Indian Statistical Institute, Kolkata

M. Tech. (Computer Science) Dissertation

On the identification of grocery products

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

Author:
Nishant Kumar
Roll No: CS-1409

Supervisor:
Dr. Dipti Prasad Mukherjee
ECSU Unit, ISI

M.Tech(CS) DISSERTATION THESIS COMPLETION CERTIFICATE
Student: Nishant Kumar (CS1409)
Topic: Product Recognition
Supervisor: Dr. Dipti Prasad Mukherjee

This is to certify that the thesis titled "***On the identification of grocery products***" submitted by **Nishant Kumar** in partial fulfillment for the award of the degree of Master of Technology 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: Dr. Dipti Prasad Mukherjee

Dedication

This dissertation is dedicated to the sweet memories of my dear friend Late Mr. Sahukara Harsh Choudhary who pushed me to study Computer Science and suggested working in the field of Computer Vision because he considered it to be the easiest as one could literally 'see' the results of any algorithm.

Acknowledgements

I would like to thank my dissertation supervisor Dr. Dipti Prasad Mukherjee for agreeing to guide me and giving me this opportunity to work on a problem with direct real world application. Working with him has been an amazing experience.

I would like to thank my lab mates, my batch mates and my family for being there for me when I needed them. Without the mind-blowing ideas and the motivation provided by Archan, this would have never been possible. I would like to thank my friends from the M.Tech batch, most notably Satish and Rahul for their valuable inputs.

Abstract

In this thesis, a methodology is proposed to identify the products present in a rack in retail stores. Only a single image of each category of products is given to us. A series of preprocessing algorithms are applied on the input images in order to assist the matching algorithms used. Our methodology involves a two layered approach to solve the problem. The first layer is used to shortlist the products in order to reduce the search space for the second layer. The products shortlisted in the first layer are used as nodes in a graph. A scale-space based approach involving bag-of-words model is used to provide the feedback. The results obtained are shown to be comparable to the existing state-of-the-art algorithm in this field.

Contents

List of Figures

Chapter 1

Introduction

The major retail chains in the world face the problem of placing products in appropriate locations. In a small store, this can be taken care of manually by the shop owner. But as the store in question tends to a huge marketing space like a mall, this problem becomes something which cannot be handled by pure gut and experience. It is here that Statistics and Computer Science come to play an important role. In order to decide on where to place the products, several parameters are to be taken into consideration. One set of these parameters are those which are used to target specific customer groups, an example being women-specific products to be placed preferably at a height which is close to the average height of women in that geographical area. Placing relevant goods nearby might be another criterion, with the target being boosting sales of a particular brand. A good example could be a snacks manufacturer who would like to place his brands to be placed right next to the beverage section. A systematic placement of goods in the outlet also ensures that the movement of the consumer is minimized. One should not expect to find bread and milk in distant corners of the store. There are several other parameters to be taken into consideration.

Once a plan is prepared based on the parameters discussed above is agreed to, the goods are placed according to the plan. The next step involves the inspection process to determine whether everything is kept according to the plan. The products may be wrong location either because of human errors on part of the outlet workers or because of the action of consumers since they rarely bother keeping something back at the right location from where it was picked up. The manual testing can be very complicated and time consuming. It is here, where the work done in this dissertation comes into play. This thesis is aimed at solving the problem of identification of objects in a shopping mall. Given an image of a shelf in a shopping mall, the challenge is to identify which products are present in the shelf. In addition to this, the location of the product is also to be determined. In short, the question being answered is-"Which product is placed at which location?"

1.1 Challenges

The challenges in solving this problem can be divided into several headings, as discussed below. In this section, the challenges are highlighted while in the following section, the various approaches taken to counter the challenges are discussed.

1.1.1 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 skwewed image of the shelf. This can be explained using a flowchart as shown in the Figure 1.2.

1.1.2 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. A few examples of the color differences as observed in the dataset are shown below:

1.1.3 Differences in quality between 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 2000X ¡something¿, while the same camera can be used to capture high resolution images of the products, when photographing them one at a time.

1.1.4 Location of products not known

The location of the products are not known. So, we proposed an exhaustive searching strategy, where we crop for different products throughout the image. This is used to shortlist the products to reduce the search space for the second level of algorithms. This will be discussed in detail in the Section 2.3.

1.2 Overcoming the challenges

Several challenges come up once we analyse the images given to us. The most important amongst them are discussed in this section. The skewness of the images, the small size of the images, lack of training data and illumination changes are the ones which are discussed in this section.

1.2.1 Unskewing the Images

A typical image of the rack in our dataset resembles like the one shown in Figure 1.1a. The first step is to unskew the images and make them rectangular. The need for unskewing arises from the fact that if we decide to go ahead by cropping the images out from a skewed image, we end up getting cropped images similar to the ones shown in the Figure 1.1b and Figure 1.1c. We notice that if we crop images in manner, we would be relying heavily on our matching algorithms. So, in order to take some pressure off our matching algorithms, we make an attempt to rectify our images itself as part of a pre-processing. A manual intervention is required in the beginning, where the user is asked to select the 4 corner points of the image of the rack in question. This is the only manual intervention required. Thereafter, all the steps are automated.

(a) Original input image (b) Crop_1 (c) Crop_2

Figure 1.1: Products cropped from a skewed rack image

1.2.2 Handling the color variations

Color Quantisation Based Approach

Since the images in our dataset are color images, the first cue was to use the color information to match the images. The challenge as discussed in subsection 1.1 above limited the use of color information. The different shades of a color varied widely in pixel intensities with the changes in the color of incident light. The answer lied in color quantisation, as inspired by

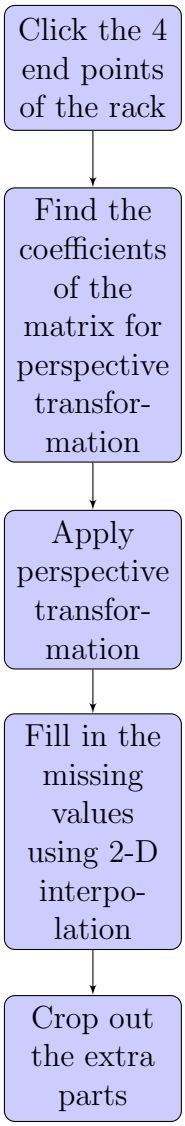


Figure 1.2: Steps in unskewing of *rackImage*

the . The entire gamut of 16 million colors was divided into a set of just 16 colors, these 16 being the ones selected uniformly and equidistant from each other within the 256*256*256 colors. Now, each of the RGB pixel in the original image goes to a particular bin based on the euclidean distance nearest neighbour. This will be clear from the diagram shown in the figure 1.3. Once the colors are quantised, the output looks something like the examples shown in Figure 1.4.

