Object Instance Detection

Simone Vaccari 915222
Davide Vettore 868855

1. Objective:

The objective of this activity consists in precisely localize an object of interest inside an image by employing two different approaches: firstly, an intuitive template matching sliding window technique, and secondly, a keypoints-based method.

2. Introduction:

The reference object we're interested in is an elephant statue, while the scene under examination is a cluttered desk.

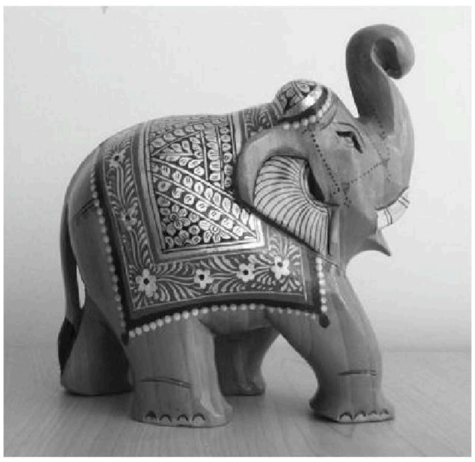
Figure 1: Reference object

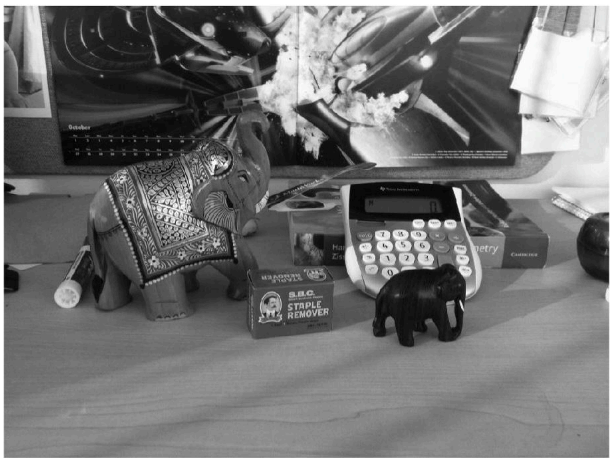
Figure 2: Scene image

Initially, we implement a simplified version of intuitive **Template Matching**. This method involves computing the *Sum of Squared Differences* between two patches to discern similarities. However, this approach is computationally inefficient, as it requires evaluating all possible combinations of factors such as location, scale, rotation, lighting, and perspective transformation. To expedite the process, we will forego exploring multiple scales, rotations, etc., and instead manually resize the reference object to align it with the target in the scene image.

Following this, we turn to a **Keypoints-Based** approach. This method operates under the assumption that keypoints retain their relevance in the new scene image. Consequently, the algorithm does not necessitate retraining for each new application, which makes it particularly suitable for real-time implementations. Specifically, we will employ the *Speeded Up Robust Features (SURF)* algorithm, which represents a swifter and recently introduced alternative to the once-prominent *Scale Invariant Feature Transform (SIFT).*

3. Procedure:

The application of the simplified template matching approach is straightforward. The manually computed scale factor is set to 0.9 in such a way that, by resizing the template image with it, the object will appear at the same scale in both images. Additionally, the step parameter is configured to 5, meaning that the sliding window will move five pixels at a time both horizontally and vertically. A smaller step size would result in a more exhaustive search but would also increase the computational cost.

On the other hand, the keypoints-based approach has to follow a more complex pipeline:

1. Keypoint detection: implemented with the *SURF* algorithm and a *metric threshold* decreased to 800 (from the default value of 1000), in order to return more keypoints. By doing so, we provide a larger set of possible matches, hoping to increase the probability of a correct match;
2. Keypoint description;
3. Features matching: implemented with a *match treshold* set to 40, representing the percent of the distance from a perfect match, and a *max ratio* set to 0.8, with which ambiguous matches are rejected. By increasing it from the default value of 0.6, we were able to obtain a higher number of matches;
4. Geometric consistency check: implemented with an affine *transform type*, and a 80% *confidence* of finding the maximum number of inliers;
5. Bounding: both with a box and a 20-edges polygon;

3. Results:

The result provided by the sliding window approach can be seen in Figure 3. Blue spots indicate regions with minimal squared differences, while yellow spots denote areas with higher squared differences.

