Product Recognition on Store Shelves

Adriano Cardace, Project of Computer Vision and Image Processing 2018/2019 *

*University of Bologna

Abstract

The goal of this project is to use computer vision techniques to perform object detection. In particular, we want a system able to recognize products on store shelves. The task is complex, due to the high number of different possible variation in which the same object can occur into the scene. The simple solution proposed here combines both state of the art detectors based on deep learning and traditional techniques for object detection.

Description of the task

The problem is structured into two main step: the first task is the simplest one and relies only on traditional techniques of computer vision to perform object detection. We are given a set of model images, one for each product that we want to recognize, referred here as **reference** images, and a set of scene images, called **query**. The goal is to recognize just one single instance of the references that occur in a query image, by marking with a bounding box all the recognized products. Figure 1 shows an example of a query image and the expected result.

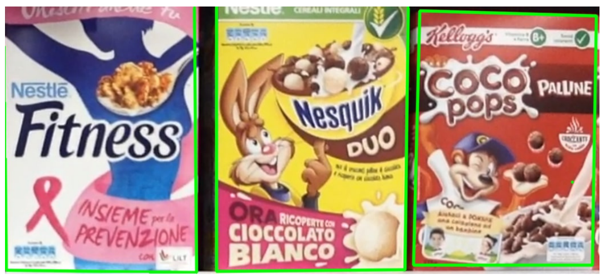
Nestlé
Fitness
INSIEME per la PREVENZIONE
Nesquik
DUO
ORA RICOPERTI CON CIOCCOLATO BIANCO
Kellogg's
COCO pops
PALLINE

Fig. 1: Output of Task A

```
Product 0 - 2 instance found:
    Instance 1 {position: (256,328), width: 57px, height: 80px}
    Instance 2 {position: (311,328), width: 57px, height: 80px}
```

Fig. 2: Output 1 of Task B

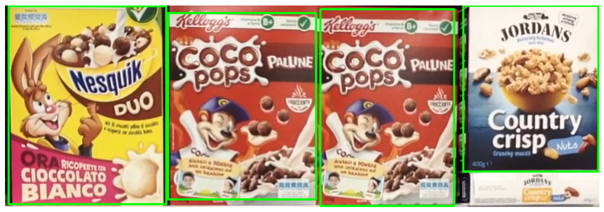

Fig. 3: Output 2 of Task B

The second task foresees that the system should be able to recognize all the objects present in the query images, thus also multiple instances of the same reference image. In this case not only the bounding boxes are required, but we want to give in output also the position of the product in the image reference system (center of the bounding box that encloses it in pixel) with the corresponding width and height. Figure 2 and 3 show examples of the expected results for this task.

Proposed Approach

Task 1

The strategy followed for solving the first task is the usual one deployed for object recognition, that is composed by three main steps:

- Detection of keypoints
- Computation of a keypoint's descriptor based on its neighborhood
- Matching descriptors among images

The first step is to find and store the keypoints for each reference. For each of them, a SIFT [1] descriptor is computed and stored, thus at the end of this process we have for each reference image its own set of descriptors. These two steps can be done very easily in Python using the OpenCV library: one just needs to get a SIFT object invoking the function *xfeatures2d.SIFT_create()* and then to call the *sift.detectAndCompute()* function, that returns both the keypoints and the corresponding descriptors. Since the task requires that we only need to localize just one instance for each reference (if present) we can simply scan the descriptors saved previously, and look if there is a strong evidence that one of the reference image is present into a query image. In order to accomplish this, keypoints and SIFT descriptors of each query are computed online and matched with the stored descriptors with a K-NN similarity search. In Python this boils down in getting a *BFMatcher* object and calling the method *knnMatch* with the two sets of descriptors. The two best matches are found to apply Lowe's ratio test [1] in order to reject as many as possible wrong matches and keep only the good ones. We conclude that a reference is occurring into the scene if there are at least 250 good matches detected. The 250 has been decided empirically and seems to work well in this scenario. When the number of good matches is above this threshold, the homography from the reference to the query is estimated through RANSAC. Finally, to get the bounding box of the detected object, the four corner of the reference are mapped using the homography to estimate the four corner of the object in the scene.

Task 2

The method used in the previous task does not work anymore in this scenario, because if there are multiple occurrences of the same object in the query image, the keypoints of the reference would match with points of different instances, ending up in a completely wrong homography. One possible approach to address the problem can be to deploy local invariant feature with GHT (Generalized Hough Transform), but the method presented here tries to do something completely different and can be thought as a simplification of the work done in [2].