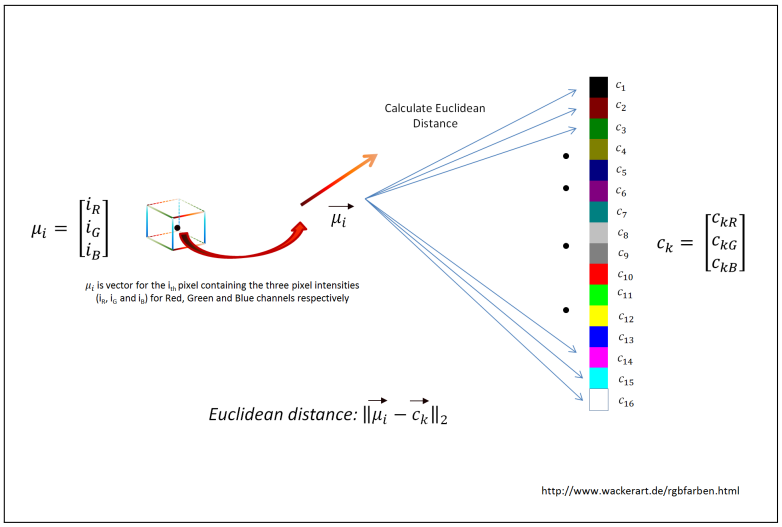
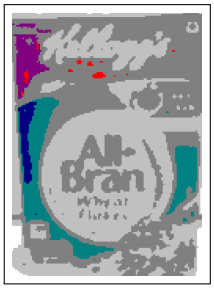


Figure 1.3: Quantisation of colors into 16 standard colors based on Euclidean Distance

(a) Query image



(b) Quantised query image

(c) Target image

(d) Quantised target image

Figure 1.4: Examples of color quantisation

Once the quantization of colors is done, we move on to establish a relationship between the colors present at different pixels in the image. One such approach is to create the color-co-occurrence histograms, to keep track of the distance between a pair of color locations, as proposed by Chang and Krumm [17]. A tri-variate histogram is constructed based on

three parameters C_i, C_j and d_{ij}. At an abstract level, the framework of the matching using color-cooccurrence histogram is shown in Figure 1.6.

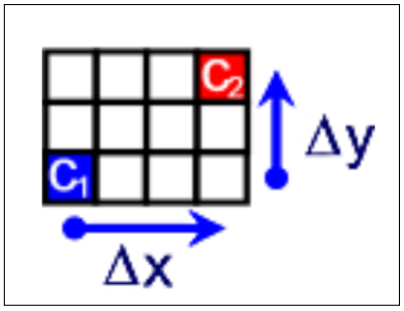


Figure 1.5: Two color pixels C_1 and C_2 located at a distance $d_{ij} = \sqrt{\Delta x^2 + \Delta x^2}$. Image taken from [17]

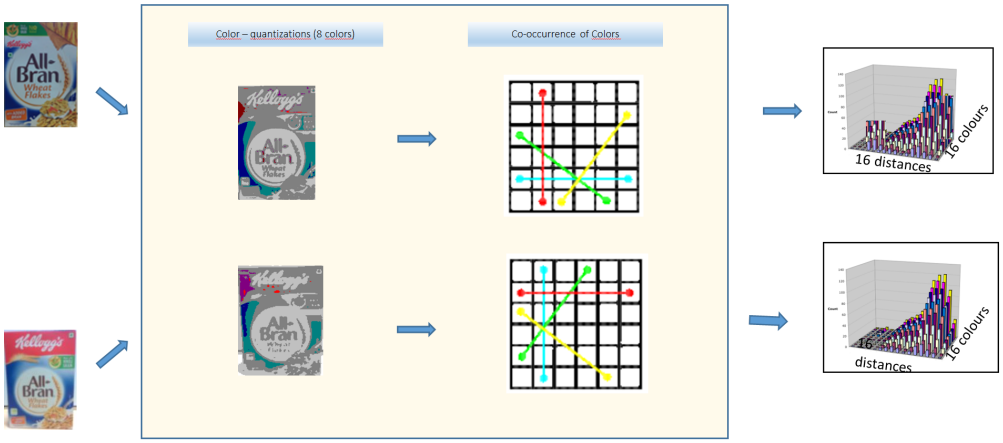


Figure 1.6: Methodology of the color-occurence histograms

Exploring different color spaces

In order to counter the illumination invariance, a shift from the RGB color space to a different color space was explored. One such promising color space was the l4-l5-l6 space. Assuming the dichromatic reflection and white illumination, the l4-l5-l6 space, as proposed by Gevers *et al.* [11] has been proven to be independent of viewpoint, orientation of the surface and direction of the incident light. Any pixel in the RGB color space is transferred to the l4-l5-l6 color space as given by the equations below.

$$l_4(R,G,B) = \frac{|R-G|}{|R-G|+|B-R|+|G-B|}$$

$$l_5(R,G,B) = \frac{|R-B|}{|R-G|+|B-R|+|G-B|}$$

$$l_6(R,G,B) = \frac{|G-B|}{|R-G|+|B-R|+|G-B|}$$

Towards rotation and scaling invariant features

Once we are done with this, rotation and scaling invariant operators based on angles between two sides of a triangle formed by taking three points at a time and projection invariant using concepts of cross ratio by selecting five points at a time. However the skewness of the image in our case was already taken care of, as discussed in 1.2.1, so the projective invariance part was not required. Once done with these, a multi-variate histogram H_4(with 4 variables) was constructed. The value accumulated in each bin of the multi-variate histogram H_4 represents the number of times three different colors form an angle θ between them. Once the histogram H_4 is constructed for each of the two images(the target and the query image), the overlap between the two is taken as the score for match between the images.

1.2.3 Towards a machine learning based approach

In order to use any machine learning based approach in our problem, we had to cater to two problems:

1) Convert each image to a vector.

2) Obtain training data for training our model, be it Support Vector Machine(SVM), Multilayer perceptron(MLP), Random Forests etc.

The first issue was taken care of by converting the image to a vector of length 384, as shown in 1.2.3 while the second problem was solved using the mechanism discussed in 1.2.3

Generating synthetic data

Since, we are given only one instance of the target image, we need to generate the different instances of the image synthetically in order to train our model. It was noticed that the brightness changes, the changes in quality and the changes in size were the three major differences between two instances of the same image. So, in order to simulate these conditions and generate synthetic data, we use the following three methods:

1) Brightness changes approximated by incrementing/decrementing all the pixel intensities of the image by a constant value k. This is done for several values of k, (say p times, so that for each $k_i(1 \leq i \leq p)$, we get a copy of an image with a certain level of brightness

2) The changes in the quality of the images are simulated using gaussian blurring with varying values of σ. Gaussian kernels of varying sigma(σ_i, where $1 \leq i \leq q$) values are convoluted with the image to generate q different versions of the image, each with a different level of smoothing.

3) Since we are not sure about the exact scale of the image as present in the query image but have an estimate of the same, so we generate r different sizes of the image, at

different scales. For example, if the estimated height of the image is 200 pixels, we might consider taking 5 different scales 180,190,200,210 and 220 pixels in height, while maintaining the aspect ratio of the image.

The three steps mentioned above yield a total of $(p * q * r)$ images in place of one image. The values of p, q and r are taken such that the total number of samples generated are close to 1000. A value of 30 for each one of these was able to produce reasonable results while at the same time having a practically feasible training time.

Converting the image to a vector

The color quantisation into 16 VGA colors is carried out as discussed in the Subsection 1.2.2. In order to localise the features, the image was split into 24 parts Each of the 16 parts has its own histogram of 16 bins for each of the colors quantised. The histograms are normalised and appended one after the other. Thus, we come up with a 24*16 (=384) length vector for one image. These vectors will be fed into the SVM. A pictorial representation of the methodology to convert the image to a vector is shown in Figure 1.7.

