Recognition and Volume Estimation of Food Intake using a Mobile Device

Manika Puri Zhiwei Zhu Qian Yu Ajay Divakaran Harpreet Sawhney
Sarnoff Corporation
201 Washington Rd,
Princeton, NJ, 08540
{mpuri, zzhu, qyu, adivakaran, hsawhney}@sarnoff.com

Abstract

We present a system that improves accuracy of food intake assessment using computer vision techniques. Traditional dietetic method suffers from the drawback of either inaccurate assessment or complex lab measurement. Our solution is to use a mobile phone to capture images of foods, recognize food types, estimate their respective volumes and finally return quantitative nutrition information. Automated and accurate food recognition presents the following challenges. First, there exist a large variety of food types that people consume in everyday life. Second, a single category of food may contain large variations due to different ways of preparation. Also, diverse lighting conditions may lead to varying visual appearance of foods. All of these pose a challenge to the state of the art recognition approaches. Moreover, the low quality images captured using cellphones make the task of 3D reconstruction difficult. In this paper, we combine several vision techniques (visual recognition and 3D reconstruction) to achieve quantitative food intake estimation. Evaluation of both recognition and reconstruction is provided in the experimental results.

1. Introduction

Studies have shown that healthy diet can significantly reduce the risk of diseases. This motivates a need to monitor and assess dietary intakes of individuals in a systematic way. It is known that individuals do a poor job of reporting their true dietary intake. Even dietitians need to perform complex lab measurements for accurate assessment. To overcome these problems, the Food Intake Visual and Voice Recognizer (FIVR) system has been designed to measure nutritional content of a user's meal. Since cellphones have become ubiquitous devices and mostly come equipped with cameras, the FIVR system works with a calibrated cellphone as an capture device.

Given a set of three images of a user's plate of food, our system first attempts to identify each food item on the plate and then reconstructs them in 3D to measure their respective volumes. The task of food identification is particularly difficult in our case because of the existence of a large number of foods. Purely relying on visual cues is insufficient to produce acceptable classification accuracy on such a large database using state-of-the-art techniques [11]. For example, it is hard to tell apart between shredded carrots and cheese from the images alone, unless presented with some extra information. Therefore, the FIVR system asks the user to list food items through speech, which is used to assist in the recognition task.

In order to recognize food with high accuracy and repeatability on a large database, we first train classifiers offline between all pairs of foods. Each pairwise classifier uses a combination of color and texture features at different scales learnt using Adaboost [12]. A multi-class classifier is assembled on the fly by selecting the pairwise classifiers according to the user's speech input. This classifier is applied on the input image to generate a segmentation map, which is used to calculate volumes for each food label. The volume estimation module first calculates the camera pose between multiple frames using a preemptive RANSAC [2] based method. After rectifying the images using the camera pose, the dense stereo reconstruction is computed. The final food volumes are estimated by first triangulating the food surface and then projected on the reference plane.

Figure 1 shows the architecture of the entire FIVR system. The user first uses a cellphone to provide three images of the plate of food along with speech data listing out each of the food items on it. This data is then sent to a remote server through the cellphone using wireless communication. A remote processing site is included in the framework so that the system can run on even cellphones that cannot support on-board sophisticated image analysis. At the arrival of data at the server, FIVR processing is initiated which, based on the food labels provided, first performs food identification to segment each item on the plate. After this food classification process, stereo-based techniques are employed to reconstruct each food item in 3D and determine its corresponding volume. Once the volumes have been computed, they are stored on the server in a database as well as communicated back to the user to appear as a text message on his cellphone screen.

978-1-4244-5498-3/09/$25.00 ©2009 IEEE

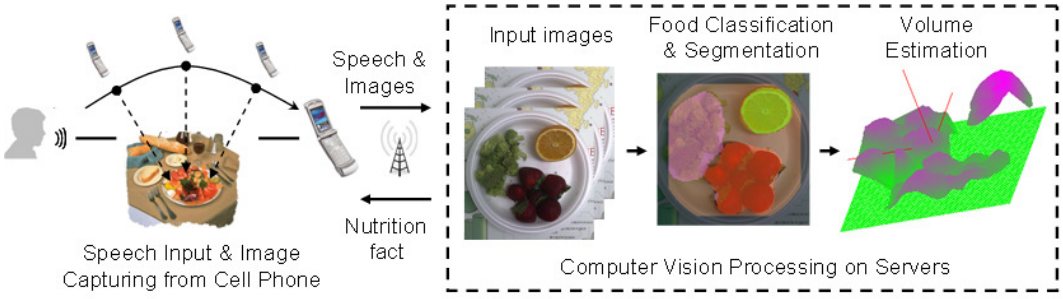


Figure 1. Data flow of the entire system

In the rest of this paper, we first discuss some related work in Section 2. The pairwise framework and feature selection in the recognition module are presented in Section 3. We present the details in 3D volume estimation algorithm in Section 4. The experimental results are shown in Section 5 followed by summary and future work.

