Image Analysis and Computer Vision

F10. Template matching in depth images

Corti Greta (Personal Code: 10598667)
Chiari Giuseppe (Personal Code: 10576799)
Shalby Hazem (Personal Code: 10596243)

POLITECNICO
MILANO 1863

1 Problem formulation

Template matching is a very practical technique for finding all the occurrences of a known target in a given query image. Deep-learning techniques are not very flexible, requiring re-training for new templates. Traditional computer vision techniques (e.g. SIFT + RANSAC) are more practical but require an underlying model (homography) describing the transformation between the template and its occurrences in the query image. When there are several occurrences of the same template in the scene, template matching becomes a robust multi-model fitting problem, which requires disambiguating among multiple spurious matches.

The goal of this project is to expand a template detection pipeline based on images only, to RGB-D data. Point clouds should be leveraged to constrain the homography search in regions belonging to the same planar surface in the scene. Another research direction would be to adaptively identify which areas of the target image to search, which is a very relevant problem when the number of key points is large (e.g. a few hundred) by exploiting the 3D data of the scene.

The specific case tackled by the following project is recognizing squared boxes in the scene, such as those found in a supermarket.

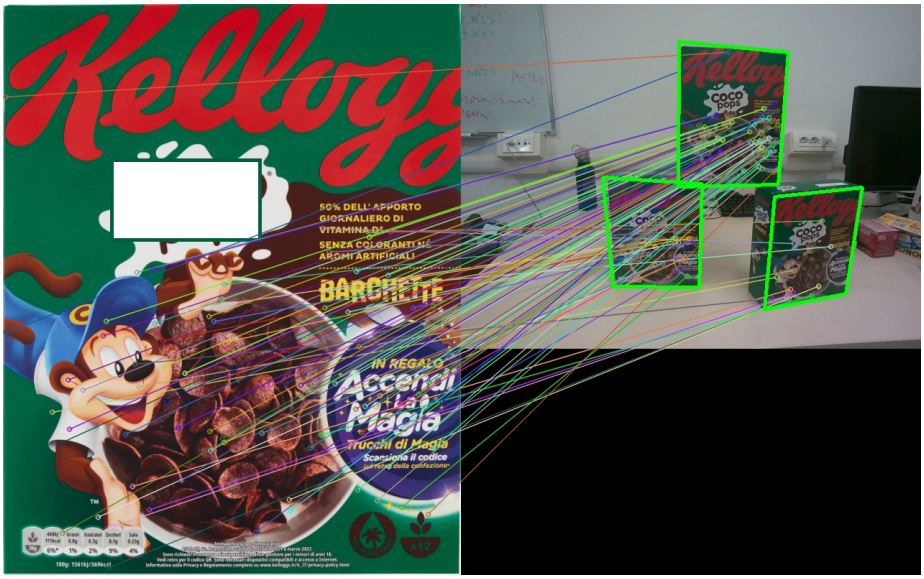


Figure 1: Template Matching example

2 State of the art

The Template Matching method simply scans the complete image and slides over the template image across the full image and identifies the matching area

as a heatmap: this method is mostly static and it gives very poor results if the full image has slight deviations like the direction change, image intensity change, scale changes. Hence, it cannot be used in real-world applications.

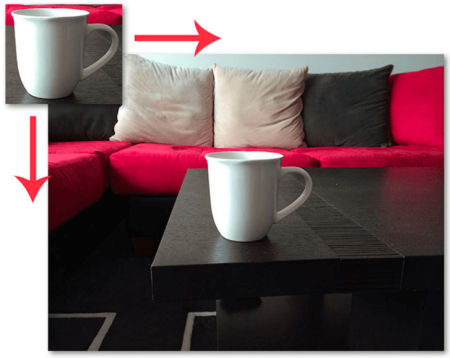


Figure 2: The easiest way to do template matching

To overcome the above difficulties of the Template Matching method, key point detectors are the solution. Key points may be defined as small areas of an image having some particular features, e.g. rapid changes in image brightness domain, like corners. A set of detected key points and parameters describing them, known as descriptors, collect the most important features of an object and create its representation in a compact structure of a template. In literature, different feature detectors exist: Scale Invariant Feature Transform (SIFT), Speed Up Robust Feature (SURF), and Oriented FAST and Rotated BRIEF (ORB). SIFT is mainly used because it performs the best in most scenarios and the key point detectors are distributed over the image. ORB is the fastest algorithm, but the extracted features are mostly concentrated in objects at the center of the image. ORB and SURF outperform SIFT only when the angle of rotation is proportional to 90 degrees [1].

