Recognition of Grocery Items in a Shopping Cart

Aniq Masood
Department of Electrical Engineering
Stanford University
Stanford, CA
amasood@stanford.edu

***Abstract*— The goal of this project is to create an algorithm that allows a grocery store shopper to identify and price the items in their shopping cart by simply taking an image of their shopping cart. The proposed algorithm successfully detects and identifies multiple grocery items using the Scale Invariant Feature Transform (SIFT) and image matching techniques.**

Keywords*— *SIFT; object recognition; image matching; homography

I. Introduction

Object detection is a ubiquitous topic found in image processing and computer vision circles. Numerous methods have been conceived to extract useful information from images: Color based detection analyzes RGB and HSV histograms on a pixel by pixel level [1]. Image segmentation uses supervised and unsupervised thresholding to classify pixels in an image as foreground or background [2]. Morphological image processing techniques probe images with small structuring elements to detect shapes and patterns [3].

However, employing a feature based image matching method was a fast and reliable technique for identifying objects with variations in color, orientation, size and shape. For the purpose and scope of this project, this method proved to be a viable approach. Features and descriptors were determined using the SIFT algorithm invented by Lowe [4]. Features from an image with items in a shopping cart are compared against features from a set of database images. The corresponding feature matches are then filtered by pixel location and objects are identified based on the number of feature matches between each database item and the query image. The algorithm was developed in MATLAB and utilized the VLFeat library [5] to compute SIFT features and descriptors.

This paper outlines the algorithm used to detect multiple grocery items in a shopping cart. Section 2 presents the overall image processing algorithm. Section 3 explains the methodology step by step. Section 4 presents the experimental results of testing the algorithm with up to 10 items in a shopping cart image. Finally, Section 5 presents possible improvements to the method and future work.

II. Algorithm Overview

Fig.1 shows the image processing pipeline for this project.

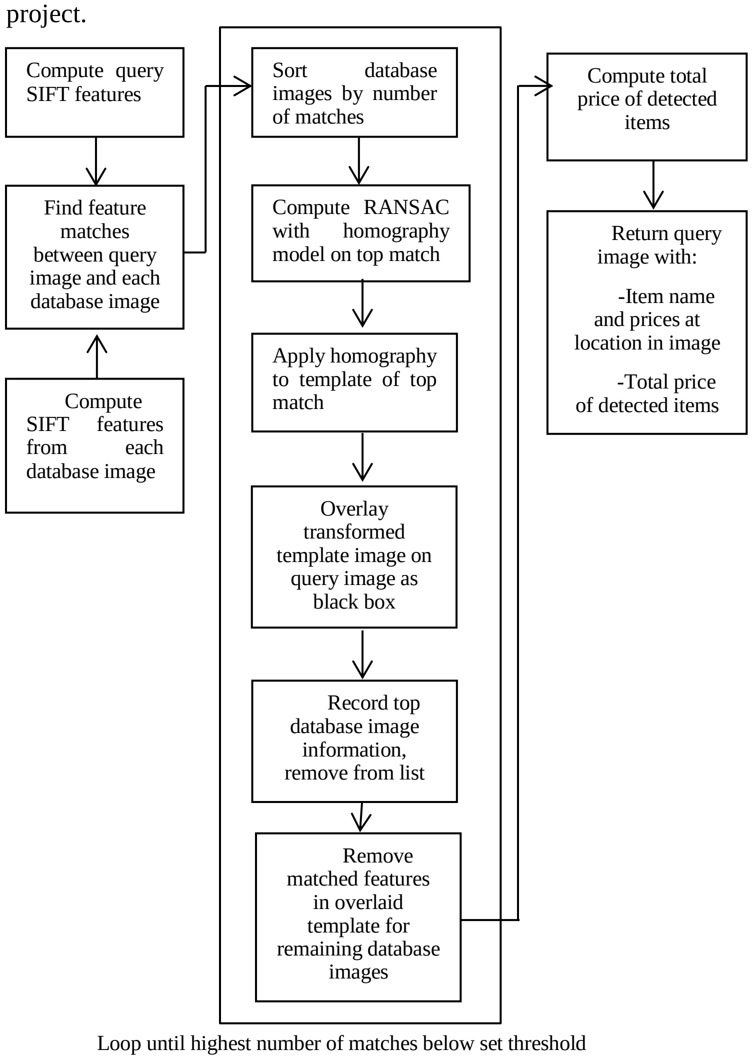

Compute query SIFT features
Find feature matches between query image and each database image
Compute SIFT features from each database image
Sort database images by number of matches
Compute RANSAC with homography model on top match
Apply homography to template of top match
Overlay transformed template image on query image as black box
Record top database image information, remove from list
Remove matched features in overlaid template for remaining database images
Compute total price of detected items
Return query image with:
-Item name and prices at location in image
-Total price of detected items
Loop until highest number of matches below set threshold

Fig. 1. Algorithm overview.

