Indian Statistical Institute

M. Tech. (Computer Science) Dissertation

Solving Object Detection and Localization in an Image

A dissertation submitted in partial fulfillment of the requirements for the award of Master of Technology in Computer Science

Author:
Rajeev Baditha
Roll No: CS-1523

Supervisor:
Prof. Dipti Prasad Mukherjee
Electronics and Communication Sciences Unit

M.Tech(CS) DISSERTATION THESIS COMPLETION CERTIFICATE

Student: Rajeev Baditha (CS1523)

Topic: Solving Object Detection and Localization in an Image

Supervisor: Prof. Dipti Prasad Mukherjee

This is to certify that the thesis titled "***Solving Object Detection and Localization in an Image***" submitted by **Rajeev Baditha** in partial fulfillment for the award of the degree of Master of Technology in **Computer Science** is a bonafide record of work carried out by him under my supervision. The thesis has fulfilled all the requirements as per the regulations of this Institute and, in my opinion, has reached the standard needed for submission. The results contained in this thesis have not been submitted to any other university for the award of any degree or diploma.

Date: Prof. Dipti Prasad Mukherjee

Dedication

To my parents and my well wishers, without your help and encouragement it would not have been possible.

Acknowledgements

I would like to thank my dissertation supervisor Prof. Dipti Prasad Mukherjee for agreeing to guide me and for helping me to undertake work in the topic.

Abstract

Object recognition in natural images using the database images taken under ideal lighting conditions has been a challenging problem in the field of computer vision. In this report, we solve the problem of identifying the products in the rack image of a grocery store. In the past few years this has been the interest of many computer vision researchers. We have the database images of different products available in a grocery store using which the products in a rack image need to be found. The problem can be divided into two major sub problems of matching and localization. We first use the matching from SIFT and try to improve the matching using a patch based matching algorithm. After matching, we are left with localization part and a product could be present at more than one location in the rack. We locate multiple instances of a product by clustering matched points using density based clustering methods. Although this method is vulnerable to outliers, we try to reject few of them by ignoring the less dense clusters. Finally we run this algorithm on our data set. We report good accuracy on our data set.

Contents

List of Figures

Chapter 1

Introduction

Object detection and localization play an important role in many computer vision applications. By object detection and localization we desire to find the location of a template image in the given input image. This problem gets harder when the template image is taken under ideal lighting conditions and the input image is a natural image. In this report we try to solve the problem of identifying the product images on a grocery shelf image. Recently this problem has been of great interest of many computer vision researches.

Major retail businesses are interested in applications that will help them enhance the shopping experience, more easily perform stock inventory, and monitor shelf status more frequently. Other applications of this include assisting the visually impaired through grocery stores.

Section 1.1 has a detailed explanation of the problem statement. Section 1.2 presents the challenges of solving the problem. The work happened in this area is presented in Section 1.3. Overview of the methodology is given in the Section 1.4

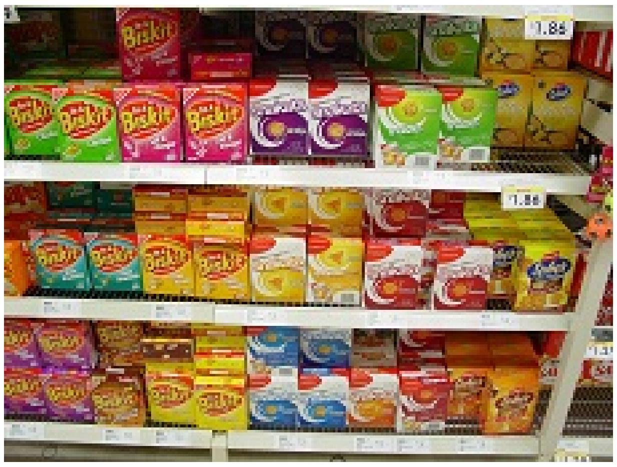


Figure 1.1: Typical rack image

1.1 Problem Statement

In this report we are trying to solve the problem of locating the products in the rack image of a typical grocery store. We have the database of all the product images available in the store. Using this database we need to recognize the products in the given rack image. A single product could be present at more than one location. We have one instance of every product available in the grocery store. The products in the rack image will be at different scales, lighting conditions compared to the database product images.

We don't assume any information about the rack image meaning that any product could be present in the given rack image. Details like number of shelves in the given rack, type of products that could be present in the rack, size of the rack, number of instances of product in the rack are all unknown. Although having such information makes the problem simpler we try to solve the problem without that information. A typical rack image can be seen in the Figure 1.1 from web market database [25].

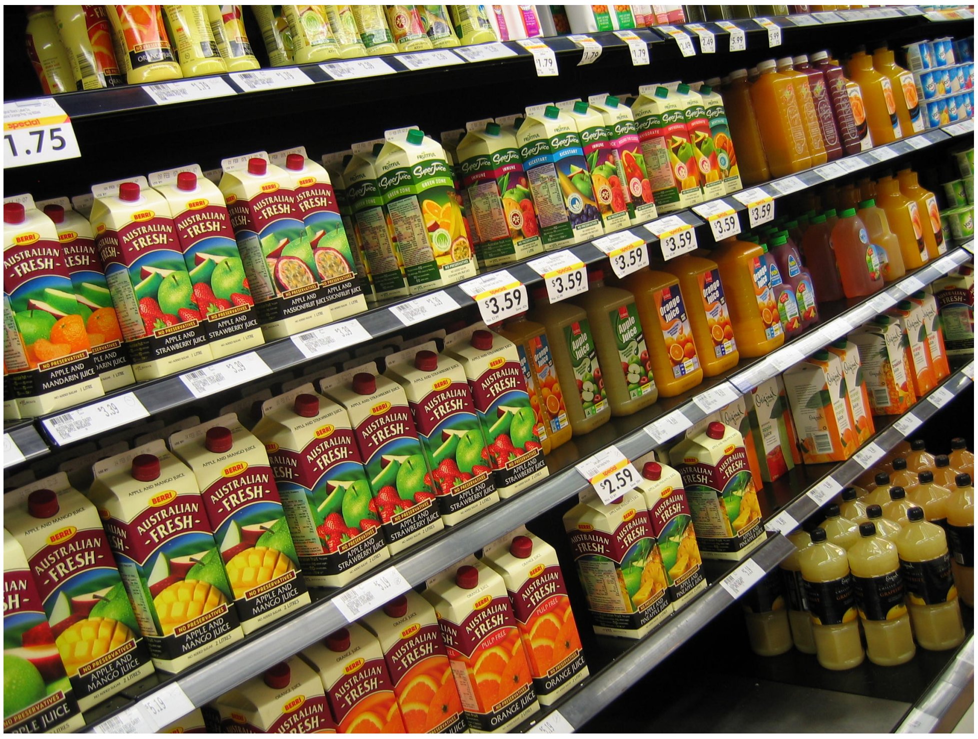


Figure 1.2: Skewed rack image

1.2 Challenges

The challenges in solving this problem can be divided into several headings, as discussed below. In this section, the challenges are highlighted.