Once the features get detected and matched, only 4 matches would be enough to fit the 2D homography mapping the template to the scene's plane, but the reality is that, in any real-world case, several bad matches will occur. After a coarse cleaning of the outliers with a ratio test, an outlier detection method is still needed. In computer vision, random sample consensus (RANSAC) is used as a robust procedure to estimate the foundational matrix in stereo vision, to find if two sets of points are equal. It is a recursive method for estimating a mathematical representation from a data set that contains outliers. The RANSAC algorithm works by identifying the outsiders in a data set and approximating the wanted model using data that does not contain outliers. Given a set of features F, RANSAC algorithm works as follows:

1. Randomly selects a subset S of the original data set F.

2. Fit a model M to the points of S.

3. All the data belonging to the set F-S are then tested against M. Those points that fit M well, according to some loss function, are considered as part of the consensus set S. The estimated model M is reasonably good if sufficiently many points have been classified as part of S.

4. Afterwards, the model may be improved by re-estimating it using all members of the consensus set.

5. Repeat steps 1-4 for a fixed number of iterations.

At each iteration, it will be produced either a model which is rejected because not enough points are part of the consensus set, or a processed model together with a corresponding consensus set size.

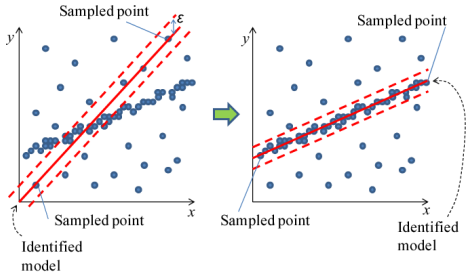


Figure 3: Conceptual figure of RANSAC

3 Solution approach

The most known algorithms that solve the template matching problem, reach the goal by performing the following operations:

- feature extraction from the template and the scene.
- feature matching and selection.
- fit the homographies via sequential Ransac.

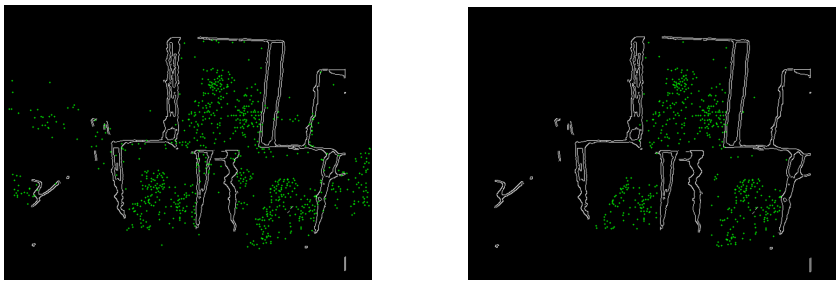
As can be easily noticed, the most vulnerable parts of the above pipeline are clearly the elaboration of the huge number of features extracted from the scene at the beginning of the process and the identification of key points to use for the computation of the homography.

The approach followed during the project development [2] was to start from the above pipeline and try to improve its performance by acting on the most vulnerable parts; in particular, the techniques implemented use filters to reduce the number of key points to examine and to select the few key points needed to fit the homography in different ways than the ones proposed by Polidori and Stagnoli [3].

In the following sections, three main techniques will be described in detail.

3.1 Canny Filter Method

To decrease the number of key points extracted by SIFT from the image, a Canny edge detection has been implemented. It is assumed that object edges can be exploited to choose the most useful key points to improve the performance of key point detection and matching [4]. On both RGB scene image and depth image a denoised function is applied a and then the Canny method. On the resulting images, Probabilistic Hough Line Transform is applied; it is more efficient implementation of the Hough Line Transform because it gives as output the extremes of the detected lines. The extracted lines from RGB image are compared with the extracted lines from the depth image so to have more accuracy and precision in line selection since some are wrong due to image noise. Thanks to these lines, probable rectangles can be created and they represent the boxes present on the scene. Having the rectangles and the key points of the scene, all the key points that are outside the rectangles can be deleted. In this way, the number of key points usually decreases by one-third of the initial total number.