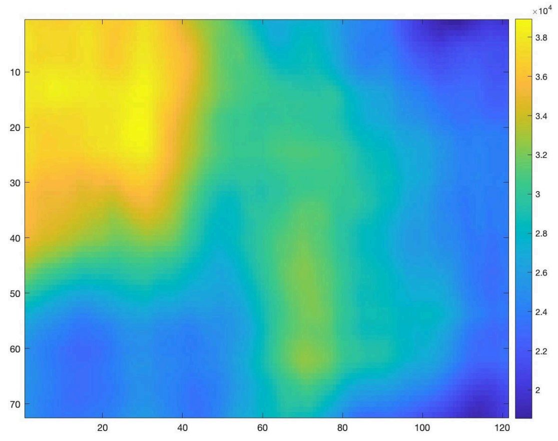

Figure 3: Sliding window result

We can now observe the results of the SURF keypoints-based approach. The first output shows the keypoints detected in both the reference object and the scene image, displayed in Figure 4 and Figure 5, respectively.

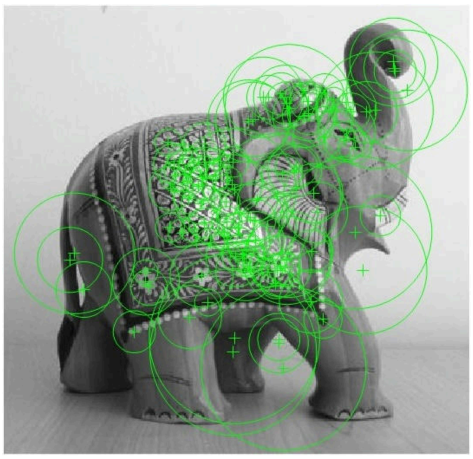
Figure 4: Reference object keypoints (100 shown)

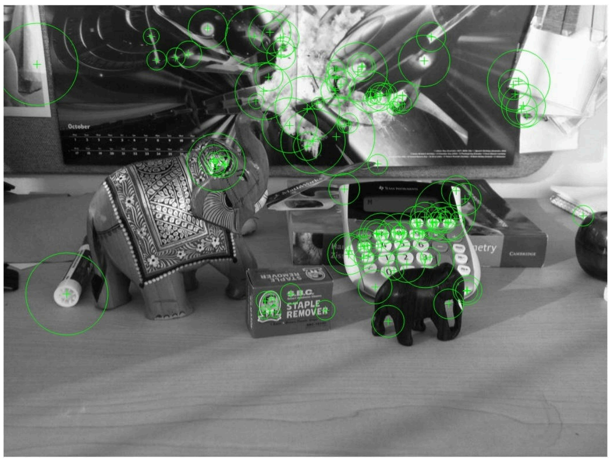
Figure 5: Scene image keypoints (100 shown)

In Figure 6 we show all the matches found between the keypoints of our template and the ones of the scene image. As we can see, not all matches are correct, and this would compromise the object detection results if we didn't apply the geometric consistency check, which allows us to remain with only a few correctly matched features, as shown in Figure 7.

