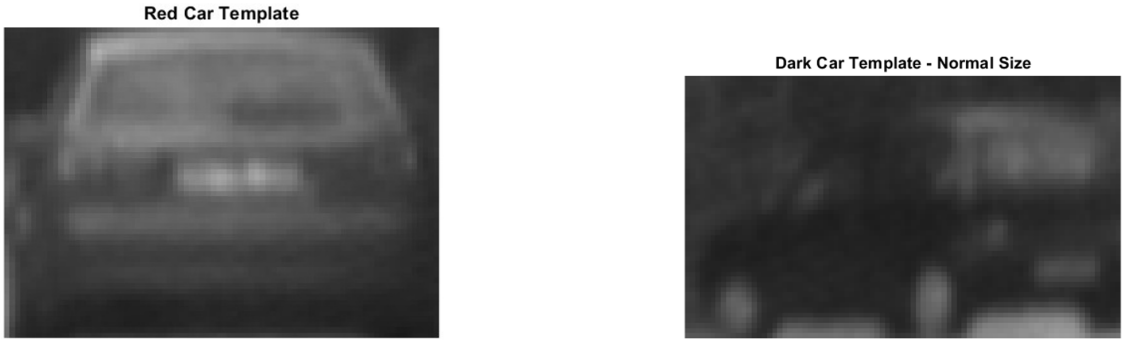
The objective of this exercise was to detect two specific vehicles: a red car and a dark car, within a sequence of six grayscale images using Normalized Cross-Correlation (NCC). This technique identifies regions in an image that are most similar to a given template, allowing for the localization of objects based on their intensity patterns. The task involved selecting templates for both the red and dark cars in the first image, applying NCC across all six images, and marking the detected regions with bounding boxes.

1.1 Template Selection

Template matching involves selecting a small representative patch (template) from an image and convolving it with other images to find regions that closely match the template. The similarity is measured using cross-correlation, and the Normalized Cross-Correlation (NCC) improves this by counteracting intensity variations through normalization - subtracting the mean and dividing by the standard deviation of the intensities.

For the red car, the template size was 80x57 pixels, starting at coordinates (689, 360) for the top-left corner (Figure 1a). For the dark car, starting at top-left corner coordinates (555, 362), the template chosen had 85x51 pixels (Figure 1b).

The templates for the red and dark cars were successfully extracted from the first grayscale image, as seen in Figure 1.



(a) Red Car Template (b) Black Car Template

Figure 1: Templates used to identify the two cars

1.2 Normalised Cross-Correlation

Normalized Cross-Correlation (NCC) is a technique used to measure the similarity between a template and regions of the target image. It normalizes intensity values by subtracting the mean and dividing by the standard deviation, making it robust against changes in brightness or contrast. The NCC score map was computed using MATLAB's *normxcorr2* function.

In this lab, for each image, the peak of the score map was identified as the location with the highest similarity to the template and the coordinates of the peak were adjusted to the original image's coordinate system by accounting for the padding introduced during the correlation. With that, a bounding box was drawn around the detected region, and the maximum score was marked with a red star (*).

The NCC-based detection accurately localized the red car and the dark car across the six images, as shown in Figure 2 and Figure 3.

(a) Template Detection in Image 1

(b) Template Detection in Image 3

(c) Template Detection in Image 6

Figure 2: Detection of the Red Car in three images

(a) Template Detection in Image 1

(b) Template Detection in Image 3

(c) Template Detection in Image 6

Figure 3: Detection of the Dark Car in three images

1.3 Comparison to Lab4

In Lab4, a color-based segmentation approach was used to detect the same two vehicles by analyzing the Hue component of the HSV color space. This method involved selecting thresholds based on the mean and standard deviation of the Hue values for the regions of interest. While effective for the red car due to its distinct color, the method struggled to segment the dark car consistently across the sequence, as seen in Figure 4, which is the last image in the sequence. Variations in lighting and shadows caused the Hue values to deviate, leading to incomplete segmentation for the dark car in some images.

In contrast, the NCC-based method demonstrated superior performance, detecting both vehicles consistently in all six images. By normalizing intensity patterns, NCC was less sensitive to lighting variations and accurately localized the dark car even in challenging conditions. For the red car, both methods performed well.

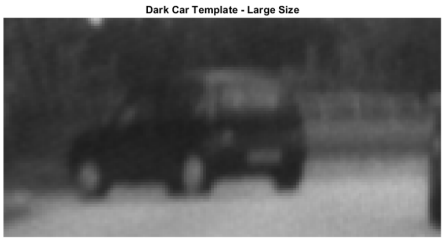
1.4 Analysis of Different Template Sizes

To evaluate the effect of template size on the performance of NCC-based detection, three different window sizes were tested for tracking the dark car across the six images. These sizes were defined as follows:

Figure 4: Color-based Segmentation Done in Lab4 Applied to the Last Image

- **Small Template:** 60×30 pixels, focusing on the core features of the car, presented in Figure 5a.
- **Normal Template:** 85×51 pixels, capturing the entire car (baseline size), as seen before in Figure 1b.
- **Large Template:** 160×80 pixels, including the car and its surrounding context, shown in Figure 5b.