(a) Key points found by SIFT (b) Key points after applying the filter

Figure 4: Example of key points found on image 5

3.2 Homography based on boxes dimensions

This technique is based on randomly extracting one key point and then finding other three key points around it, so to be able to find an homography with four key points.

The last three key points are to be found near the first one, where *near* is specified by the template's dimensions: each template is defined by a *width*, *height* and *depth*, expressed in centimeters (see below). The coordinates of the last three key points should be no further away from those of the first one than the dimensions found in the *dimensions.json* file, depending on the template.

Clearly, the approach has to be carried out after key points identification: a task that can be completed either with or without the use of Canny Filter Method. The use of the filter can influence execution time and consistency, as illustrated in Results.

The main flaw of this approach is its dependence on randomization, which

is key in improving performance, but it can also lead to output slightly different solutions each time the algorithm is ran.

```
{
 "barchette": {
  "width": 0.192,
  "height": 0.241,
  "depth": 0.061
 },
 "camomilla": {
  "width": 0.1524,
  "height": 0.0762,
  "depth": 0.0762
 },
 "frontline": {
  "width": 0.14,
  "height": 0.104,
  "depth": 0.012
 },
 "herolight": {
  "width": 0.145,
  "height": 0.142,
  "depth": 0.035
 },
 "infuso": {
  "width": 0.074,
  "height": 0.075,
  "depth": 0.066
 },
 "kellogs_classic": {
  "width": 0.194,
  "height": 0.305,
  "depth": 0.095
 },
 "krave": {
  "width": 0.192,
  "height": 0.241,
  "depth": 0.055
 }
}
```

Listing 3.1: JSON file containing boxes dimensions

3.3 Homography based on Rectangles

The proposed approach is based on the identification of Rectangles inside the scene and then uses them to select the regions of the image in which the algorithm can match the template. The use of rectangles to identify the area to analyze is due to the geometry of the boxes (template) to identify.

The identification of the areas in which the algorithm is more probable to match the pattern can be also seen as a filtering approach in which only the key points inside rectangles are used in the rest of the process. In the proposed implementation, the rectangles are used both to filter the key points and to fit the homography.

A noteworthy part of this approach is the identification process of rectangles, which is performed by executing the following operations:

- Identify lines (Canny + Hough) both from the scene and the Depth Map[1]
- Select the most promising lines
- Use pairs of almost perpendicular lines (angle of 70/90 degrees) to identify the rectangles

The bottleneck of the whole process is the identification of the rectangles and further improvement of this approach can focus on it.

(a) identified rectangles (b) final results

Figure 5: Example of rectangles identified on the image 5

4 Results

In general, the performance of this type of algorithm is measured in terms of speed; thus, the time of execution is taken as the main index of comparison between the base algorithm and those proposed.

[1]Denoising process of the image and of the Depth Map is performed in order to improve the identification of the lines

The following results show execution time based on the search of Kellog's Barchette and compare our solutions with those of customFindHomography3D-Plane, developed by Alessandro Polidori and Nicolò Stagnoli [3].

Time is expressed in seconds and has been approximated to the second decimal digit.

n° box	Dimension w/ filter	Dimension w/o filter	Rectangles	base
1	3.15	3.07	-	5.94
2	5.23	5.19	-	18.58
3	6.96	6.96	-	25.59

Figure 6: Image 1

n° box	Dimension w/ filter	Dimension w/o filter	Rectangles	base
1	2.70	2.63	0.28	5.94
2	4.79	4.83	0.47	10.72
3	6.36	6.70	-	14.56

Figure 7: Image 2

n° box	Dimension w/ filter	Dimension w/o filter	Rectangles	base
1	2.48	2.09	-	6.76
2	4.04	3.36	-	11.86

Figure 8: Image 3

n° box	Dimension w/ filter	Dimension w/o filter	Rectangles	base
1	2.97	2.68	0.47	6.73
2	4.48	4.15	-	12.02

