Task 3: Recognize a key using pattern matching

Javier Galindos (214373IV) Omar El Nahhas (214316IV)

11/10/2021

Abstract

The third task for the Machine Vision course (EEM0040) is to recognize a specific key using pattern matching. The presented work is developed in both Python (using OpenCV2) and MATLAB.

1 Methodology

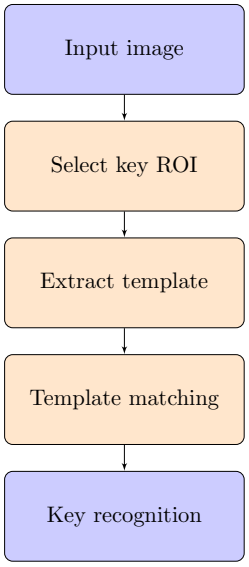


Figure 1: Flowchart of the image processing methodology for key recognition.

In this assignment, 10 keys of similar colour and overall shapes are presented, see Fig. 2. The goal is to identify a key, and then recognizing the exact same key as it is put in a different location. Pattern matching with a single template of the unique part of the key is applied to identify the relevant key anywhere in the picture. The camera is situated directly above the image. This avoids to correct for perspective, as template matching in its basic form is very sensible to varying scales and angles with respect to the template image. The overall flow of image processing for this task is shown in Fig. 1.

Figure 2: Input image of the keys.

1.1 Template selection

To start off, a key is selected in a live environment. After choosing the key and extracting the image, a rectangle can be drawn on the image using OpenCV2's selectROI and MATLAB's Image Viewer. Then, the rectangle is automatically extracted in Python, and needs to be manually saved to the workspace in MATLAB to be used for the next step. OpenCV2's template selection process is visualised in Fig. 3.

1.2 Template matching

Both in Python and Matlab, Normalized Cross-Correlation (NCC) was used to identify the locations with the highest probabilities of the template matching with the to-be-analysed image. In Python, template matching results in start and end coordinates of a rectangle. Matlab returns single coordinates, where the highest probabilities have a higher pixel intensity. Both implementations allowed for template matching with all three RGB channels, which makes object detection increasingly more accurate than having to choose one channel to differentiate from. On the professor's suggestion, HSV channels were also considered. Again, one HSV channel could not compete with the level of differentiation that three RGB channels could offer, hence this one-channel approach was rejected.

2 Results

The experiments performed during this lab show that the template matching algorithm is capable of identifying the correct key everywhere on the image while the orientation of the keys is maintained

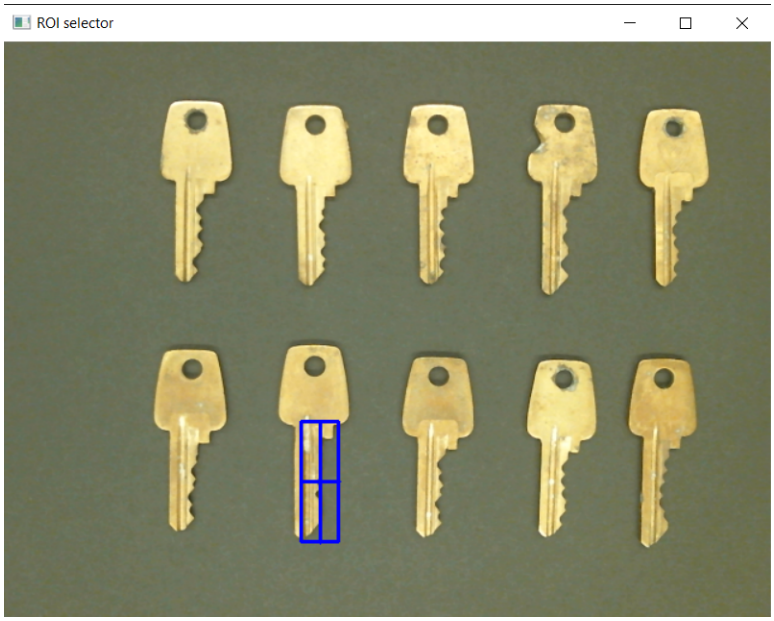
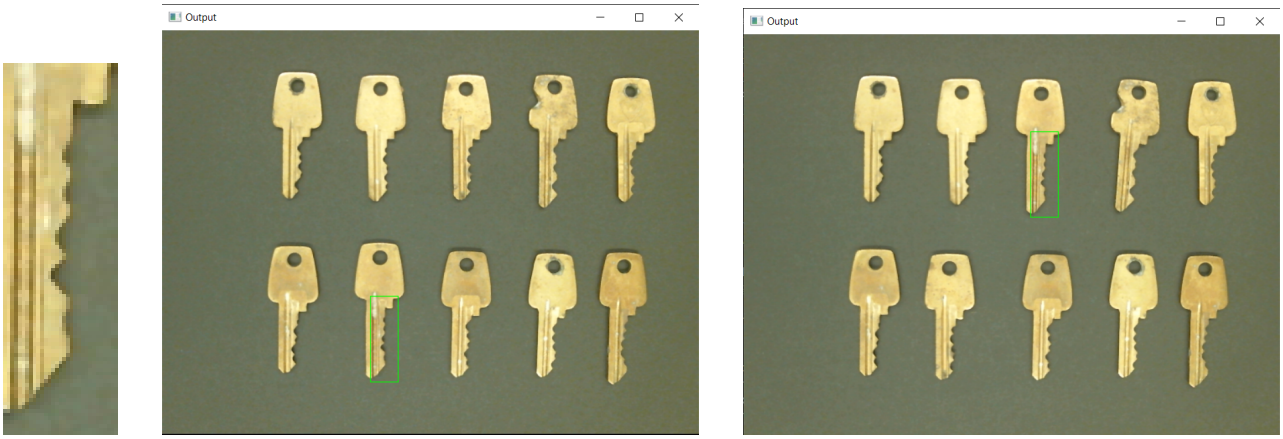