III. Methodology

A. Feature Extraction From Database Images

The database item images and shopping cart test images were taken from a Lucky's supermarket. The database consists of 58 common grocery store items such as dairy products, snacks, office supplies, and toiletries. The items were chosen on the basis of variety in shape, size, and feature uniqueness. For each database image, the SIFT features, descriptors, corner pixel coordinates, name, and price were recorded and extracted, as shown in Table 1 and Fig. 2.

TABLE I. INFORMATION PER DATABASE IMAGE

Item #	Metadata			
	Name	***Price***	***# Features***	***Corner Pixels***
28	Cheez-it	4.79	947	[38,74],[2758,74], [2758, 2088], [38,2088]

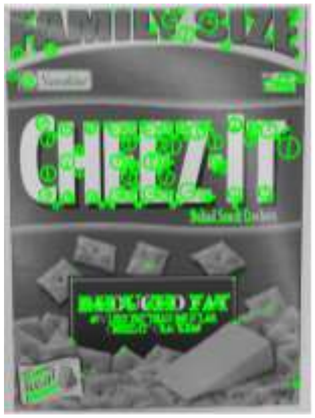


Fig. 2. Sample database image with SIFT features.

Using the *vl_sift* function, the recorded features were thresholded such that the maximum number of features per image did not exceed 3,000. This was done to reduce the computational time taken during the matching step. The peak and edge thresholds used were 10 and 7, respectively. Information extracted from database images was determined beforehand and imported into the MATLAB workspace when the algorithm was implemented.

B. Feature Extraction From Query Image

The SIFT features extracted from the query shopping cart images were thresholded using the same parameters as the database images. A sample image is shown in Fig. 3.

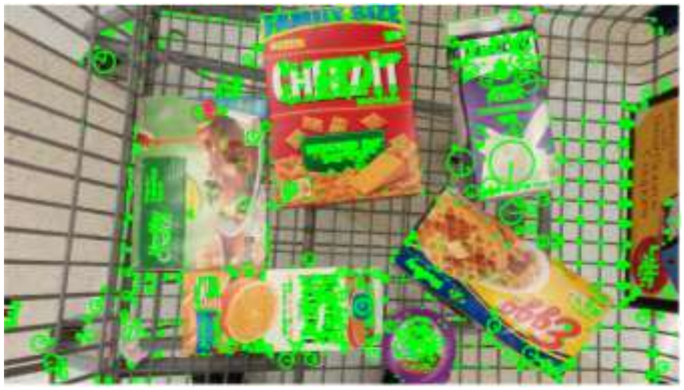


Fig. 3. Sample query image with SIFT features.

C. Finding Feature Matches Between Query and Database Images

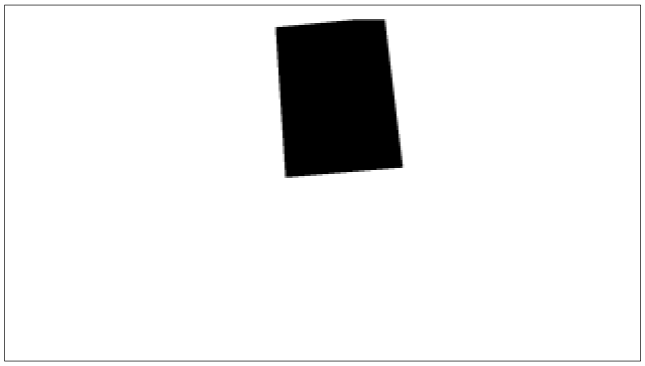
Now that we have the feature and descriptor information, we can now find the corresponding feature matches. The descriptors in each database image are compared to the descriptors in the query image using the VL Feat command *vl_ubcmatch* with a threshold of 1.5. This command matches the descriptors using a nearest-neighbor search, which is defined as the keypoint with the minimum Euclidean distance in descriptor space. The matches are then further filtered to remove features in the database image that are matched to a single feature in the query image.