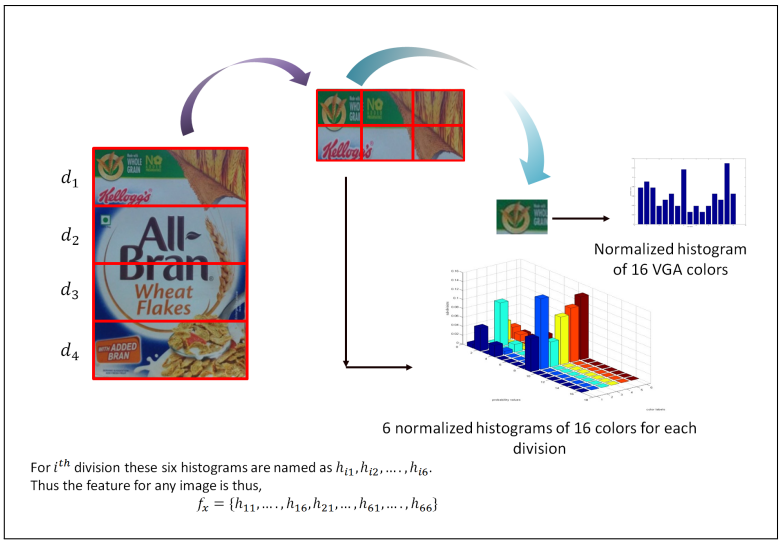


Figure 1.7: Converting an image to a vector

1.2.4 Identifying the location of products

Since the location of the products were not known an exhaustive searching method was proposed. The query images, were resized several scales, shifted for vertical localisation errors (in the Y-direction). Each time, we crop for all the possible products in the database and compare with the corresponding *productImages* using correlation. Throughout this thesis, *productImage* is referred to the images taken from the product image database, which are typically good quality high resolution images. The exhaustive searching methodology has been discussed in detail in Section 2.3.

1.2.5 Exploring the scale-space based algorithms

Keypoint feature based algorithms like SIFT [5], SURF [9] and were tested and marginally better accuracies were obtained. A counter intuitive observation was that the performance was noted to go down with color-SIFT [2]. The reasons for poor performance of these algorithms was analysed and three main problems were found to occur:

1) Huge variations in the number of *keypoints matched* [13].
2) The size of the masks used in these algorithms to approximate the *Laplacian of Gaussian* masks, were found to be too large for some of our images.
3) The poor performance of the color-SURF was that the algorithm was unable to handle such extreme variations in color.

1.3 Related work

A plethora of work has been done in the field of object detection and recognition from color images and the extension of these works in the field of product recognition [18] [7]. In [4], the authors compared several algorithms and evaluted their performance on a wide variety of images, while at the same time proposed the metrics to be used for perfomance analysis. Binary classifiers based on the on bag of words model [12] have been used to decide whether an object belongs to the a given category or not. Several important works in the field have been done in the recent past in the specific field of product recogniton. In [8], George and Floerkemeier used a "per-exemplar multi-label image classification" to provide an efficient and fast classification. In [14], Merler *et al.* compared several known algorithms so as to measure their performance quantitatively in case of grocery images and also analysed the failure cases of these algorithms. In [24], the authors worked towards the development of a real time grocery scanner for the visually impaired. Most of the works in this field are done using trained models based on a huge dataset. The methodology proposed in this thesis does not require any sort of training.

1.4 Contributions

The contributions made by this thesis include the application of Directed Acyclic Graphs to find the optimum placement of products in a shelf. We use a two layered approach to find the correct products. The first layer involves shortlisting the products and the second layer involves providing a feedback to improve upon the Graph generated in the first layer. Initially, SURF [9] was used at the second level to provide a feedback mechanism to the Graph. Later a modified version of SURF-based bag of words model was used to provide a feedback to the Graph. This modified model is based on image specific sub-sampling of the scale space. The shelf detection algorithm might also be considered to be a minor contribution. The problem is solved without any trained model since our dataset has a single instance for every product.

1.5 Proposed approach

After exploring all the above mentioned algorithms, it is concluded that any one of these algorithms is not sufficient in its stand-alone version to solve our problem. We need to limit the search space for these algorithms. In order to accomplish this, an idea based shortlisting the products in the first round and then going for a second level of matching, using any of these algorithms. Moreover in cases where a prior estimate of the location of the product was not available, we depended solely on the two level approach to solve the problem. The products were shortlisted based on the correlation values and later on a Graph was constructed incorporating the key point based features and the shortlisted products. This approach will be discussed in detail in the next chapter. The overall flow diagram for the proposed two layered approach is shown in the Figure 2.1.

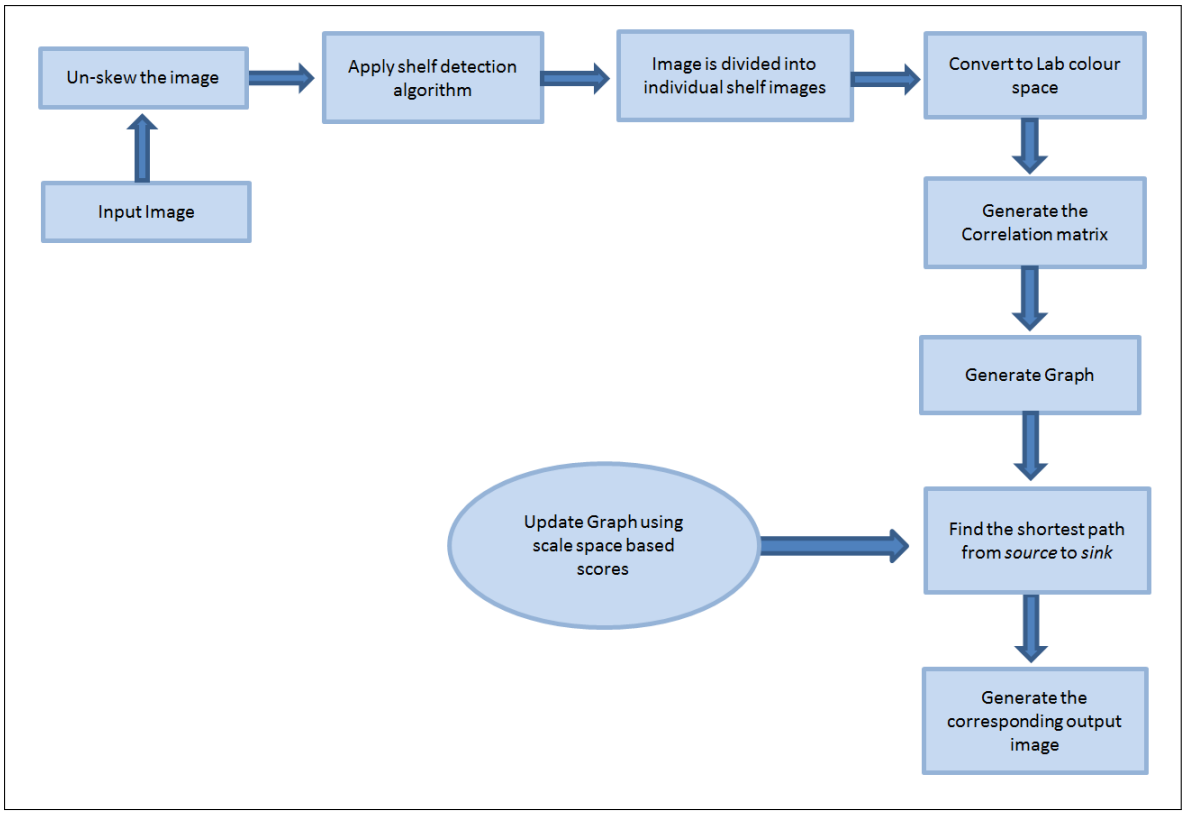


Figure 1.8: Block Diagram showing the steps in the two layered approach

Chapter 2

Methodology of the proposed approach

The block diagram shown in Figure 2.1 summarises the proposed methodology. In this Chapter, we discuss the steps in detail. We start with discussing the preprocessing techniques which include the unskewing of the images and the slef detection algorithm. Then we move on to discuss our exhaustive search based shortlisting of product. Thereafter, we discuss the construction of the Graph, followed by the feedback system for the Graph. The feedback system is based on the bag-of-words model constructed using the keypoints derived from SURF [9]. We also explore the sub-sampling of the scale space and incorporate it into our algorithm.