Figure 3: Selecting the template ROI using OpenCV.



(a) ROI. (b) Original position. (c) New position of key.

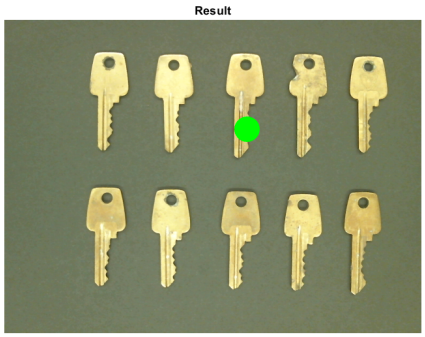
Figure 4: Template matching for different positions (Python).

with respect to the template. The results in two different positions are shown in Fig. 4 for the Python environment and in Fig. 5 for MATLAB environment.

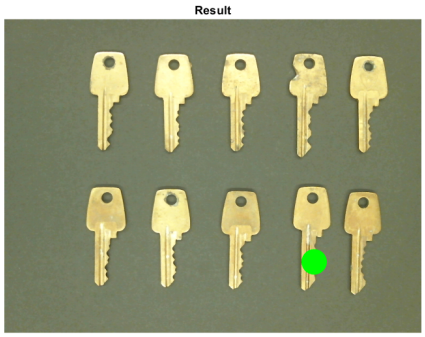
3 Discussion

The limitations of the template matching approach to identify a specific key are related to positioning of the keys. When the key is slightly rotated with respect to the original template orientation (see Fig. 6), the algorithm is not able to identify the correct key. One possible solution is to perform

(a) ROI.



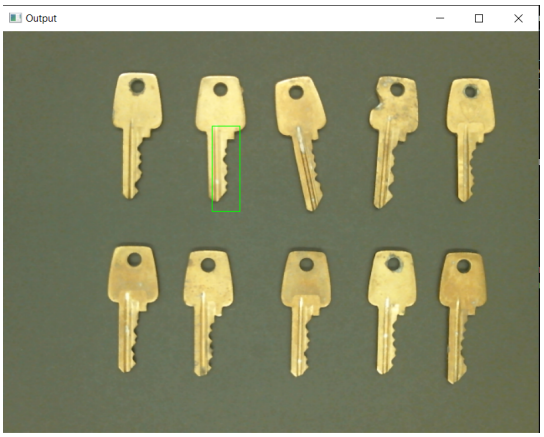
(b) Original position.



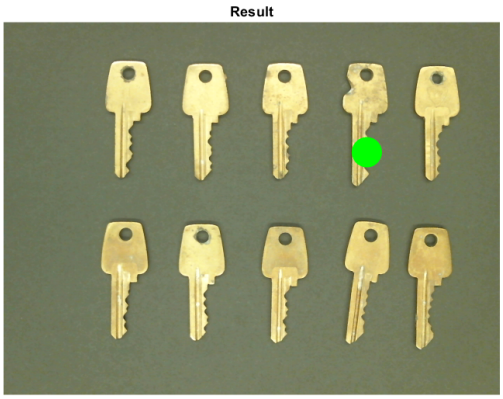
(c) New position of key.

Figure 5: Template matching for different positions (MATLAB).

template matching multiple times while rotating the key template. The algorithm checks if the images matches with all possible rotations of the template and returns the one with the highest probability. But in practise, this is computationally inefficient and is not recommended to be applied in a real-time application. To make the pattern matching more robust, deep-learning networks can be used to recognize the keys at different orientations, sizes and lighting conditions.



(a) Top-middle key is slightly rotated (Python).



(b) Fourth key on the bottom is slighlty rotated (MATLAB).

Figure 6: Template matching fails when the to-be-detected key is at an angle with respect to the selected key template.