D. Rank Database Images, Estimate Homography Model of Top Match

The database images are now ranked based on the number of feature matches each image has with the query image. To create an accurate geometric mapping of features from the top database image to the query image, an implementation of the RANSAC algorithm is utilized to generate the best possible homography to map the bounding box of the database image to the query image [6]. For 100 iterations, RANSAC uses a random subset of feature matches to determine a homography. The best homography is scored and chosen that best retains the inlier matches.

E. Apply Homography, Overlay Transformed Template Image On Query Image

Once the homography model is determined, it is applied to the four corners of the database image. As shown in Fig. 4, a binary mask is created that has the dimensions equal to that of the query image. The transformed coordinates are used to create a bounding box at the matched item's location in the query image. The centroid of this bounding box is computed and stored for later use. Pixels that fall inside the bounding box are set to zero and those that are outside are set to 1. The mask and query image are then multiplied to produce a new query image.



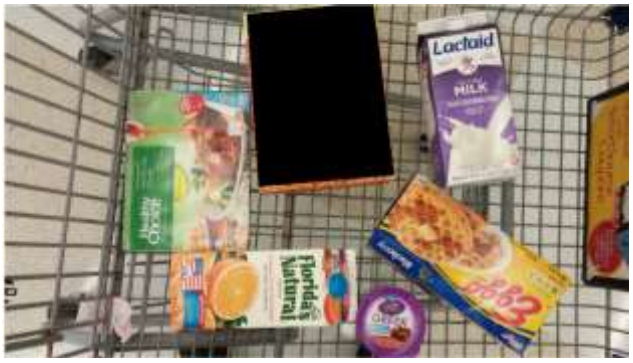


Fig. 4. Coordinate transformation and bounding box creation.

F. *Remove Matched Features in Bounding Box For Remaining Database Images*

Once this is done, the price, name, and bounding box's centroid coordinates are retrieved and the top match is removed from the list of possible matches. For every remaining database image, the matched features that are located in the query image's black box are removed. With the reduced number of feature matches, the database images are then ranked again and the process is repeated.

Sections D-F are repeated until the number of matches from the highest ranked database image falls below a set threshold empirically set to 86. Once the top matched database image has a total number of feature matches below this threshold, the loop stops and the results of the algorithm are returned.

G. *Total Price Computation and Displaying Results*

The prices of all detected items are added together and returned on the top right corner of the image. Each item's name and individual price are displayed at their centroid positions. Fig. 5 shows an example of the output of the algorithm.

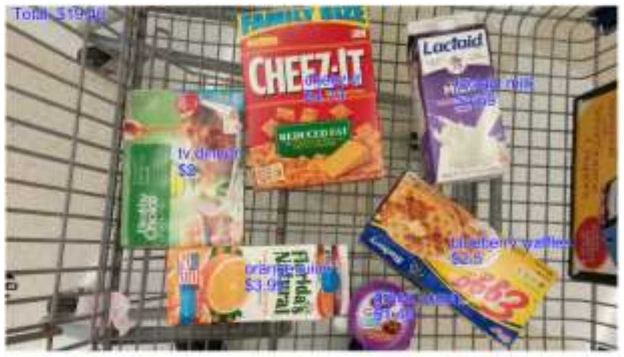


Fig. 5. Algorithm output.

IV. EXPERIMENTAL RESULTS

The proposed method was successful in identifying numerous combinations of items within the 58 item database. The algorithm was 100% accurate in locating the position of detected items correctly. It correctly detected up to 8 items in a given image, with 10 being the maximum number of objects on the test shopping carts.

A. *Varying Threshold Values as a Function of Item Count*

Table 2 shows the success rate of the algorithm as a function of number of items in the shopping cart.

TABLE II. ALGORITHM STATISTICS

Number of Items in Cart	***Success Rate (%)***	***Average Threshold value (# of matches)***	***Standard Deviation (# of matches)***
2	100%	310	115.9
3	100%	103.5	51.6
4	60%	88.4	45.9
5	60%	81.4	28.1
6	60%	89	39.3
8	66.6%	89.7	35.4
10	0%	50	0 (only 1 image was taken)

A success was defined when all of the detected items are correct, regardless of what threshold was chosen. For example, if a shopping cart has 4 items in it, the result would be deemed successful when the top 4 database matches correspond to the correct items in the image. A failure would occur if an incorrectly matched database item that is not in the image was in the top 4 matches. Furthermore, the threshold values used to determine the average threshold value corresponds to the lowest number of feature matches of a successfully detected grocery item. For example, with a shopping cart with 4 items, the item with the lowest number of feature matches was 88.4 on average.