2.1 Unskewing the images

As shown in the block diagram in figure 2.1, the first step in the proposed methodology is to unskew the image. The input image given to us looks like the image shown in Figure 2.1a. It is observed that the straight lines remain straight but the parallel lines are no longer parallel. So, we go for the projective transformation equations. The equation contains 8 unknowns, hence a set of 8 equations are required to solve them. Each point in a 2D space gives us two sets of co-ordinates x and y, hence the user is prompted to select 4 points from the image manually. The 4 points which are to be selected are the corners of the rack in the input image. Since a rack is always supposed to be rectangular, hence we map these 4 points to a rectangular of the same size. Thus, we are able to solve the 8 unknowns in the projective equation. Each pixel of the input image is now transformed to a new location. Thus, we obtain an image as shown in the Figure 2.1b. The unskewed image output image typically resembles the one shown in the Figure 2.1b. A flow diagram for the unskewing of the images has been shown earlier in the Figure 1.2.

(a) Input Image

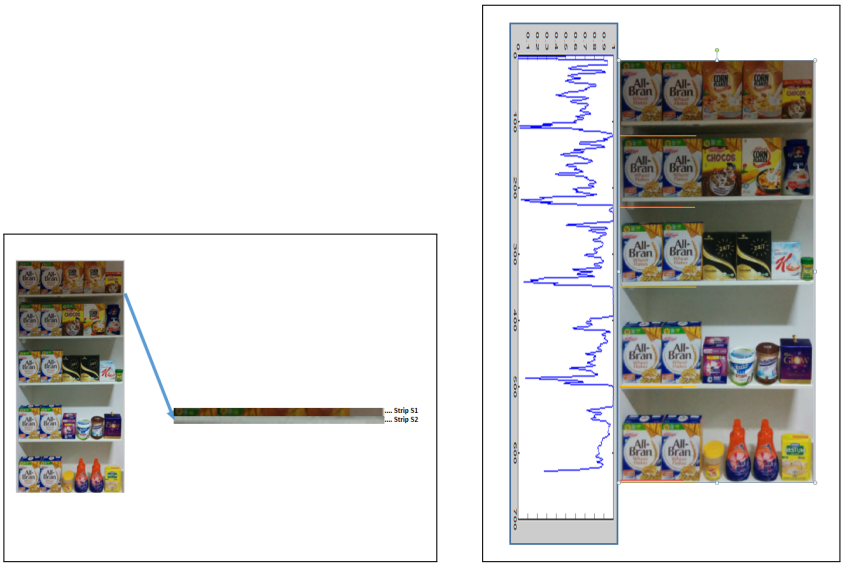
(b) Unskewed Image

(c) Cropped out image

Figure 2.1: Block Diagram showing the steps in the two layered approach

2.2 Shelf detection algorithm

The second step in the identification of shelves automatically. The *Shelf Detection Algorithm*, as proposed in this thesis is based on the output of the step mentioned the previous section 2.1. Once the images are unskewed, the shelves become horizontal. It was noticed that since the products placed on the shelves cannot be hanging in the air, so a sudden change in the intensity of the image is observed just at the starting of the shelf. This particular property was utilized in determining of the shelves. It should be noted that since we are aware of the height of the shleves and also the height of rach rack inside the shelf, so we have an estimate of the location of the starting and ending point of each shelf. In order to determine the shelves, we use the 2-D correlation coefficient. We crop out thin horizontal strips from the image, the thickness of the strips being t. The value of t, is estimated using the global scale, as discussed in the section 3.1. The strips are selected one after the other. The cropping is done consecutively as shown in the Figure 2.2a.



(a) Cropping of strips (b) Steps in shelf detection

Figure 2.2: Logo after scaling and rotation

Let us refer to the two consecutive strips cropped out as $S1$ and $S2$. We convert both of these strips to grayscale. Let us refer to these two strips as Sbw_1 and Sbw_2. Now, we proceed to find the correlation coefficient between these two strips. The 2D- correlation coefficient c_{12} between two 2D-matrices A and B is defined in eq 2.1.

$$c_{12} = \frac{\Sigma\Sigma((A - \bar{A})(B - \bar{B}))}{\sqrt{(\Sigma\Sigma(A - \bar{A})^2)(\Sigma\Sigma(B - \bar{B})^2)}} \tag{2.1}$$

where $\bar{A}$ and $\bar{B}$ are the means of the distributions A and B respectively.

We do this for each consecutive pair of strips in the unskewed image and end up with a set of values of the correlation coefficient. If these values are plotted against the image,

it is noticed that a peak is invariably present at the location of the ending point of a shelf, as shown in Figure 2.2b. Now, in order to use this information, we needed to select the local maximas in this plot. But defining a local maxima for our purposes was difficult. So, the idea of the absolute maxima in a neighbourhood was proposed. We define a search neighbourhood around the estimated location of shelves. The height of this seach space is taken to be equal to the height of the shelf which has the minimum height. This is done in order to ensure that there is only one maxima in that neighbourhood and this maxima corresponds to the location of the shelf.

The *Shelf Detection algorithm*, proposed here is found to work fine with the simulated datasets, with almost 100% accuracy. However, its performance went down in case of the real world datasets. In those cases, we have an option for the user to select the shelves manually(locate each shelf with a mouse click), just in case the rack detection output was found to be erroneous.

Figure 2.3: Correction in the location of shelves

Once we are done with detection of the shelves, whether automatically or by manual intervention, we are able to separate the rack into individual shelves. Henceforth, all the processing will be done on individual shelves, hence forth to be referred to as the *croppedShelf*. It should be noted that the processing done on each shelf is independent of the other, hence they can be processed in parallel. Once the shelves are solved individually, the results from all of them can be appended to get one complete reconstructed image of the shelf. Each of the shelf was converted to the CIE L*a*b [3], before proceeding to the steps discussed in the forthcoming sections. The conversion was done using the MATLAB function and the model used was the srgb model. A typical shelf separated and converted to the CIE L*a*b color space is as shown in Figure 2.4

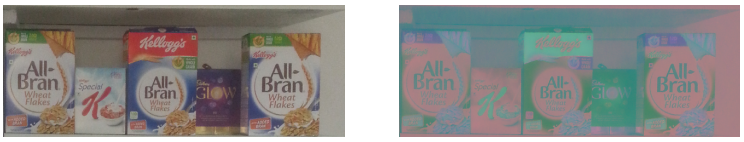


Figure 2.4: Cropped-out shelf converted to the CIE L*a*b color space

2.3 Exhaustive searching to shortlist the products

Every alternate pixel position along the x-axis(the horizontal) is scanned. Let the total number of *productImages* be n. So, at any pixel position x, we scan for n images. Let each of the database image be varied in m different sizes, as discussed in subsection 1.2.4. In addition to that, as we are not sure of the location of the products in the vertical direction. Since a product cannot be hanging in the air so ideally, the products start at the bottom of the shelf. But due to varied camera angles some products might appear to be starting at a certain height above the bottom of the *croppedShelf*. So, in order to take care of such conditions, we crop at varying heights above the bottom of the *croppedShelf*. Let us assume that we crop for p locations in the y-direction(the vertical), moving up by 2 pixels each time. Thus, at each location, we need to crop for $n * m * p$ different images. A cropped image is compared with the corresponding *productImage*. At each position, we begin by cropping a region from the shelf image. The cropping is done according to the height and width of the product from the database times the scaling factor. The scaling factor is estimated using the height of the shelf in c.m and the height of the *croppedShelf*, in pixels. Once cropped, both the *croppedImage* and the corresponding *productImage* are converted to the CIE L*a*b* color space. Now, the correlation coefficient between the cropped image and the database image is calculated in each of the three layers L*, a* and b*. Thus, we have three correlation coefficients, say c_1, c_2 and c_3, for the L*, a* and b* respectively. Finally we get one value of c for a pair of images, by taking the average of the three values.