1.2.1 Skewed Images

Owing to the fact that the entire image of the shelf is captured in a single shot, it is not possible to have the cameraman(or the robot, as the case may be), to stand at sufficient distance from the shelf. Hence the distance of each point on the shelf varies significantly, resulting in a skewed image of the shelf. Figure 1.2 shows a skewed rack image.

1.2.2 Extreme Variation in Illumination

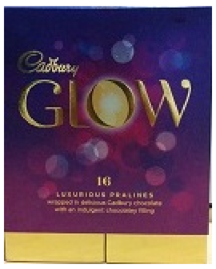
Due to the lack of source of any natural light, shopping mall is an area which is invariably lit artificially. The color of the incident light varies over a wide range between different stores and also between different locations in the same store. This difference in color results in significant changes in the appearance of the images, even at a perceptual level. This brings up a huge challenge in the identification of the algorithms because the information present in the *color* becomes fairly difficult to extract.

1.2.3 Differences in Quality of Target and Query Images

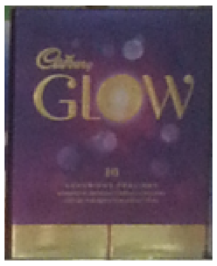
The images provided in the Product database are of high resolution, while the images cropped from the shelves are of lower resolution. Obtaining similar quality images of both the products and the rack images are not feasible because while capturing the store image, the camera covers the entire rack in a single shot, with the dimensions of the image being around 2000X1000, while the same camera can be used to capture high resolution images of the products, when photographing them one at a time. Figure 1.3 shows product image and a cropped rack image illustrating this.

1.2.4 Changes in the Target Images

Once a new product is launched, as the case is during a promotional offer, the appearance of the product image changes. This brings about new challenges in the matching of the images. The color might have been changed a bit, while the brand logo might have been shifted from the top of the product to the bottom. These changes are illustrated in Figure 1.4.

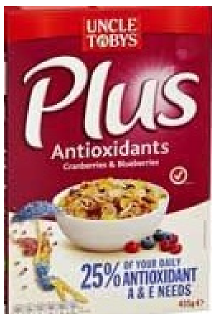


(a) Product image

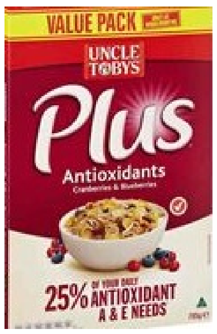


(b) Product cropped from rack image

Figure 1.3: Illustration of the quality of rack and target images



(a) Product image without offer



(b) Product image with offer

Figure 1.4: Illustration of change in the target images

1.3 Related Work

Objection detection and localization are building blocks of many computer vision applications. Naturally lot of work has been done in solving both the problems. By object detection and localization our target is to find the location of the given template in an input image. The template and the input images generally differ in scales. This can be naively solved by calculating normalized cross correlation of template and input image. Further the template image is localized at the peaks. Because the scales are different we can extend this to a multi-scale cross-correlation. This method works in very constrained environments and is not applicable in our case.

The naive method can be improved by comparing some features associated with pixels rather than comparing just raw pixel values. The work of Hae Jong Seo et al. [21] does exactly that. Rather than considering just the raw pixels values, the authors here compute locally adaptive regression kernels by considering a neighbourhood around a pixel. Matrix cosine similarity is used to compare two patches here. This method is claimed to be invariant to noise, rotation. Although this method identifies the products better, it fails when lighting is different in both the images.

A robust method to solve this problem is by using SIFT [16]. David Lowe's SIFT is undoubtedly the most used algorithm in computer vision. David Lowe in his paper explains how SIFT can be used for object detection. The fact that SIFT is invariant to scale, rotation and partially invariant to lighting, 3D viewpoint makes it most suitable for our application. But the product identification fails considerably when the target image is present at more than one location. This can solved in a couple of ways. We will mention those later in the report. Matching results using SIFT are appreciable which is why we have used it in our algorithm.