Figure 9: Image 4

n° box	Dimension w/ filter	Dimension w/o filter	Rectangles	base
1	3.75	3.84	0.53	0.07
2	5.58	5.79	0.78	5.93
3	6.89	7.01	0.92	18.29

Figure 10: Image 5

n° box	Dimension w/ filter	Dimension w/o filter	Rectangles	base
1	3.72	3.66	0.52	9.61
2	5.83	5.84	0.88	14.26
3	-	-	1.06	18.06

Figure 11: Image 6

n° box	Dimension w/ filter	Dimension w/o filter	Rectangles	base
1	4.40	4.20	1.58	1.23
2	6.21	6.25	-	4.11

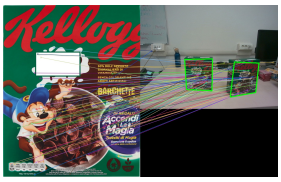
Figure 12: Image avanti_dietro

n° box	Dimension w/ filter	Dimension w/o filter	Rectangles	base
1	3.73	3.63	0.58	0.93
2	5.99	5.82	-	5.04
3	7.65	-	-	9.04

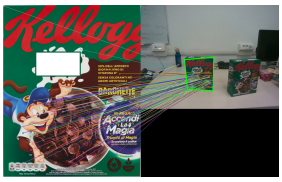
Figure 13: Image avanti_dietro2

As it can be noted from the tables, all three methods are often faster than customFindHomography3DPlane. In particular, if customFindHomographyRectangles finds rectangles in the image, it can outperform drastically the other methods while maintaining the same precision, as can be seen in the following screenshots. On the other hand, customFindHomographyDimension (with or without filter) is more consistent, meaning that it finds almost always all the boxes, but it is slower.

(a) customFindHomographyDimension with filter

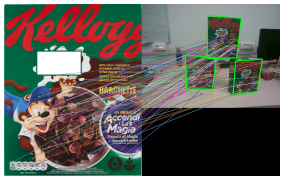


(b) customFindHomographyDimension w/o filter

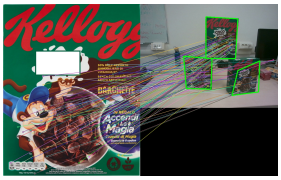


(c) customFindHomographyRectangles

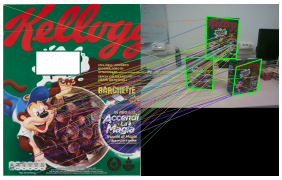
Figure 14: Image 4



(a) customFindHomographyDimension with filter



(b) customFindHomographyDimension w/o filter



(c) customFindHomographyRectangles

Figure 15: Image 5

5 Conclusion and open problems

The report illustrates different techniques to tackle the problem of template matching boxes. Each technique shows different strengths as seen in the above results. Overall there is an improvement in the matter of execution times with respect to the work done by Polidori and Stagnoli [3].

Several issues are left open, such as improving the effectiveness of the rectangles identification in customFindHomographyRectangles.

Another problem is that it may happen that any of the proposed algorithms finds degenerate bounding boxes, which are clearly not linked to any box and have to be removed.

At last, randomization poses a question: different executions of the program may lead to different solutions. While it is key to improving the performance of the proposed algorithms, its impact has to be constrained so not to damage the overall consistency of the program.

References

[1] Ebrahim Karami, Siva Prasad, and Mohamed Shehata. *Image matching using SIFT, SURF, BRIEF and ORB: Performance comparison for Distorted Images*. Oct. 2017. URL: https://arxiv.org/abs/1710.02726.

[2] Greta Corti, Giuseppe Chiari, and Hazem Shalby. *Github Repository*. Aug. 2022. URL: https://github.com/shalbyhazem99/IACV-project.

[3] Alessandro Polidori and Nicolò Stagnoli. *Github Repository*. Feb. 2022. URL: https://github.com/AlessandroPolidori/Template-matching-in-depth-images.

[4] Karol Matusiak, Piotr Skulimowski, and Pawel Strumillo. *Depth-based descriptor for matching keypoints in 3D scenes*. 2018. URL: https://www.journals.pan.pl/Content/107737/PDF/41_1451.pdf.