Letizia Mancini - letizia.mancini3@studio.unibo.it

Product Recognition on Store Shelves

Introduction:

Object detection techniques based on computer vision can be deployed in super market scenarios for the creation of a system capable of recognizing products on store shelves.

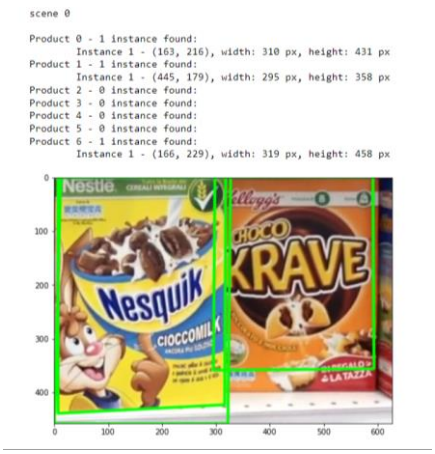
Given the image of a store shelf, such a system should be able identify the different products present therein and may be deployed, e.g. to help visually impaired costumers or to automate some common store management tasks (e.g. detect low in stock or misplaced products).

Step A - Multiple Product Detection:

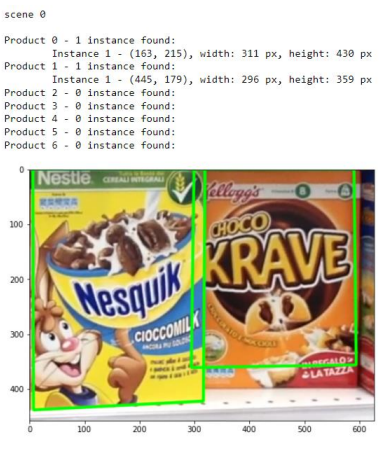
The local invariant features paradigm allows to identify single instance of products given one reference image for each item and a scene image. The first step is loading the images and detecting the keypoints. This can be done using the **Scale-Invariant Feature Transform (SIFT)** algorithm.

We initialize a SIFT detector object to *detect* the keypoints and *compute* for each one a unique description, based on the local neighbourhood. We can determine the best parameters experimentally. Considering the other parameters fixed, we obtain a better result incrementing the number of octave layers and the contrast threshold. The parameters used in this case are:

- **Number of octave layers** = 5
- **Contrast threshold** = 0.1
- **Sigma** = 1.4



1. Using the standard values



2. Changing the parameters

In the feature matching phase, descriptors extracted from the scene are compared with those extracted from the models to find couples of similar ones. To search more efficiently, we will use an approximate indexing algorithm from the Fast Library for Approximate Nearest Neighbors (FLANN). The search space consists of 5 trees recursively traversed *checks* times: using higher value takes more time but gives better accuracy.

A good trade-off between time and precision is the value 100. The matches found were then filtered out with Lowe's ratio test so as to keep only the good ones.

Finally, we can use the Random Sample Consensus (RANSAC) algorithm to compute an homography from the good matches, while identifying and discarding the lists of matches containing less than the minimum MIN_MATCH_COUNT.

We extract the location of the matched keypoints in both the images building correspondences arrays and using them to estimate the homography with the function *cv2.findHomography*. We define the **BoundingBox** class to store the output information and correct the corners of the bounding boxes obtained with the *cv2.perspectiveTransform* function.

The **check_overlapping** function checks if there are any overlapped boxes by confronting their center and adds to the **dst** matrix the one with more good points.

Parameters used:

- **MIN_MATCH_COUNT** = 80
- **Overlap Threshold TH** = 10.0

Step B - Multiple Instance Detection:

To detect multiple instance of the same product, we can use the Generalized Hough Transform (GHT) with local invariant features.

In the offline phase, we detect the keypoints and compute the descriptors as before. The new class **ExtendedKeyPoint** is used to add the **Vi** vector to the detected keypoint.

In the online phase, we start by computing keypoints and their respective descriptors and then we match descriptors between target and model features, as done in Step A. After filtering the false matches, the barycenter position's votes are cast.

The class **HoughSpacePoint** represents a point in the Hough Space, defined by:

- **src_kp** = the model keypoint
- **dest_kp** = the matched keypoint
- **s** = the scale factor
- **phi** = the rotation angle
- **pt** = the barycenter

For each good match, we create a new Hough Space point. The barycenters are clusterized by dividing the scene images for a QUANTIZATION parameter. The coordinates of the point are then used as dictionary keys for the barycenters hypothesis area.

After casting the votes, the hypothesis area with less than TH votes are removed and we use the RANSAC algorithm to compute the homography and estimate the position.

As in the previous step, we check for overlapping bounding boxes so to keep only the one generated by the greater set of barycenters.