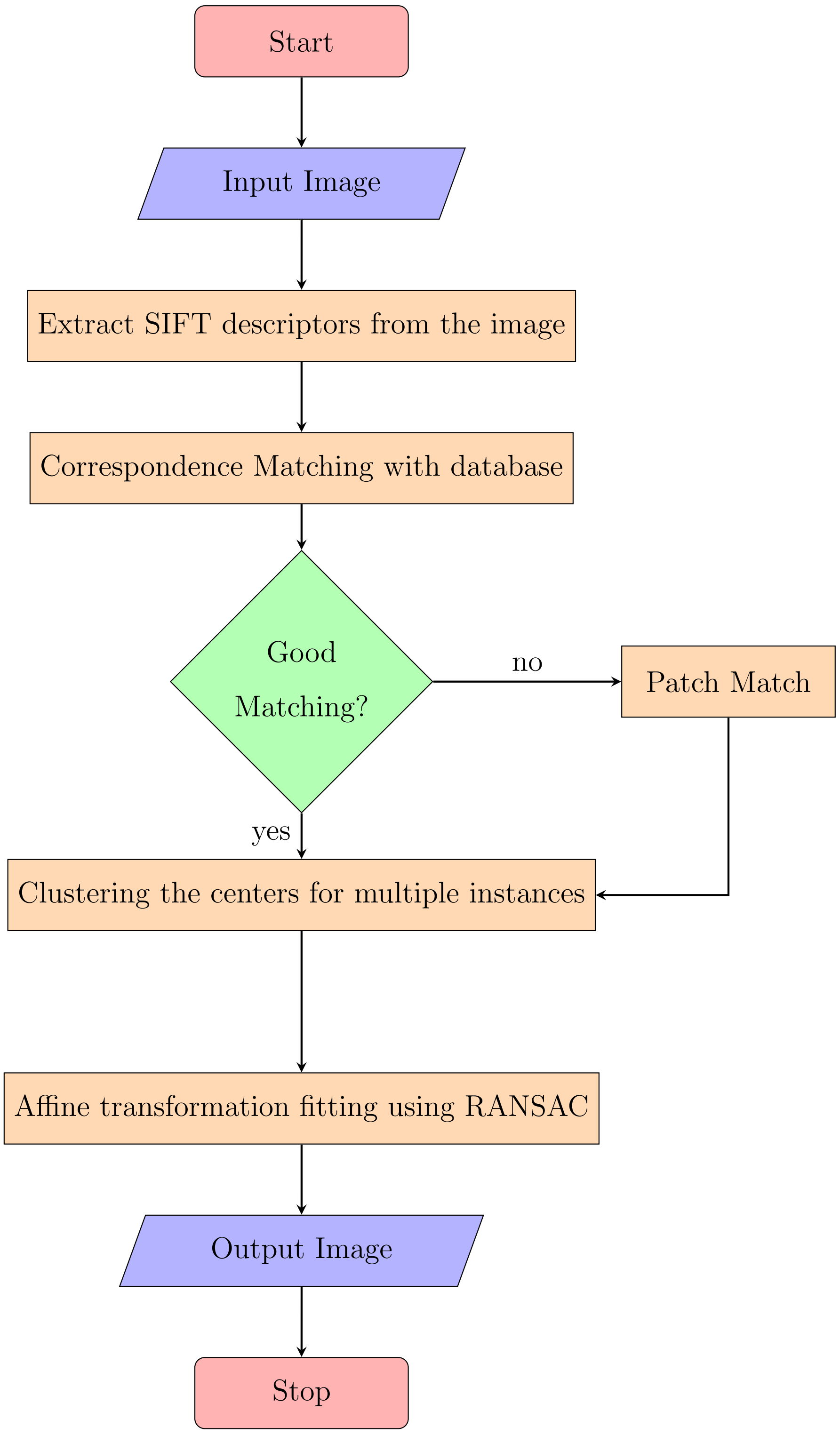
The problem of recognizing the products on the rack image of a grocery store attained a lot of attention in the last decade. As mentioned in the introduction solving this problem has two applications. One is planogram compliance and the other is assisting the visually challenged while shopping. Few researchers concentrated on building a mobile application for assisting the blind whereas others concentrated on planogram compliance. The work of [5], [12], [23] focuses on assisting the visually impaired in . Rabia Jafri et al. in [13] provides a survey on different methods focusing the same problem. Making a mobile application is the target of most of the works so it is necessary that the algorithm runs fast to work real time. Planogram compliance is another important problem which saves a lot time and money for the store owners. The major work done in this area is by [15]. In these works authors assume that planogram is already given.

Earliest work on this problem dates back to the work by Merler et al. [18] where they have introduced a data set by name GroZi-120. In this paper they propose 3 methods using colour histogram, SIFT, Boosted Harr like features and solve the problem. The main contribution of this paper is the GroZi-120 data set as it is the first publicly available data set available for other researchers to work with.

Other work on this problem is by Aucliar et al. [1] where they extended SIFT for multi instance object detection. The concentrated mainly on making the detection faster so that it can can be used in real time.

Few other works can be found be found at [10] and [17]. George et al. in [11] introduces a new data set called Grocery Products data set. Varol et al. in [22] proposes a machine learning approach to solve this problem. Multi instance objection approaches can be found in [2] and [24]. Chong et al. in [6] uses deep learning to solve the problem.

1.4 Overview of the Algorithm


Start
Input Image
Extract SIFT descriptors from the image
Correspondence Matching with database
Good Matching?
no
Patch Match
yes
Clustering the centers for multiple instances
Affine transformation fitting using RANSAC
Output Image
Stop

In this section we present the overview of the proposed methodology. Flow chart summarizing our algorithm is given in Figure 1.4. Our algorithm has two main parts one is matching and the other localization. Brief explanation about these two is given below.

In matching step we intend to find the correspondence between product image and the input rack image. We used SIFT to match the key point descriptors between product images and rack image. In Section 1.2 we have given the challenges for solving the problem. Most of them can be solved using SIFT. SIFT is invariant to scale, rotation and partially invariant to 3D viewpoint and lighting. These features of SIFT answers the challenges posed by skewness and varied lighting conditions.

Even though SIFT is robust it doesn't necessarily produce good matches every time. Therefore the correspondence matching needs to be improved in case of mismatches. We improve it using a patch based matching algorithm. The main idea of this algorithm is to use the found good match and propagate it to its neighbours. This ensures spatial consistency. Another major step in this algorithm is to randomly check for good match in the neighbourhood of existing match. To check if two patches match or not, their HOG descriptors [7] are compared. Lesser the euclidean distance between the descriptors better the match. Overall we observed improved matching with this approach. The detailed description of this algorithm is given in Chapter 3.

In the localization part we first interpolated the product centre using every correspondence. The product centers are sent to a clustering algorithm to detect the multiple instances of a product. Finally RANSAC [9] is used to solve for the affine transformation matrix between product and rack images. We generate a reconstructed rack of the input by transforming the product images using the found affine transformation.

Chapter 2, 3 combined has the methodology of the approach. Chapter 2 has the baseline approach whereas chapter 3 has the patch based matching algorithm. Results on our data set are presented in the Chapter 4. Conclusion and Future work is in Chapter 5.

Chapter 2

Methodology

In the introduction we have given the flow chart of the algorithm. In this chapter we present the detailed methodology of our baseline algorithm. The baseline approach is heavily dependant on SIFT and is only as good as SIFT. We tackle its limitations using a patch based matching algorithm. Increased detection rates are achieved using the patch based matching algorithm. The details of it can be found in the next chapter.

As mentioned before the algorithm can be divided into two main parts matching, localization. Section 2.1 has a brief overview about SIFT. Section 2.2 describes the use of SIFT features to perform affine matching. It is followed by a robust method for multi instance object localization. Section 2.3 has a detailed explanation of clustering the multiple instances. Section 2.4 describes about the final affine fitting algorithm. The final filtering step is detailed in Section 2.5. This chapter is concluded by listing the pros and cons of the approach followed by the necessity to go for a patch based matching algorithm

We reiterate the problem statement and continue with the methodology of the proposed approach. Our target is to identify the products present in the rack image of a grocery shelf. Number of shelves in the rack image could vary from one to five. We have a database of all the product images taken under ideal lighting conditions. Our input is a rack image and our job is to find what products constitute the rack.