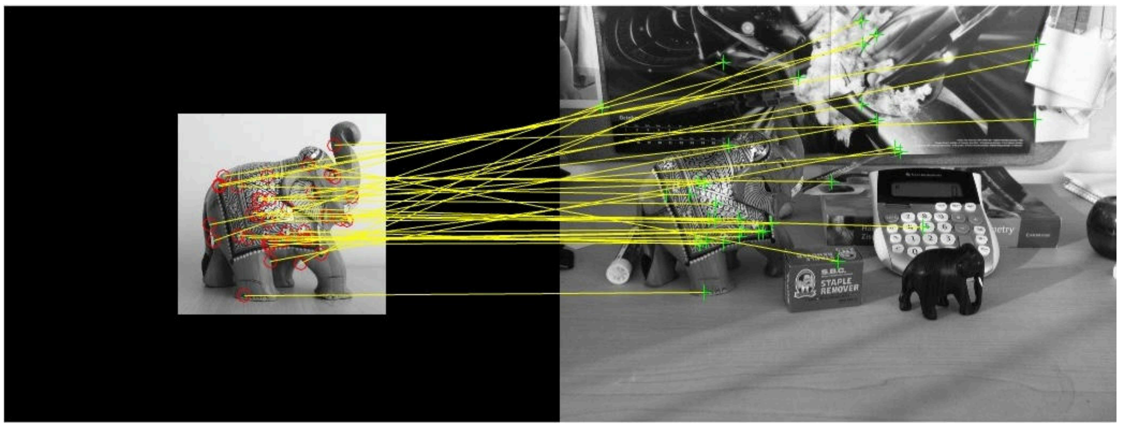
Figure 6: Feature matching

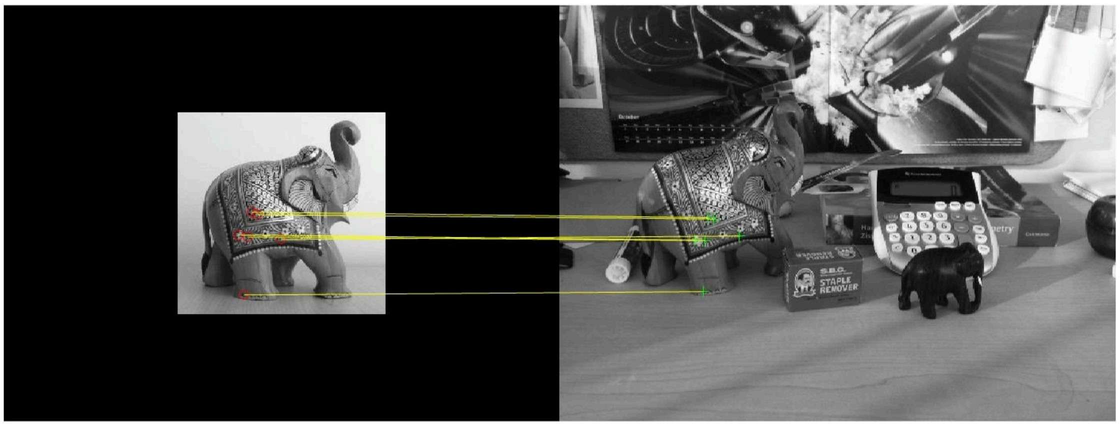
Figure 7: Feature matching after geometric consistency check

Finally, Figures 8 and 9 show how the template object was detected in the scene image using a bounding box (a quadrilateral) first, and then a 20-edge polygon, defined in such a way that it follows the profile of the elephant more precisely.

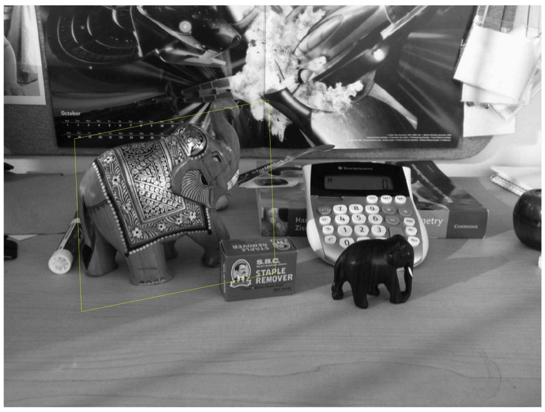
Figure 8: Object bounding box

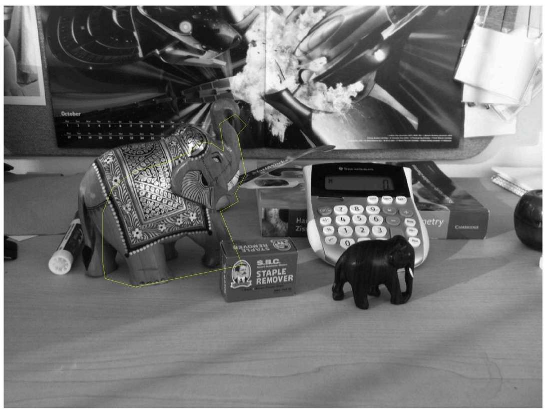
Figure 9: Object bounding with 20-edges polygon

5. Conclusions:

It is clear that the results obtained with the sliding window approach are very poor: in Figure 3 there is no clear spot of where the object is, and this results in an inconclusive search. This method turns out to be less efficient with respect to the keypoints-based approach even on the computational point of view: the execution time of the first method is 4-5 times larger than the second one's, despite having simplified it notably.

The detection achieved by the SURF technique is much more precise, even though it's not perfect. This could be due to the fact that in the scene image the elephant appears rotated with respect to the vertical axis, while in the template image it is seen from a lateral perspective.