One possible reason for the overall decreasing average threshold value with increasing number of items is that the shopping cart images with a lower number of items in them were taken at closer distances than those with larger number of items. Because of this, lower item carts have much larger item bounding boxes. This results in much more accurate results because the outlier matches from the database images that are not present in the image have a higher probability of being filtered out due to the increased bounding box pixel area. Conversely, shopping carts with a larger number of items in them are taken at further distances to capture the items fully, resulting in smaller item bounding boxes. The probability of filtering out incorrect features decreases because there are more background pixels in the image, resulting in less effective filtering using the bounding box approach.

B. *Sensitivty to Object Overlap*

The proposed method was robust against partial overlap of objects. The detection success of overlapped items depended on multiple factors. First, the percentage of features in overlapped areas. Fig. 6 is an example of a successful detection of overlapped items. The majority of the features from both overlapping objects are in the non overlapped areas.

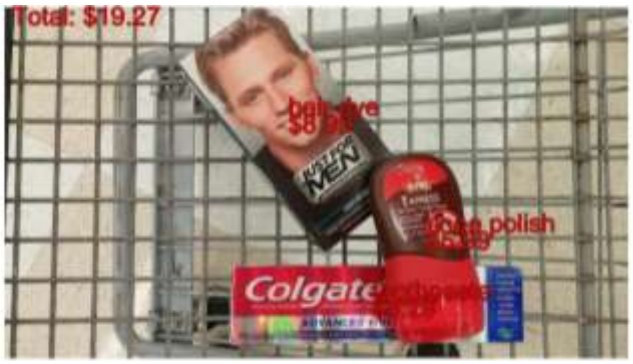


Fig. 6. Correct detection of overlapped items.

This allows the majority of features to be preserved by the bounding box filtering method. Second, the algorithm is sensitive to the order of detecting overlapped items and the percent overlap. For items with 100% overlap, such as smaller objects placed on top of larger ones, the smaller object was detected only if it had a larger number feature matches than the larger item. When this was the case, the smaller database item was removed from the possible matches before the larger item, allowing the features to be untouched by the feature filtering method. When the larger item is detected first, all of the inlier feature matches of the smaller item are thrown out because they are within the bounding box of the larger item.

C. *Sensitivity to Object Shape*

The experimental results determined that the method was more accurate for flat objects. Canned foods, for example, were more likely to be falsely undetected. This is attributed to the fact that all of the features from the template images were taken from the front labels of the items. Any angular deviations from the template image's front facing label would considerably hinder the algorithm's ability to do feature matching.

V. Improvements and Future Work

The varying loop threshold is a major setback in streamlining the algorithm for multiple applications. The sensitivity to number of shopping cart items limits the number of objects that can be detected at a time. One possible solution to remedy this would be to do RANSAC and eliminate the outlier feature matches for all images at the third step of the algorithm. It would reduce the overall number feature matches found and the inliers would not be susceptible to being falsely removed from the image.

Moreover, the method of generating keypoints favors grocery items with unique features. For example, fruits and vegetables are common grocery items that a grocery store shopper would benefit from pricing on the fly. However, these objects have large variations in color, shape, and size and would have varying keypoints from item to item. Integrating object detection techniques based on color into the algorithm would expand the type of detectable items rather than limiting the algorithm to items with unique labels and uniform shapes.

Finally, the computational time could be greatly reduced by using a vocabulary tree to match SIFT features and descriptors from query and database images. It would also allow for a much larger database to be used at virtually no cost of time.

VI. Acknowledgements

I would like to thank my project mentor Jason Chaves for his guidance on this project. His direction and advice was extremely helpful in refining the algorithm and successfully completing the project on time.

References

[1] Gonzalez, Rafael C., and Richard E. Woods. 1992. Digital image processing. Reading, Mass: Addison Wesley.

[2] Otsu, N., "A Threshold Selection Method from Gray-Level Histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 9, No. 1, 1979, pp. 62-66.

[3] Meyer, Fernand, and Serge Beucher. "Morphological segmentation." *Journal of visual communication and image representation* 1.1 (1990): 21-46.

[4] Lowe, D.G., "Object recognition from local scale-invariant features," Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on , vol.2, no., pp.1150,1157 vol.2, 1999

[5] http://www.vlfeat.org/index.html

[6] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” Communications of the ACM, vol. 24, no. 6, pp. 381-395, 1981.