$$c = \frac{c_1+c_2+c_3}{3}$$

At each of the x-location, hence we have a set of $(n * m * p)$ correlation values. We next obtain the top 500 values from these scores. We sum the values for each of the n products from the top 500 values. We say the products with highest values among the cumulative scores of the products are more probable to be present in that exact x-position. We take the top 3 products. The score assigned to each of the products is its respective cumulative correlation score, hence forth referred to as *corrScore* and at each point in the horizontal direction(the X-axis) we have a maximum of three *corrScores*. It should be noted that fewer than three corrScores are obtained in cases where the correlation coefficients from only one or two of the products fell into the top 500 bracket.

2.4 Creation of a directed acyclic graph

Now, we will be using these three values of the *corrScore* at each location to construct a Graph. Let G be the graph constructed using the three values of *corrScore* at each location in the horizontal direction. For each column we have 3 nodes in the graph, which represent the 3 products in the column. Thus if there are a total of q locations in the x- directions, the graph has $3 * p$ nodes. We add a source S and a sink node T to the graph, hence the total number of nodes is $3 * p + 2$. There exists a directed edge $E(S, n_i) = \varepsilon$, where S is the source

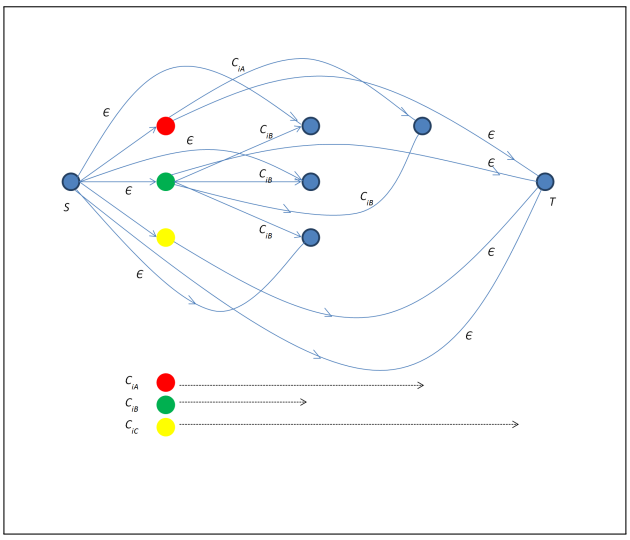


Figure 2.5: Graph(G) generated based on correlation matrix

and n_i is any other node excluding the source. There also exists a directed edge $E(n_i, T) =$, where n_i is any node in the graph except the sink T. Let at any i_{th} position the score of a product P be denoted as P_i, the width of product P be denoted as width(P), the node for product P in i_{th} position be denoted as n_{iP} and the position of i_{th} column in the shelf image be pos(i). There exists a directed edge $E(n_{iP}, n_{jQ})$ if and only if $pos(j) - pos(i) \geq width(P)$, such that $E(n_{iP}, n_{jQ}) = C_{iP}$. After assigning the weights, the graph looks as shown in the Figure 2.5. The red, the green and the yellow dots are products in i_{th} position. The straight black lines at the bottom shows the width of each product (after pixel to column number conversion).

Thus the construction of our Directed Acyclic Graph(DAG) from the *corrScore* is complete. We then obtain the maximum weighted path in this graph. One needs to understand why we intend to find the maximum weighted path in this graph. The sole reason is that we expect the sum of the correlation scores for the selected products to be the maximum. One can argue that this is dependent on the position of the products being placed in the final image. It is true, but any misplacement will lead to the sum being lower that any correct placement. Also the fact that the correct product almost always occur among the top 3 in any column helps in justifying the fact. It should be noted here that obtaining a maximum weighted path in any graph is an NP-hard problem [10]. The only solution is to negate the weights assigned to each of the edges and obtain the minimum weighted path. There are many existing algorithms which does this job efficiently. The ones which were tested upon are Bellman-Ford (time complexity O(N*E)), BFS (time complexity O(N+E)), Acyclic (time complexity O(N+E)), and Djikstra (time complexity O(log(N)*E)). A detailed description about all these algorithms can be found in the chapter 24, Single Source Shortest Paths, in the book by Cormen et al [21]. The shortest path returns which product starts at what location. This indicates that we can obtain a solution at this stage. But if we go ahead to generate the shortest path from the Graph G, we get poor results. A typical result is as shown in the Figure 2.7.

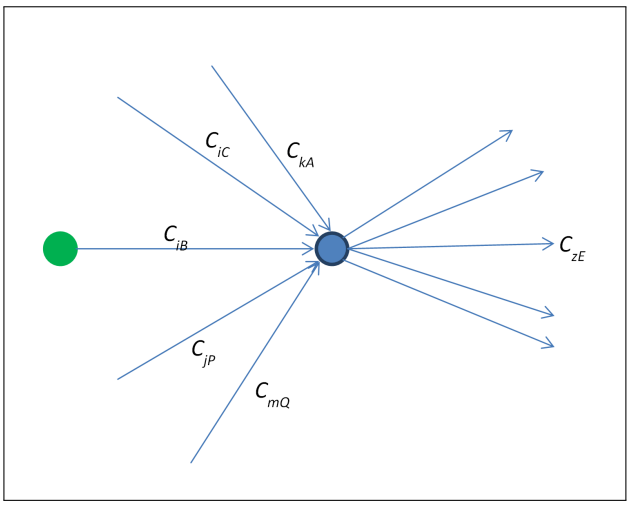


Figure 2.6: A portion of Graph(G) without any feedback system

Figure 2.7: Result generated using the shortest path

2.5 Incorporating a feedback mechanism in the graph

In the previous section we see that even though we are able to identify the correct products among the top 3 products at any location on the x-axis(the horizontal direction), the results obtained are poor. This is because the correct product always do not have the highest correlation score in the correct position. Thus, there arises a need to boost the correlation score for the correct products while suppressing the scores of the other products. In order to analyse this we need to revisit our Graph G. Let us consider a small portion of the G as shown in the Figure 2.6.