2.1 Affine Descriptor

In this section we mention about the main building block of our algorithm, SIFT[16]. The robustness of SIFT makes it an ideal tool for solving our problem. We used SIFT for matching key points between product images and rack image. This section introduces the SIFT algorithm and Section 3.2 explains how we can use SIFT in our algorithm. This is just a brief overview of this descriptor and the complete details can be found in [16].

2.1.1 SIFT

Scale Invariant Feature Transform by David Lowe is the most popular image descriptor in the field of Computer Vision. It has been shown in a survey [19] that SIFT is the best available descriptor compared to other local descriptors. In the challenges section we have mentioned that the products in the rack image are at a different scale, rotation, and lighting compared to the product images. SIFT as the name suggests is invariant to scale and rotation. It is partially invariant to 3D viewpoint and lighting. This makes SIFT ideal descriptor for our application. The next paragraph has a brief overview of SIFT.

SIFT works by first calculating interest points in an image and it calculates a descriptor for every point. The interest points are called key points. These are the points which can be found repeatedly in different scales of the image. Key points are calculated by finding the extrema in the scale space. The point location and the scale at which it is found are noted down. Around every key point, an orientation is calculated which is the largest gradient orientation in the key point's neighborhood. For every key point a 128 length vector is calculated by considering a patch around it. Histogram of gradients in that patch is calculates and it forms the 128 length vector.

Therefore after SIFT; every key point detected is associated with a scale, orientation, and descriptor. The figure below has an example of SIFT output.

2.2 Matching Using SIFT

First step in our algorithm is to precompute the SIFT descriptors for all the product images. Number of products images could be high so precomputing the SIFT descriptors for all of them would speed up the algorithm. We calculate the SIFT descriptors for every product image and store it in a database. Although it increases the memory overhead, it is necessary to make the algorithm run quickly.

2.2.1 Precompuation

Let there be a total of N product images. From the product images every product is selected at once and sent to the SIFT algorithm. We create a database where $\mathrm{db_i}$ represents the database of SIFT descriptors for $\mathrm{i^{th}}$ image. For every image we get key points in the range of 200-500. For every key point scale, rotation and a descriptor is calculated. Therefore all the descriptors are of the form $\langle$ location, scale, rotation, 128-length vector $\rangle$.

2.2.2 Finding the Correspondences

After the pre computation step, the SIFT algorithm is run on input rack image. It generates the descriptors in the same way it has generated for product images. We store the SIFT output for rack image in IN. The number of key points in rack image can be as high as few thousands.

We intend to find the correspondence between rack image and product images. As we don't have any prior information about the type of products present in the input rack image we need to search for all the product images. So we assume that all the products have potential matches in the rack image. Corresponding matching is implemented in a linear fashion as follows.

Every feature in IN is compared to all the features in DB. We consider only the ε-neighbourhood of every feature. We do the same thing with all the features in the rack image. Here the correspondences could be from any of the product image. So we take the matches one by one. First we consider the matching with first product image, then with the second and so on till the last product. For a single product, this is same as calculating the pairwise distances between the rack image descriptors and product image descriptors. If the number of key points in the product image is m and the number of key points in the rack image is n, the pairwise distance matrix will be of size m*n. We take the Euclidean distance between the two descriptors while filling the matrix. This can be optimized by considering the square of Euclidean distance to reduce the computation.

The each value of matrix lies between 0 and 2. The value of ε has to be chosen to consider only the good matches. A threshold of 0.3 has given us good results for most of the products. Any distance less than 0.3 is considered a match otherwise a mismatch. Although few mismatches were found we resolve it in the further steps. Because a product could be present at more than one location this type of threshold is implemented. Other way of putting a threshold is by considering k-nearest neighbors for a fixed k. As we don't have information about the number of instances of product present in the rack this method is not feasible. Although we can put a cap on the maximum number of instances present, we will end up with lot of mismatches when the actual number of instances present is much smaller than the chosen k value.

The number of product images in the database could be high and the calculation of euclidean distance between two features is also computationally expensive. This makes calculating the ε-neighbourhood an expensive operation. A simple way of speeding up the process is to apply Principal Component Analysis(PCA) [14] on the descriptor vectors and reduce its dimension. This method though offers only a insignificant improvement over the previous one. Another way is the usage of k-d trees. Approximate nearest neighbour algorithm can be used with k-d trees for computing the ε-neighbourhood of a feature. A similar performance boost can be achieved by using hashing. Even though they are fast, it effects the matching because these algorithms only output the approximate neighbours.

2.3 Clustering Multiple Instances

The matching results show good matches with few outliers. . We need to come up with a solution to reject the mismatches. Multi instance object recognition is a challenging task in computer vision. Generalized Hough Transform is used to cluster the number of instances in [20]. This method gives good results when matches are correct. But the main problem with this approach is choosing the grid size in Hough Space. If the grid size is too small a single cluster might be detected as two clusters and when the grid size is large two or more clusters might be merged and found as one. Also high dimensionally of Hough Space makes it harder to implement.

Above mentioned reason necessitated the need for a different way to tackle the problem. First idea is to cluster the matched 2D points on rack image using a density based clustering algorithm. Because of the insignificant differences between the inter class and intra class distances this method failed to recognize the clusters. Another idea is to cluster the 4D vector containing the (x, y, scale ratio, theta) obtained from SIFT matches. Here scale is the ratio of scales between two matched features and rotation is the difference in orientation between the features. (x, y) is the location of key point on the rack that got matched. This almost has the same output as the previous approach.

To overcome the above mentioned challenges we makes small changes to the above clustering algorithm. As observed before we need to make the intra cluster distance small and inter cluster distance large. To achieve this we estimate the center of the product for every matched point in the rack image. This will make clustering easy. How we estimate the centers is as follows. [26] tackles the problem in the same way.

As mentioned before all the products images are cropped to exact boundaries. Therefore the center location of the product image is known to us. Using this information we try to interpolate the center of the product in the rack image. Let's say a point A from product image matches with a point B on rack image. Let C be the center of product image. For both the points we know the scale and rotation from SIFT.

