Product Recognition on Store Shelves

Nicholas Antonio Carroll, Laura Mazzuca

Computer Engineering,
Computer Vision and Image Processing
Exam Project
`https://github.com/lauramazzuca21/shelf-product-recognition`

Abstract

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).

1 Step A: Single Instance Detection

To detect single instances of products on shelves the local invariant features paradigm is used starting from a single model image per object.

1.1 Offline phase

Once the model and scene images have been loaded and transformed into greyscale the first step is identifying keypoints in all our images. This is done by using the Scale-invariant feature transform (SIFT) algorithm from the OpenCV library.

With respect to the parameters suggested in the original paper, the values did not give great results and, through trial and error, these parameters were found to be the best ones in this case:

- **Number of octave layers** = 5
- **Contrast threshold** = 0.11
- **Sigma**: 1.4

For each keypoint is then computed a unique SIFT descriptor.

1.2 Online phase

1.2.1 Feature Matching

For the feature matching phase, the Fast Library for Approximate Nearest Neighbors (FLANN) is used, with a search space of 5 trees traversed CHECKS times. When using higher check values, higher accuracy can be obtained, but it also takes more time. During our tests we have found that a value of 500 gives us optimal results. The matches found were then cropped as per Lowe's ratio test so as to keep only the good ones.

1.2.2 Position estimation

Lastly the Random Sample Consensus (RANSAC) algorithm is used to estimate an homography from good matches. In this phase, the lists of matches containing less than the minimum match count threshold are ignored to exclude the less promising models for each scene Then the OpenCV function findHomography is used and it returns the homography and the mask to exclude the discarded points. Finally the bounding boxes are obtained with the perspectiveTransform OpenCV function and added to the dst matrix. To store all the information needed for the output and to correct overflowing corners, the class **Box** is defined. The **checkOverlap** function is used to make sure there are no bounding box overlapping by confronting their centers. If they appear to be overlapping, the one with the most good points gets added to the **dst** matrix, while the other gets removed from it. Then, when a whole scene is scanned, the bounding boxes are finally drawn. The parameters we used are:

- **minimum match count threshold** = 80
- **overlap threshold** = 15

2 Multiple Instance Detection

The process done for a single instance can't be directly applied to the multiple instance problem, so the strategy used for this step is the Generalized Hough Transform with Star Model geometric validation.

2.1 Offline phase

In the training phase, the most important difference is the introduction of a class called **ExtendedKeyPoint** which stores the detected keypoint, its descriptor and the computed **Vi** vector.

2.2 Online phase

The online phase is the same as 1 up until the the good matches cropping by Lowe's ratio. Then, the barycentre hypothesis' votes are cast.

The class **HoughSpacePoint** represents a point in the hough space and it is composed of:

- **srckp**: the model keypoint
- **destkp**: the matched keypoint
- **s**: the scale factor
- **phi**: the rotation angle
- **pt**: the computed barycentre point

For each couple scene-image, we generate the **HoughSpacePoints**, for which the pt value does not take into consideration the rotation matrix because the result was somewhat worse than the one without it. This is most likely given by the fact that the rotation is trivial. To make clusterization easier and faster, each scene image's hough space is generated by dividing the image into squares 100px*100px, represented by couples (x, y). Then, these couples are used as dictionary keys for the barycentre hypothesis areas in which the barycentres can fall into.

Once the votes are cast, the barycentre hypothesis areas with less than **THRESHOLD** votes are cut and, finally, the actual position estimation is made using the RANSAC algorithm. To check for overlapping bounding boxes, an evolution of the algorithm used in 1 is used. This algorithm extends the search space for overlaps to instances of the current image being analyzed, so that if there are multiple bounding boxes generated by different sets of barycentres, the most numerous set is kept.