It can be observed in this case that the blue node has not informed its predecessor nodes that whether it(the blue node) was a correct choice or not. Thus, we sense a need for a feedback system to inform the previous nodes in a path, about how the choice made at one particular position affected the performance later in the path. We build our own feedback mechanism in which we supply each node(say node N) with a new score S_{ze} which will serve as a response to the previous nodes in the path from the present node(node$N$). This score(Sze) is multiplied to the previous nodes to boost/diminish the nodes importance in the path. Thus the Graph(G) has been modified to form Graph(G'), as shown in the Figure 2.8. Earlier it has been discussed in Subsection 1.2.5, that the performance of the scale-space based algorithms faced a major problem of thresholds when the search was performed amongst a large number of products. But now after shortlisting three products at each location, we are in a position to use these algorithms. The selection of the scale-space

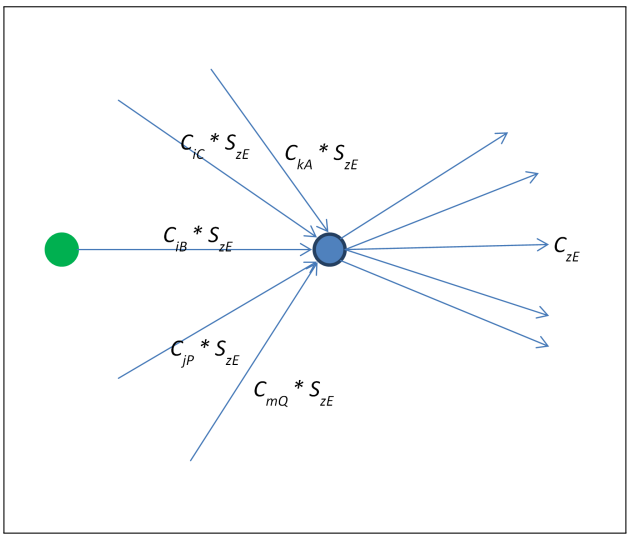


Figure 2.8: Graph(G) updated with SURF scores to give(G')

based algorithm has been discussed in the next section.

2.6 Selection of the scale-space based algorithm

An important consideration in selecting the algorithm to be used to determine the score S is its speed, since we have a large number of comparisons to make. Hence, SURF [9] is selected owing to its fast performance which in turn is due to the usage of integral images [16]. There is an inherent problem in using the default implementation of SURF for our images. Some of these problems have been discussed in subsection 1.2.5. The most important problem in this case, is the presence of images of small size. The size of images used are typically 200 pixels in height and the *log-* approximated kernel used varies from 9 to 99, spread over several octaves. To put simply, an octave is a division of the entire scale space *octaves*. The octaves and the corresponding sizes of the kernel, as proposed in the SURF paper, are as shown in the Figure 2.9. The response, which is the collection determinant of the Hessian Matrix [20] generated using various kernels from the first three octaves on one of our images, is as shown in the Figure 2.10. Since the responses vary over a wide range, it should be noted that while displayed using MATLAB, thresholding using the Otsu's method [15] has been applied convert the responses to binary image. This is just for the sake of ease of viewing.

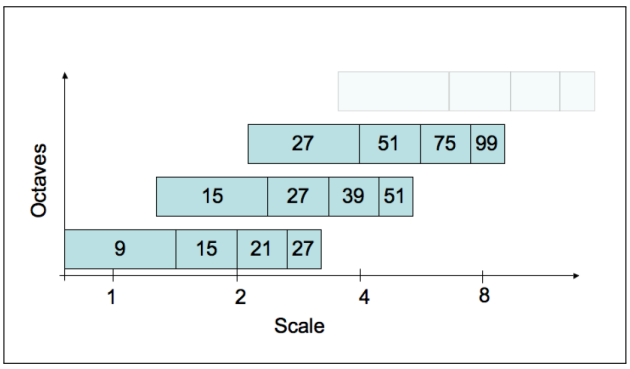
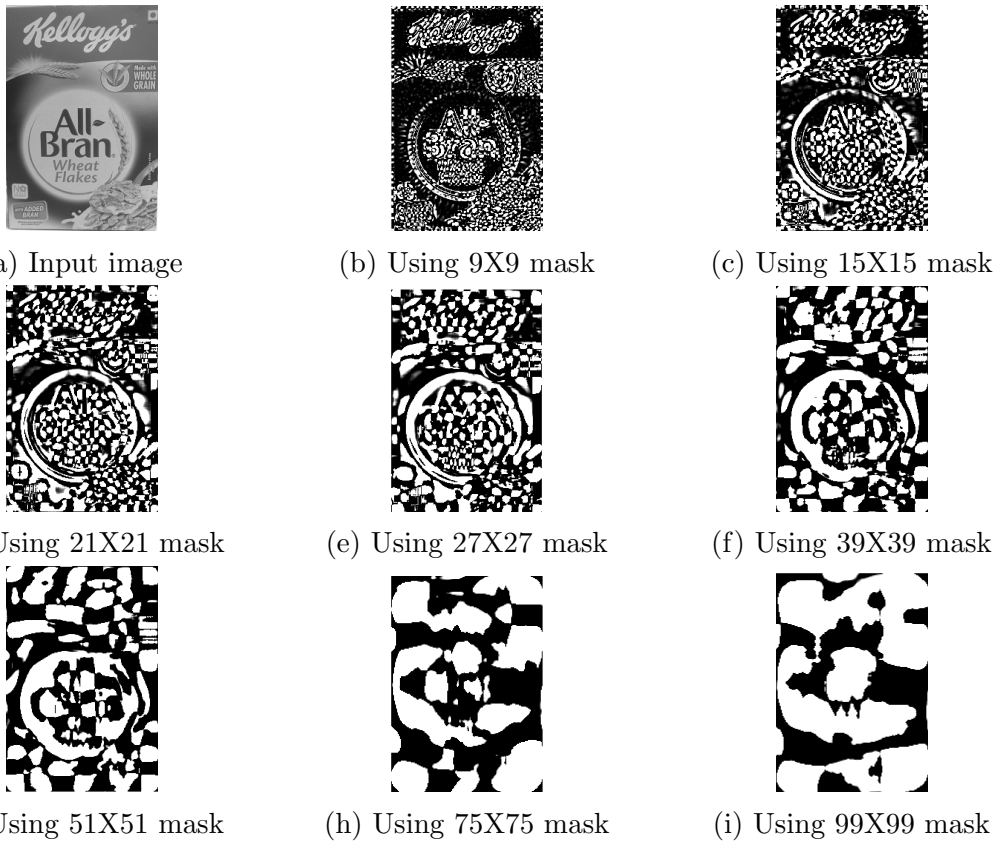


Figure 2.9: The kernel sizes used in various octaves, image taken from [9]



(a) Input image (b) Using 9X9 mask (c) Using 15X15 mask

(d) Using 21X21 mask (e) Using 27X27 mask (f) Using 39X39 mask

(g) Using 51X51 mask (h) Using 75X75 mask (i) Using 99X99 mask

Figure 2.10: Filter responses on a sample image for first three octaves

The responses shown in Figure 2.10 are produced by using a zero padding to keep the size of the response same as the size of the image. Padding with zeros is not a concern if the kernel size is very small as compared to the image size. But in our scenario, the size of the kernel is comparable to the size of the image. For example, the image used for the illustration here is 185 pixels in height. So, using a kernel of size 9X9 implies that the response obtained in Figure 2.10b has erroneous values near the boundary. The erroneous response starts at locations which are 9 pixels from the boundary and the error reaches to a maximum near the boundary. In order to remove this error and get rid of false keypoints, we discard the responses near the boundary in accordance with the size of the kernel used. In other words, we can say that we consider only those responses which are obtained without zero padding. In this case, the responses from the various kernels is as shown in Figure 2.12. This figure

explains why we do not need to consider working with octaves higher than 2. The region of interest completely disappeared as we moved to a kernel size of 99X99 in the third octave. Taking this into consideration, we propose a new set of kernels in order to sub sample the scale space in accordance with the size of our images. We propose three new sub octaves in order to capture the maximum possible information from small images which we have. In order to do so, we vary the the kernel size uniformly for the two octaves, as shown in the Figure 2.11.