CA is the vector joining center of product and point A. To interpolate the center on the rack image we need to change the scale and rotate this vector using the SIFT outputs.

After the estimation of the centers the resulting 2D points are clustered using DBSCAN algorithm. Here we could have used kmeans algorithm for clustering if the information about the number of clusters is known. As we have assumed no-prior information it takes out kmeans from our options. DBSCAN [8] requires two input parameters namely epsilon and min points for the radius of neighborhood and minimum number of points in the neighborhood. 3 points are required to calculate the affine transformation from product image to rack image so epsilon value is 3. But because of noise we might wrongly predict clusters as epsilon is low. Therefore epsilon = 10 is chosen for better results. A minimum radius of 5 is chosen as it is reasonable to assume that center will be estimated with the error of at most 10 pixels in both height and width.

2.4 Affine Fitting

Finally we are left with estimating the transformation matrix between product image and rack image. If we assume that most of the points of the product come from a planar surface we can fit an affine transformation between product image and rack image. Because the affine transformation has 6 degrees of freedom we need 3 matches to solve the affine transformation matrix. Each cluster from the clustering output represents an instance of the product and is associated with a transformation matrix. Every cluster may have more than 3 points in which few may be mismatches. Therefore to find the correct transformation we need a robust method. Thankfully RANSAC does it.

Random sample consensus (RANSAC) is an iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers.

RANSAC works by selecting 3 matches at random and solves the system of linear equations to find the affine transformation matrix. It then calculates the number of inliers and outliers with the found affine matrix. It repeats the process iteratively and returns the affine matrix with highest number of inliers. Here inliers refers to the matches that agree the affine matrix and outliers the opposite.

RANSAC fails when the number of inliers are less than 50 percent. Because RANSAC is randomized algorithm it may different outputs in every iteration. To overcome this problem we run RANSAC thrice and choose the best one every time.

2.5 Filtering

For every potential occurrence we get a triplet ⟨id, T, n⟩ where id is the product number, T is the found transformation matrix, n is the number of points in the cluster. We sort the triplet based on the number of points in descending order. Each occurrence is taken in the descending order and is pasted on an image with the same dimensions as the input rack image. As only one product can be present at a location we reject any occurrence if it is intersecting with an existing one.

Here we also reject all the transformed images which lie outside the output image. If the image is just few pixels outside the output image it is still accepted. The threshold used here is 10 pixels. This occurs mostly at the image boundaries because the calculated affine transformation may not be exactly correct.

Chapter 3

Patch Match

As mentioned before we try to improve the baseline approach using a patch based matching algorithm. We make use of Patch Match algorithm described in [3]. It is a randomized algorithm for finding the matches between two images. This paper aims at finding the approximate nearest neighbours for every patch. This is fast algorithm and can match for different scales and rotations. In this chapter we give a brief idea about the algorithm. Readers can go through [4] for the full details about the algorithm.

The main idea of this algorithm is to use the found good match and propagate it to its neighbours. This is called propagation. Random Search is another major step in this algorithm. In this step we randomly check for good match in the neighbourhood of existing match. To check if two patches match or not, their HOG descriptors [7] are compared. Lesser the euclidean distance between the descriptors better the match.

Section 3.1 has the brief description about HOG. Section 3.2 explains the main in the patch based matching algorithm. Section 3.3 describes its usage in our algorithm.

3.1 Histogram of Oriented Gradients

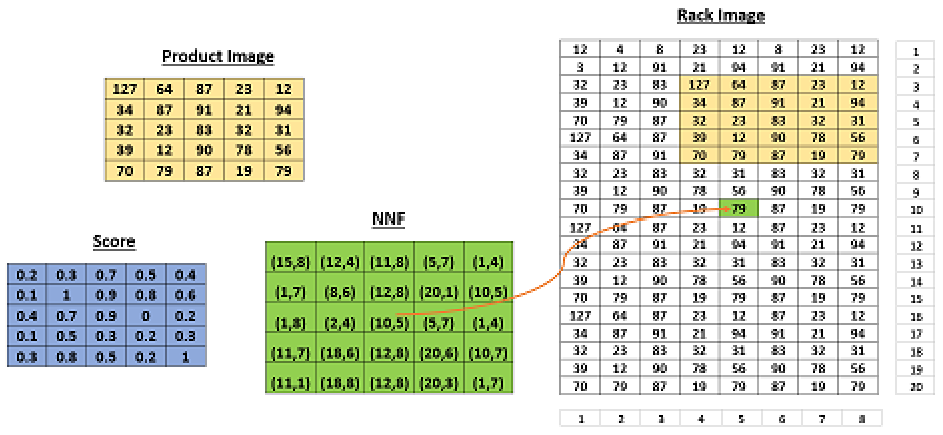
HOG is one of the popular image descriptors in computer vision. The original work of HOG can be found at [7]. Although HOG is a image descriptor here we use it as a point descriptor. For every pixel, a 17*17 neighborhood patch is taken with the current pixel as the center ans a 36 length HOG descriptor is calculated for every pixel. When comparing two patches we just compare their HOG descriptors.

3.2 Patch Match

In this section we explain a simple version of patch match algorithm. In this algorithm we try to find the approximate neighbours of the product image patches in the given rack image. For a product image, patch match algorithm returns the Nearest Neighbor Field (NNF) which has the same size as the product image where each location has the co-ordinates of the patch it has matched to in the rack image. Patch match algorithm has 3 main steps: Initialization, Propagation and Random Search.

3.2.1 Initialization

NNF matrix is initialized randomly. Let r, c be the number of rows and columns in the rack image. For every pixel, we randomly choose two integers x, y such that $x \leq$ r and $y \leq$c and make NNF (current pixel) = (x, y). We calculate the respective matching score and store the same in the score matrix as depicted in the figure 3.1.