The goal is achieved using the YOLOv3 [3] implementation publicly available at [4] to find the bounding box of all the objects present in the query. Finally each cropped instance is extracted from the original image to feed a classifier. Due to the fact that for each model we only have one reference image, the whole process has been split in two, since CNN-based object detectors, require a large training dataset to be able to predict both positions and labels of the objects. The network has been trained with the dataset provided by the authors of [2], that has 1247 images of store shelves, in which the annotations consist in coordinates for each object. It is important to highlight that the label is the same for all the objects, since in this step we only want to localize them. Before starting the training, is necessary to tune some parameters of the configuration file in order to say to the network that it has to recognize just one category. The network has been trained for only 15000 steps due to some hardware limitations, although this number is still sufficient for the small test done in this project, since eventually the system will be able to recognize and classify correctly all the instances in the proposed queries. Once the training has been completed, all the network's configuration parameters and weights can be saved into external files to run some inference later. In this project it has been used a YOLOv3 python wrapper called *Pydarkent*, that gives the possibility to run inference on YOLOv3 using the trained weights and the configuration file stored previously without actually installing the network (although it is still necessary for the training part). This make easier the problem of extracting all the coordinates of each product that occurs in the input query image. Another option is to directly modify the source code of the network, but less practical. Given the coordinates, all the detected objects are then cropped from the original image and compared with the references stored in the offline phase with the same technique used for task A, although now we use the Hellingher distance rather than the Euclidean distance in order to compare two descriptors. This can be done by L1 normalizing each descriptor vector, and then taking the square root element-wise. At this point applying the classical Euclidean distance to this normalized vectors is the same as computing the Hellingher distance. This modified version of SIFT is called RootSIFT [5] and as explained in the evaluation it allows us to increase the performances of our system. The code for the SIFT extension can be found in the file named *rootsift* under the *detector* folder. Other two customization deployed are to resize each reference and each cropped image to a fixed size and to set a more restrictive Lowe's ratio, 0.6 rather than 0.8 as in task A, since now we don't have to estimate the Homography and we are more concerned to get the right matches. The classifier in fact, chooses the reference with the maximum amount of good matches. In the project directory is possible to see the network's parameter under the YOLO directory, while the detector module contains the *Matcher* class, which is able to perform inference given a query image with the *predict()* method.

Qualitative and Quantitative results

Task A

Due to the simplicity of the first task a simple and classic approach has been used. Nevertheless the system is able to perform well although there are one false negative and one false positive in the five images provided for this task as a small test set (images in folder *taskA/scenes*). For example, in Figure 1, in addition to the three visible objects, also the product showed in Figure 4 is wrongly detected, probably because it is very similar to the middle cereal box. This problem will be solved in Task B, since for each box in the scene only one possible prediction is given (the best reference match).

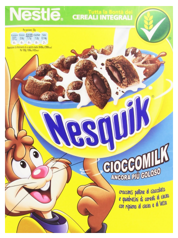

Fig. 4: False positive

Task B

The query images used for Task B may contain more instances of the same cereal box. In the restrict test cases provided in the problem set (only 5 queries) the system works perfectly and is able to recognize correctly all the instances with the corresponding labels. As highlighted previously, we don't have the problem in which one object can be associated with more references, since only the best match will be kept. The problem is not completely solved tough, because now the reliability of the system depends also on the output of the first step (localization). If in fact the neural network wrongly detects a non existing object, this will be associated anyway with one reference stored in the database that provides the best match, hence resulting in a false positive. This can be partially solved by maintaining only the detection in which the confidence of the network is higher than a certain threshold t. To get a better sense of the quality of the system a more rigorous test has been performed. In particular, it has been computed the mean average precision-mAP on the test set provided in [5] that consists in 70 labeled images. The best result is achieved using RootSIFT and a more strict Lowe's ratio (0.6). In table 1 all the results with different combinations of hyper-parameters are summarized. It is interesting to notice how RootSIFT always outperform SIFT. Reaching a peak of 65% is a noticeable result considering the simple approach used here, but at the same time is still far from the best result achieved in [2], and we still have to keep in consideration the small size of the test set. Other strategies have been tested, in particular for the classification task of the cropped images, such as computing HOG features to feed a SVM classifier or to perform a KNN search, but all this methods seem to perform worse than the object recognition implemented with RootSIFT.

Table 1: Comparison between SIFT and RootSIFT

Ratio Threshold	SIFT mAP	RootSIFT mAP
0.8	52%	61%
0.7	58%	64%
0.6	59%	**65%**

1. D. Lowe: Distinctive image features from scale-invariant keypoints. IJCV 60, 91110 (2004)
2. A. Tonioni, E. Serro, L. Di Stefano: A deep learning pipeline for product recognition on store shelves. arXiv:1810.01733, 2018
3. J. Redmon and A. Farhadi: YOLOv3: An Incremental Improvement. arXiv:1804.02767, 2018
4. https://github.com/AlexeyAB/darknet
5. A. Tonioni and L. Di Stefano. Product recognition in store shelves as a sub-graph isomorphism problem. In International Conference on Image Analysis and Processing, pages 682693. Springer, 2017.
6. R. Arandjelovic and A. Zisserman. Three things everyone should know to improve object retrieval. In CVPR. 29112918, 2012