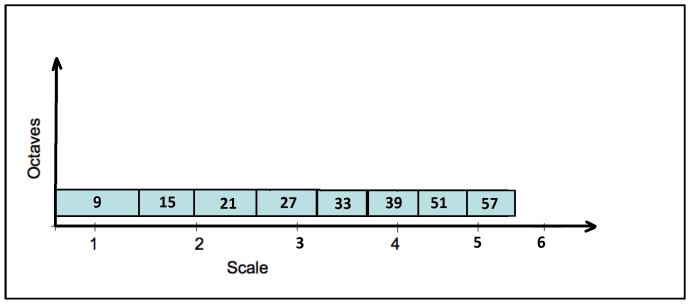
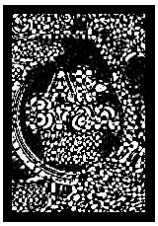


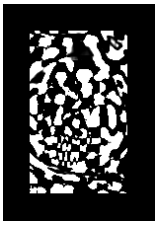
Figure 2.11: Sub sampling the scale space



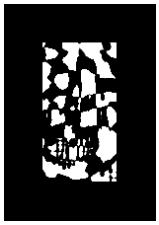
(a) Using 9X9 mask

(b) Using 15X15 mask

(c) Using 21X21 mask

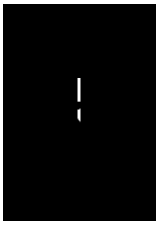


(d) Using 27X27 mask



(e) Using 39X39 mask

(f) Using 51X51 mask



(g) Using 75X75 mask

(h) Using 99X99 mask

Figure 2.12: Filter responses on a sample image for first three octaves, assuming there was no padding

In the Section 2.3, it was discussed that the correlation was found to be performing well on our dataset. So, we deviate from SURF and move to a correlation based solution involving

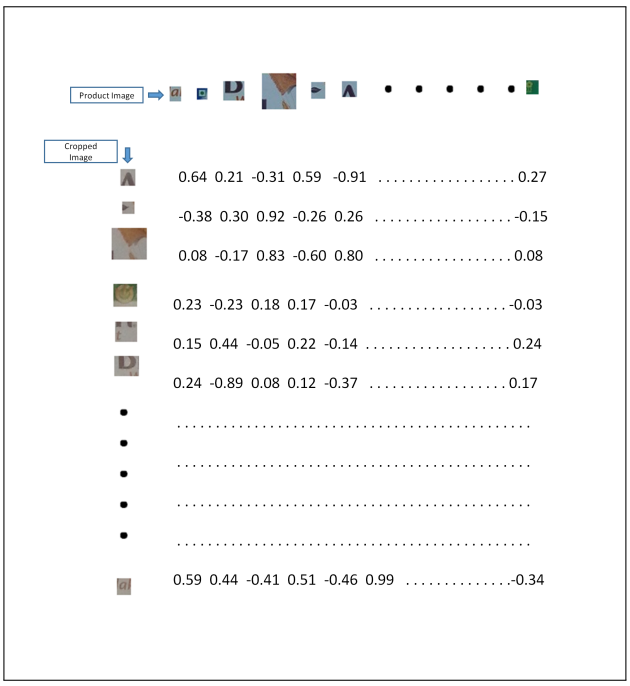


Figure 2.13: Using correlation on a bag of words to get a score, instead of SURF [9] vectors

the concept of Bag of words model [12]. We find all the keypoints from the responses obtained using the kernels as specified by SURF and find the keypoints and its scale. A keypoint is a blob in an image, where the value of the response is a local maxima. The scale of a keypoint is a measure of how big is the blob. A detailed yet simplified explanation of concept of scale-space can be found in the VLfeat documentation [1]. Once we know the location of the keypoints and their scales, we can crop those small portions of the image out. Each keypoint gives us a small square portion of size sXs. The value of s is directly proportional to the scale of the keypoint. Once we have all the small portions from both the query and the target images. Let us assume we had k_1 keypoints in the query image and k_2 keypoints in the target image, then if we find the correlation coefficient between each extracted portion of the target and the query image, we end up with a matrix of size $k_1 X k_2$, as shown in the Figure 2.13. correlation coefficient is calculated for each the three layers to each of the three layers in the L*a*b space and then an average of the three values is recorded in a similar fashion as we discussed in Section 2.3. It should be noted that the image shown in the Figure 2.13 shows the small portions in RGB color space, but after testing on both the RGB and L*a*b color spaces, the performance of the L*a*b based bag of words model outweighed the performance using the RGB color space. Hence the L*a*b space was preferred. Once we have a matrix of scores as shown in the Figure 2.13, we put a high threshold on these values and count the number of values in the matrix exceeding the threshold. We call this count the match score. This score is used as the score S_{ze} discussed in Section 2.5 to update our Graph G, thus yielding a new Graph G', which is used to produce the final output. It should be noted by using a colored bag of words model in the L*a*b* color space, we are easily able to capture the color information too, thus boosting thus boosting the performance of the algorithm.

Figure 2.14: A sample final result generated using the updated graph

Once we are able to identify the products in each shelf, these are appended to produce the final reconstructed image of the input. It should be noted that each shelf can be processed in parallel since the processing of one shelf is independent of the other. The output produced by selecting the shortest path from G' is found to be closer to the actual arrangement in the shelf. The improved output on the same shelf is shown in Figure 2.14. The result is still not 100% accurate, thus leaving some room for improvements in our approach.

Chapter 3

Experiments and results

We begin by discussing about out dataset. Then, we move on to summarise our results and accuracies. Some samples of our results, a few good results and some failure cases have been included.

3.1 The dataset

The problem at hand is to answer the question-"Which product is placed at which location in the rack?"

A benchmark dataset was constructed by choosing 20 images from a pool of around 150 images for the purpose of this dissertation. A typical image of the shelves, henceforth to be referred as *rackImage*, is shown in the Figure 3.1.

Other information provided along with the *rackImage* are:

1) Height of each shelf in the *rackImage*(in c.m. or inches)

2) Width of the rack(in c.m. or inches)

Figure 3.1: Sample image of the store

Apart from the *rackImage*, we are also given a set of high resolution images of the

products, henceforth to be referred as *productImages*. A few examples of the *productImages* are shown in the Figure 3.2.

Other information provided along with each of the *productImage* are:

1) Height of each product(in c.m. or inches)

2) Width of each product(in c.m. or inches)

It should be noted that the above information can be used to get a prior estimate of the number of pixels per unit length(c.m or inch). We refer to this value this as the global scaling.

Figure 3.2: Sample product images

Some examples of our reconstructed images are shown in Figure 3.3 and Figure 3.4.

Figure 3.3: Some examples of the reconstructed outputs- *rackImage* 9 and *rackImage* 17

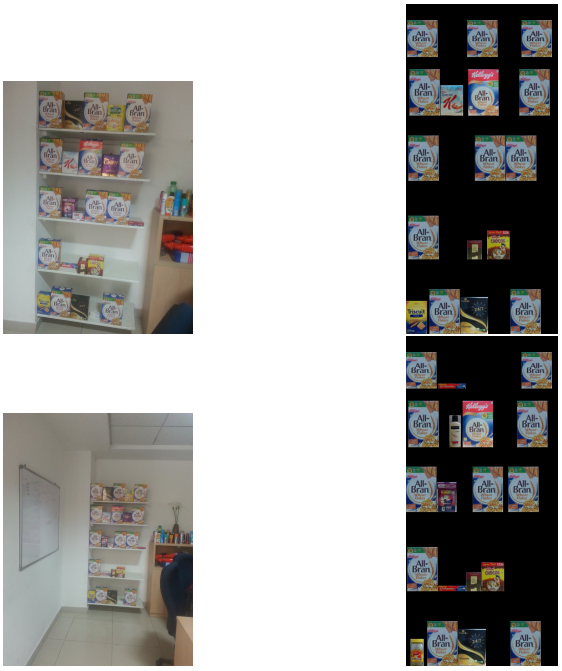


Figure 3.4: Some examples of the reconstructed outputs- *rackImage* 18 and *rackImage* 20