Product Image

127	64	87	23	12
34	87	91	21	94
32	23	83	32	31
39	12	90	78	56
70	79	87	19	79

Score

0.2	0.3	0.7	0.5	0.4
0.1	1	0.9	0.8	0.6
0.4	0.7	0.9	0	0.2
0.1	0.5	0.3	0.2	0.3
0.3	0.8	0.5	0.2	1

NNF

(15,8)	(12,4)	(11,8)	(5,7)	(1,4)
(1,7)	(8,6)	(12,8)	(20,1)	(10,5)
(1,8)	(2,4)	(10,5)	(5,7)	(1,4)
(11,7)	(18,6)	(12,8)	(20,6)	(10,7)
(11,1)	(18,8)	(12,8)	(20,3)	(1,7)

Rack Image

12	4	8	23	12	8	23	12	1
3	12	91	21	94	91	21	94	2
32	23	83	127	64	87	23	12	3
39	12	90	34	87	91	21	94	4
70	79	87	32	23	83	32	31	5
127	64	87	39	12	90	78	56	6
34	87	91	70	79	87	19	79	7
32	23	83	32	31	83	32	31	8
39	12	90	78	56	90	78	56	9
70	79	87	19	79	87	19	79	10
127	64	87	23	12	87	23	12	11
34	87	91	21	94	91	21	94	12
32	23	83	32	31	83	32	31	13
39	12	90	78	56	90	78	56	14
70	79	87	19	79	87	19	79	15
127	64	87	23	12	87	23	12	16
34	87	91	21	94	91	21	94	17
32	23	83	32	31	83	32	31	18
39	12	90	78	56	90	78	56	19
70	79	87	19	79	87	19	79	20
1	2	3	4	5	6	7	8	

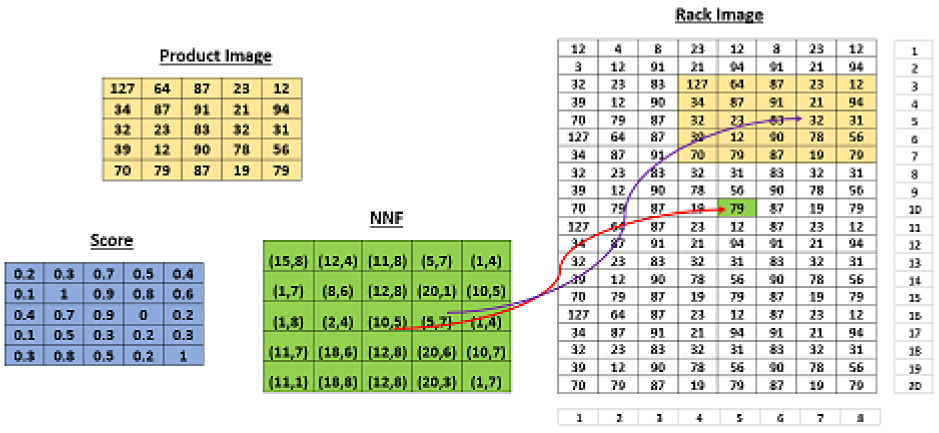
Figure 3.1: Initialization

Patch match is an iterative algorithm. Initialization step is done once but propagation and Random Search are done iteratively. In every iteration, we move from left to right and top to bottom (raster scan) covering all the pixels. Let there be a total of n pixels in the database image. In every iteration, P_1,R_1,P_2,R_2 . . . P_n,R_n this order is followed. Here P represents Propagation and R for Random Search.

3.2.2 Propagation

For every pixel, right and top pixels are its neighborhood. In the pixel neighborhood let's say the right one has the best score (Let it be R), then the current pixel is now mapped to left of NNF(R).

Figures 3.2 and 3.3 shows the two successive steps in the propagation algorithm. In the figure 3.2 Violet line represents a good match and red line a mismatch. Figure 3.3 shows how the propagation happened.



Product Image

127	64	87	23	12
34	87	91	21	94
32	23	83	32	31
39	12	90	78	56
70	79	87	19	79

Score

0.2	0.3	0.7	0.5	0.4
0.1	1	0.9	0.8	0.6
0.4	0.7	0.9	0	0.2
0.1	0.5	0.3	0.2	0.3
0.3	0.8	0.5	0.2	1

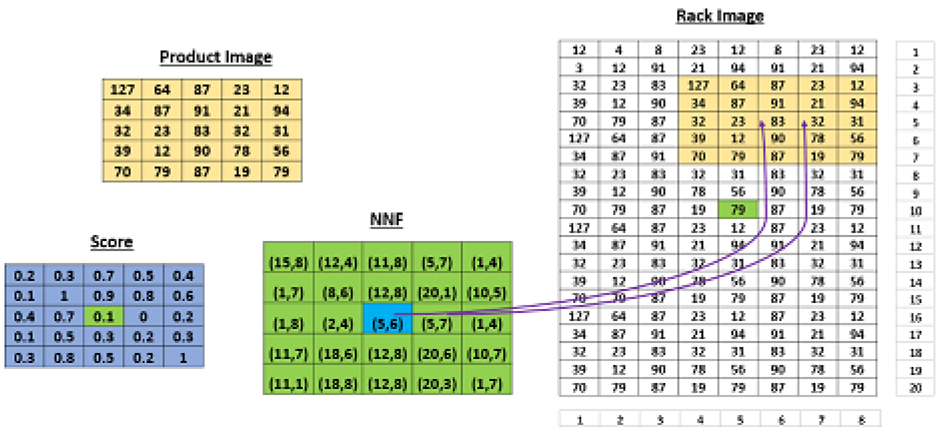
NNF

(15,8)	(12,4)	(11,8)	(5,7)	(1,4)
(1,7)	(8,6)	(12,8)	(20,1)	(10,5)
(1,8)	(2,4)	(10,5)	(5,7)	(1,4)
(11,7)	(18,6)	(12,8)	(20,6)	(10,7)
(11,1)	(18,8)	(12,8)	(20,3)	(1,7)

Rack Image

12	4	8	23	12	8	23	12	1
3	12	91	21	94	91	21	94	2
32	23	83	127	64	87	23	12	3
39	12	90	34	87	91	21	94	4
70	79	87	32	23	83	32	31	5
127	64	87	39	12	90	78	56	6
34	87	91	70	79	87	19	79	7
32	23	83	32	31	83	32	31	8
39	12	90	78	56	90	78	56	9
70	79	87	19	79	87	19	79	10
127	64	87	23	12	87	23	12	11
34	87	91	21	94	91	21	94	12
32	23	83	32	31	83	32	31	13
39	12	90	78	56	90	78	56	14
70	79	87	19	79	87	19	79	15
127	64	87	23	12	87	23	12	16
34	87	91	21	94	91	21	94	17
32	23	83	32	31	83	32	31	18
39	12	90	78	56	90	78	56	19
70	79	87	19	79	87	19	79	20
1	2	3	4	5	6	7	8	