(a) Small Dark Car Template



(b) Large Dark Car Template

Figure 5: Different Templates Used to Identify the Dark Car

All three template sizes successfully tracked the dark car across the sequence of images, as shown in Figure 6, demonstrating the robustness of the NCC-based method. However, significant differences in computation time were observed. The small and normal templates took approximately the same computation time, with only a few milliseconds of difference. The large template, however, took a bit longer.

In terms of accuracy, all three templates were able to follow the car. However, the small template showed small deviations.

These results may be due to the smaller window in the small template making it more sensitive to noise or small changes in the car's appearance, and, in the case of the large template, the window captured more contextual information, such as shadows or background features near the car, this additional information led to longer computation times.

2 Harris corner detection

The Harris corner detector is an algorithm designed to detect corners in an image, distinguishing them from edges and flat regions. This algorithm calculates the corner response value (R-value) using the image's derivative information and structure tensor. Regions with high response values are identified as corners.

(a) Small Template in the First Picture

(b) Normal Template in the First Picture

(c) Large Template in the First Picture

(d) Small Template in the Last Picture

(e) Normal Template in the Last Picture

(f) Large Template in the Last Picture

Figure 6: The Different Size Templates Applied to the Same Images

I implemented the Harris corner detector and applied it on the "image i235.png". I explain the implementation steps and obtained results.

2.1 Implementation

First, using the Sobel filter, the gradients of the image in the x-direction and y-direction (I_x and I_y) are calculated and visualized. These gradients quantify the rate of change in the image. The products of the gradients (I_x^2, I_y^2, and I_xI_y) are calculated, and a Gaussian filter is applied and visualized. It is for calculating the Gaussian-smoothed results of gradients in the x-direction, y-direction, and x-y direction (S_{x^2}, S_{y^2}, and S_{xy}). This step reduces the influence of noise.

Then, the R-values for each pixel are calculated using the following formula:

$$R = \det(\mathbf{M}) - k \cdot (\text{trace}(\mathbf{M}))^2$$

where $\mathbf{M}$ is the structure tensor defined as:

$$\mathbf{M} = \begin{bmatrix} S_{x^2} & S_{xy} \\ S_{xy} & S_{y^2} \end{bmatrix}$$

A threshold is set at 30% of the maximum R-value. Regions exceeding this threshold are identified as corners and R score map and the corner regions are visualized. The 'regionprops' function is used to calculate the centroids of the corner regions, providing the precise locations of the detected corners. The detected corners are overlaid on the original image for visualization.

2.2 Results and Discussion

Figure 7 shows the detected corners overlapped on the image. The implementation confirmed that the Harris corner detector can almost accurately identify corners in the

image. The application of Gaussian smoothing effectively reduced noise, improving the accuracy of the corner response value calculations. Additionally, setting an appropriate threshold for the R-value allowed control over the precision of corner detection. However, some corners are not detected, and conversely, some parts that are not corners are detected as corners. This is thought to be due to the corner detection threshold, and it was suggested that an appropriate threshold must be specified for the Harris corner detector. Also, because it has a strong response near edges, it may erroneously detect edge parts as corners. In addition, although the Harris corner detector is rotation invariant, it is not scale invariant, so in some cases other methods such as SIFT or SURF are necessary.

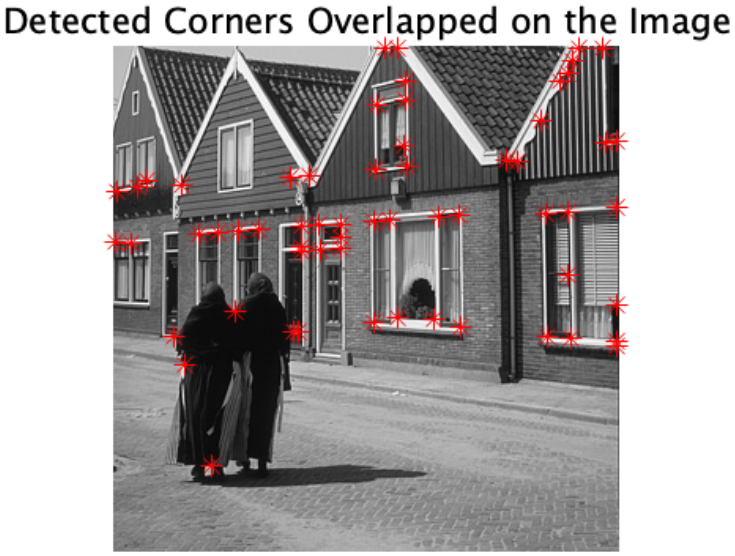


Figure 7: Detected Corners Overlapped on the Image.