3.2 Accuracies obtained using various methods for the 20 *rackImages*

The accuracy for any particular image is calculated as the ratio of the number of products identified correctly to the total number of products present in the *rackImage*. The average accuracy calculated over the 20 images is reported as a measure of the performance of the entire algorithm. The Table 3.1 presents the number of products identified correctly for each of our 20 images. This is done for each of our approaches. It should be noted that in all the approaches whose accuracies are presented, the correlation based shortlisting of products, as discussed in 2.3. The average accuracies for each method is mentioned in the table. The graph based approach with the bag-of-words based model as a feedback mechanism (Method E) is found to give comparable results to the one using the default implementation of SURF as a feedback mechanism (Method F). The Method E is able to identify smaller products better as is shown in the Image Number-18 and Image Number-20 Figure3.4.

$$\text{Accuracy} = \frac{\text{Number of products identified correct}}{\text{Total number of total products present in the image}}$$

Image Number	Total Products	Method A	Method B	Method C	Method D	Method E	Method F
1	25	2	10	10	13	17	18
2	25	5	12	13	12	13	16
3	25	5	12	13	13	16	18
4	25	4	10	11	13	17	18
5	25	2	11	10	13	15	15
6	25	4	12	13	14	17	19
7	25	4	13	14	13	16	18
8	25	4	12	14	13	15	17
9	25	3	13	10	12	17	17
10	25	1	10	13	12	16	17
11	25	4	11	11	11	17	16
12	25	4	15	14	14	17	16
13	25	3	13	13	13	16	16
14	25	4	12	12	12	13	16
15	25	4	16	14	16	17	15
16	25	5	9	13	12	18	15
17	25	3	11	11	11	13	17
18	25	4	13	10	14	18	17
19	25	4	10	15	12	18	17
20	25	3	11	13	10	17	15
	Mean Accuracy	**14.4**	**47.2**	**49.4**	**50.6**	**64.6**	**66.6**

Table 3.1: Accuracy over 20 *rackImages* for the various approaches proposed in the thesis

Method	**Explanation**
Method A	Shortlisting based on correlation and selecting the optimum path (No feedback mechanism)
Method A	Shortlisting based on correlation and selecting the optimum path (No feedback mechanism)
Method B	Shortlisting based on correlation and feedback mechanism using the color-co-occurrence histograms
Method C	Shortlisting based on correlation and feedback mechanism using the L4-L5-L6 color space trivariate histograms
Method D	Shortlisting based on correlation and feedback mechanism using the Synthetic data trained SVM
Method E	Shortlisting based on correlation and feedback mechanism using the proposed bag of words model
Method F	Shortlisting based on correlation and feedback mechanism using the default SURF implementation

Table 3.2: Names of methods used in the Table 3.1

3.3 Comparison with competing method

The mean accuracy obtained using the methodology proposed in this thesis (Method E) has been compared to one of the most recent works in the field of grocery object detection [8]. George *et al.* [8] reported an mAA (mean average accuracy) of 64 %. Our approach produced a mean accuracy of 64.6 %. Our dataset, being composed of 20 *rackImages*, was small as compared to theirs. Hence, it cannot be claimed that the proposed approach outperformed the algorithm given by [8].

Chapter 4

Conclusion

In this thesis, a methodology was proposed to identify the products present in a rack in retail stores, in the absence of any sort of training data. Our methodology involved the application of graph theory and scale-space based algorithms to solve the problem. Through experimental evaluations, the effectiveness of our approach was shown on a set of 20 *rackImages*. The results obtained by our method was shown to be comparable to another state-of-the-art algorithm in this field.

Chapter 5

Future Direction

Right now the time required to solve one particular *rackImage* is of the order of 20 minutes, if run on a machine with 4 processors and a processing speed of 3.4 Ghz. since most of the images in our data set had 5 shelves, the timing can be reduced to a total of 5 minutes by processing each shelf on five different machines. If this method can be made to work in real time, the possibilities can be huge- including applications in assisting the blind consumers to find the products which they wish to buy.

Several other modifications to scale space based algorithms, primarily SURF [9], are possible, which might help in improving performance. The constraint put on the graph has several avenues of improvement. As of now, the Graph suffers from a drawback that it tends to favour smaller products over wider products. One possible reason is that the information about the width of the product is not being incorporated into the feedback system. Right now, the work is being carried out in this direction, but any significant improvement has not been achieved as of now.

Bibliography

[1] A Kuijper. Gaussian scale space. *University of Wisconsin Lecture notes* (2004), 11–22.

[2] Alaa E. Abdel-Hakim, A. Csift: A sift descriptor with color invariant characteristics. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2004), 1–6.

[3] article, W. The cie lab color space.

[4] Bashir, F., and Porikli, F. Performance evaluation of object detection and tracking systems. In *Proceedings 9th IEEE International Workshop on PETS* (2006), pp. 7–14.

[5] David G. Lowe. Distinctive image features from scale-invariant keypoints. *The International Journal of Computer Vision, 2004 32*, 9 (2004), 1–28.

[6] David L. Davies, D. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI*, 2 (1979), 224–227.

[7] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence 32*, 9 (2010), 1627–1645.

[8] George, M., and Floerkemeier, C. Recognizing products: A per-exemplar multi-label image classification approach. In *European Conference on Computer Vision* (2014), Springer, pp. 440–455.

[9] Herbert Bay, Andreas Ess, T. L. Surf: Speeded up robust features. *Elsevier 32*, 9 (2010), 1–14.

[10] Kleinberg, J., and Tardos, E. *Algorithm design*. Pearson Education India, 2006.

[11] Koen E.A. van de Sande, Theo Gevers, C. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*, 9 (2010), 1582–1596.

[12] Kristen Grauman, Bastian Leibe, C. Indexing and visual vocabularies, chapter 5. *University of Texas at Austin Lecture notes* (2010), 62–69.

[13] Marius Muja, D. Fast approximate neighbours with automatic algorithm configuration. *University of British Columbia Archives* (2009), 1–10.

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

[15] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS 9*, 1 (1979), 62–66.

[16] Paul Viola, M. Robust real-time face detection, section 2.1. *The International Journal of Computer Vision* (2004), 137–154.

[17] Peng Chang, J. Object recognition with color cooccurrence histograms. *IEEE Conference on Computer Vision and Pattern Recognition, Fort Collins, CO,* (1999), 1–7.

[18] Pinz, A. Object categorization. *Foundations and Trends® in Computer Graphics and Vision 1*, 4 (2005), 255–353.

[19] Rallapalli, S., Ganesan, A., Chintalapudi, K., Padmanabhan, V. N., and Qiu, L. Enabling physical analytics in retail stores using smart glasses. In *Proceedings of the 20th annual international conference on Mobile computing and networking* (2014), ACM, pp. 115–126.

[20] StackExchange. The approximation part. *peer discussion forum*.

[21] Thomas H. Cormen, Charles E. Leiserson, R. C. *Introduction to Algorithms*. The MIT Press, http://bayanbox.ir/view/4177858657730907268/introduction-to-algorithms-3rd-edition.pdf, 1993.

[22] Van De Sande, K., Gevers, T., and Snoek, C. Evaluating color descriptors for object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence 32*, 9 (2010), 1582–1596.

[23] VLfeat. Gaussian scale space fundamentals. *API- documentation*.

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