Figure 3.2: Propagating a good match



Product Image

127	64	87	23	12
34	87	91	21	94
32	23	83	32	31
39	12	90	78	56
70	79	87	19	79

Score

0.2	0.3	0.7	0.5	0.4
0.1	1	0.9	0.8	0.6
0.4	0.7	0.1	0	0.2
0.1	0.5	0.3	0.2	0.3
0.3	0.8	0.5	0.2	1

NNF

(15,8)	(12,4)	(11,8)	(5,7)	(1,4)
(1,7)	(8,6)	(12,8)	(20,1)	(10,5)
(1,8)	(2,4)	(5,6)	(5,7)	(1,4)
(11,7)	(18,6)	(12,8)	(20,6)	(10,7)
(11,1)	(18,8)	(12,8)	(20,3)	(1,7)

Rack Image

12	4	8	23	12	8	23	12	1
3	12	91	21	94	91	21	94	2
32	23	83	127	64	87	23	12	3
39	12	90	34	87	91	21	94	4
70	79	87	32	23	83	32	31	5
127	64	87	39	12	90	78	56	6
34	87	91	70	79	87	19	79	7
32	23	83	32	31	83	32	31	8
39	12	90	78	56	90	78	56	9
70	79	87	19	79	87	19	79	10
127	64	87	23	12	87	23	12	11
34	87	91	21	94	91	21	94	12
32	23	83	32	31	83	32	31	13
39	12	90	78	56	90	78	56	14
70	79	87	19	79	87	19	79	15
127	64	87	23	12	87	23	12	16
34	87	91	21	94	91	21	94	17
32	23	83	32	31	83	32	31	18
39	12	90	78	56	90	78	56	19
70	79	87	19	79	87	19	79	20
1	2	3	4	5	6	7	8	

Figure 3.3: Propagation

3.2.3 Random Search

We try to improve the NNF by doing a random search. Let p be the current pixel. We will take k radius values from an exponential distribution. For every radius around the pixel we will randomly select one point (thereby patch). We will update the NNF if any of the k patches result in a better matching score.

We stop the algorithm after fixed number of iterations. In every iteration, propagation and random search are done for every pixel from left to right and top to bottom. Generally the convergence is achieved in 5-10 iterations, although it may sometimes get stuck at local optima. For odd iterations the neighborhood is left and bottom pixels and for even iterations it is right and top pixels.

3.3 Algorithm

In this section we explain how Patch Match can be used in our algorithm. We re size the rack image to various scales and run patch match in all the scales as HOG descriptor is not scale invariant . 10 scales are uniformly chosen from 0.7 to 1.5. We will accept the scale for which the mean of score matrix is the least. Once the scale is finalized we put a threshold (0.25*max value) and reject the matches whose score is above the threshold. After finding the matches we continue with the steps discussed from Section 2.3 to Section 2.5.

Chapter 4

Experiments and Results

In this chapter we present the results of the proposed algorithm. Section 4.1 describes about the data set where as Section 4.2 has the final results.

4.1 Data Set

Here we provide a glimpse of the data set we have used. We have two types of images product images and rack images. There are a total of 25 product images and 20 rack images. Although this is a small data set we chose this because it covers most of the challenges we have mentioned before. The product images are taken under ideal lighting conditions. Figure 4.1 has few example product images.

The rack images on the other hand are natural images and were taken in different lighting conditions. The number of shelves vary from 1-5 on typical rack image although the algorithm works fine on any number of shelves. The change in the 3D viewpoint of the rack image makes the detection hard but our algorithm is invariant to 15 degrees change of angle. The rack image with few variations can be seen in Figure 4.2.

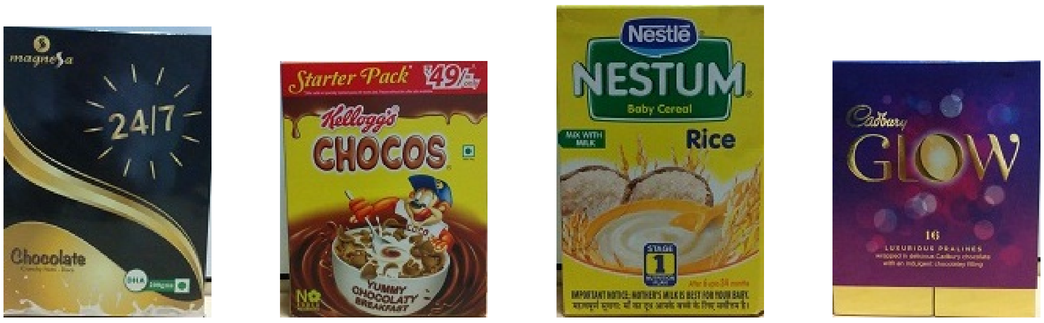


Figure 4.1: Sample product images

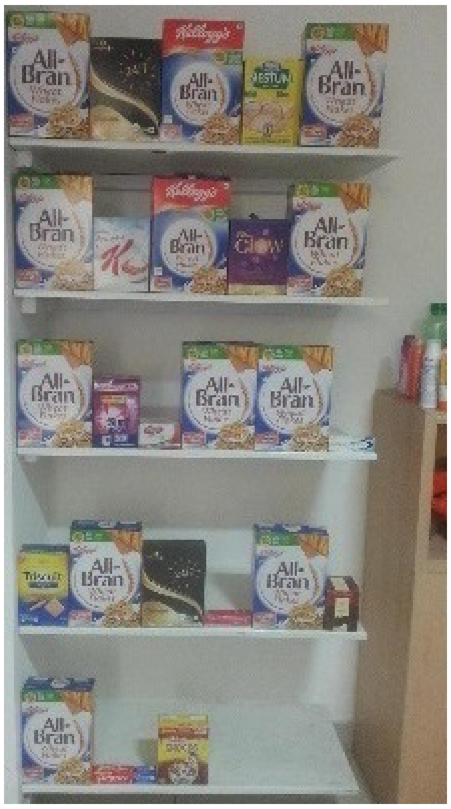


Figure 4.2: Example rack Image

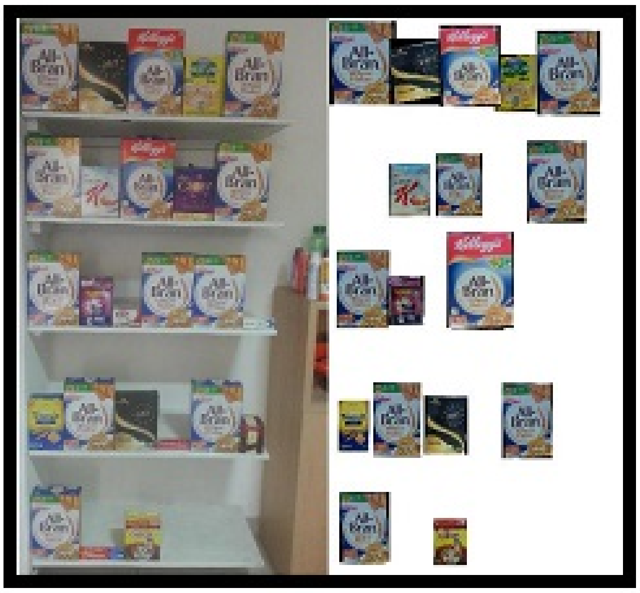


Figure 4.3: Rack image with reconstructed output

Figure 4.3 shows the reconstructed output after running the algorithm. We can see that the input image skewed. But that doesn't have effect on recognition of the products. This output is only after the SIFT matching.

4.2 Results

In this section we show our present the accuracy of our algorithm. For the 20 selected rack images number of objects detected has been shown tabulated. We compare our algorithm with the last year thesis work on this problem.

Image Number	Total Products	Previous method	SIFT	SIFT+Patch Match
1	25	18	19	23
2	25	16	17	21
3	25	18	19	20
4	25	18	18	21
5	25	15	16	19
6	25	19	19	23
7	25	18	19	20
8	25	17	18	20
9	25	17	17	20
10	25	17	18	21
11	25	16	16	20
12	25	16	17	21
13	25	16	17	21
14	25	16	16	20
15	25	15	17	20
16	25	15	16	19
17	25	17	18	20
18	25	17	19	20
19	25	17	16	20
20	25	15	15	19
	Mean Accuracy	66.6	69.4	77.8

Chapter 5

Conclusions and Future Work

In this report we described an algorithm using SIFT to identify the products in a rack image. We improve it using the patch based matching algorithm. The objective at the start of this project is design an algorithm to identify the products without taking any user input. We have done that successfully. It is also necessary to make the algorithm run after

Although the described algorithm has good accuracy it doesn't scale well. We tested our algorithm with a small database of 25 product images and the algorithm takes 10 minutes to generate the output on a machine with 4 processors and 3.4 Ghz processing speed. Even in a small store number of products will be much higher than 25 so our algorithm will need hours to generate the output. So answer this question k-d trees can be used to find fast nearest neighbors. Aucliar et al. in [1] uses locally sensitive hashing to even speed the nearest neighbors computation. These can be added to the present work to generate the outputs quickly.

Bibliography

[1] Adrien Auclair, Laurent D. Cohen, and Nicole Vincent. *How to Use SIFT Vectors to Analyze an Image with Database Templates*, pages 224–236. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[2] Ruihan Bao, Kyota Higa, and Kota Iwamoto. *Local Feature Based Multiple Object Instance Identification Using Scale and Rotation Invariant Implicit Shape Model*, pages 600–614. Springer International Publishing, Cham, 2015.

[3] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), August 2009.

[4] Connelly Barnes, Eli Shechtman, Dan Goldman, and Adam Finkelstein. The generalized patchmatch correspondence algorithm. *Computer Vision–ECCV 2010*, pages 29–43, 2010.

[5] J. P. Bigham, C. Jayant, A. Miller, B. White, and T. Yeh. Vizwiz::locateit - enabling blind people to locate objects in their environment. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pages 65–72, June 2010.

[6] Timothy Chong, Idawati Bustan, and Mervyn Wee. Deep learning approach to planogram compliance in retail stores.

[7] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[8] Martin Ester, Hans peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[9] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[10] Annalisa Franco, Davide Maltoni, and Serena Papi. Grocery product detection and recognition. *Expert Systems with Applications*, 81:163 – 176, 2017.

[11] Marian George and Christian Floerkemeier. *Recognizing Products: A Per-exemplar Multi-label Image Classification Approach*, pages 440–455. Springer International Publishing, Cham, 2014.

[12] Marian George, Dejan Mircic, Gabor Soros, Christian Floerkemeier, and Friedemann Mattern. Fine-grained product class recognition for assisted shopping. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 154–162, 2015.

[13] Rabia Jafri, Syed Abid Ali, Hamid R. Arabnia, and Shameem Fatima. Computer vision-based object recognition for the visually impaired in

an indoors environment: A survey. *Vis. Comput.*, 30(11):1197–1222, November 2014.

[14] I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[15] S. Liu, W. Li, S. Davis, C. Ritz, and H. Tian. Planogram compliance checking based on detection of recurring patterns. *IEEE MultiMedia*, 23(2):54–63, Apr 2016.

[16] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[17] M. Marder, S. Harary, A. Ribak, Y. Tzur, S. Alpert, and A. Tzadok. Using image analytics to monitor retail store shelves. *IBM Journal of Research and Development*, 59(2/3):3:1–3:11, March 2015.

[18] M. Merler, C. Galleguillos, and S. Belongie. Recognizing groceries in situ using in vitro training data. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007.

[19] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–257–II–263 vol.2, June 2003.

[20] R. Okada. Discriminative generalized hough transform for object dectection. In *2009 IEEE 12th International Conference on Computer Vision*, pages 2000–2005, Sept 2009.

[21] H. J. Seo and P. Milanfar. Training-free, generic object detection using locally adaptive regression kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1688–1704, Sept 2010.

[22] Gül Varol and Rıdvan S. Kuzu. Toward retail product recognition on grocery shelves, 2015.

[23] T. Winlock, E. Christiansen, and S. Belongie. Toward real-time grocery detection for the visually impaired. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pages 49–56, June 2010.

[24] E. Yörük, K. T. Öner, and C. B. Akgül. An efficient hough transform for multi-instance object recognition and pose estimation. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 1352–1357, Dec 2016.

[25] Yuhang Zhang, Lei Wang, Richard Hartley, and Hongdong Li. *Where's the Weet-Bix?*, pages 800–810. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[26] S. Zickler and M. M. Veloso. Detection and localization of multiple objects. In *2006 6th IEEE-RAS International Conference on Humanoid Robots*, pages 20–25, Dec 2006.