2. Related Work

The need for monitoring nutritional consumption of individuals has been existing for a long time. However, it is still difficult to measure this information in an easy yet quantitative manner. Several applications and softwares, such as CalorieKing[1], CalorieCounter[2] *etc*. are of limited use since they perform a simple calculation based on portion size which cannot be accurately estimated by users. Veggie Vision [9] was an exploratory research project at IBM in the previous decade. The goal of Veggie Vision is to automatically recognize fruits and veggies in supermarket environments for an easy checkout. There is little published technical details about how this was achieved.

To the authors' best knowledge, our system is the first quantitative dietary assessment application that extensively makes use of recognition and 3D reconstruction for estimating volumes of a wide variety of foods. Another approach in [13] tackles the same problem by first carrying out an intensity-base segmentation and then classifying each segments using color and texture feature. However, their system does not carry out volume estimation.

The performance of state-of-the-art object recognition methods [11] still cannot satisfy the requirement of a practical application on a large number of classes. To reduce the candidate classes, in [10] the authors have introduced a fast rejection step based on image semantics. This significantly speeds up the classification while improving the accuracy. In our case, we make use of user input to reduce the number of candidate classes. This information helps us to achieve an acceptable recognition accuracy over a large database of food types.

Recent success in recognition is largely due to the use of powerful image features and their combinations. Concatenated feature vectors are commonly used as input for classifiers, but this is feasible only when the features are homogeneous, *e.g*. concatenation of two histograms HOG and IMH in [14]. Linear combination of multiple non-linear kernels, each of which is based on one feature type, is a more general way to integrate heterogeneous features, *e.g*. in [7]. However, both the vector concatenation and kernel combination based methods require computation of all the features. Feature selection techniques have used to choose the best combination on a fixed number of training samples in [15]. We propose to apply feature selection in a bootstrap procedure for our application since the number of samples is large.

3. Food Recognition

Automated and accurate food recognition is challenging because there are a large amount of food types that people consume in everyday life. A single category of food may have large variations. Moreover, diverse lighting conditions and a variety of ways of capturing using a cellphone can cause varying visual appearance of foods. Some typical foods are shown in Figure 2. In this section, we formulate the task of food recognition as a multi-class classification problem. To generate the segmentation map, we apply the multi-class classifier densely (*i.e*. every patch) to an input image. We adopt an Adaboost-based feature selection approach to combine color and texture information to achieve an acceptable recognition rate over a large number of food types.

3.1. Pairwise Classification Framework

There is a feature of our application, which helps to simplify the multi-class recognition problem: our application allows the user to provide a candidate set of his meal us-

Figure 2. Typical foods

[1] http://www.calorieking.com/
[2] http://www.mobigloo.com/software/caloriecounter/

ing speech. In order to make full use of this additional cue, we designed a classification framework, where a set of one-versus-one classifiers are trained offline between each pair of foods. Based on these offline trained pairwise classifiers, a dynamically assembled classifier is created according to the candidate set on the fly during the testing phase. This framework improves the accuracy of multi-class classification as well as the time performance of the system.

Suppose there exist N classes of food $\{f_i : i = 1, \cdots, N\}$, then all the pairwise classifiers can be represented as $\mathcal{C} = \{C_{ij} : i, j \in [1, N], i < j\}$. The total number of classifiers, *i.e.* $|\mathcal{C}|$, is $N \times (N-1)/2$. For a set of K candidates, $K \times (K-1)/2$ pairwise classifiers are selected to assemble a K-class classifier. The dominant label assigned by the selected pairwise classifiers is the output of the K-class classification. If there is no unanimity among the K pairwise classifiers corresponding to a food type, then the final output is set as unknown. Figure 3 shows an illustration of the pairwise framework with a set of 10 classes. The upper triangular matrix contains 45 offline trained classifiers. Suppose 5 classes (1,4,6,8 and 10) are chosen as candidates by the user, 10 pairwise classifiers are assembled to form a 5-class classifier. If 4 out of 10 classifiers report the same label, this label is used as the final report, otherwise an unknown label is reported by the 5-class classifier.

The advantages of this framework are two-fold. First, it reduces the computation cost during the testing phase. Second, compared with one-versus-all classifiers, this framework avoids the imbalance in training samples (*i.e.* few positive samples versus a large amount of negative samples). Another strength of this framework is its extendibility. Since there are a large amount of food types, we aim to provide users with the ability to incrementally update existing classes with new instances and add new food types without re-training classifiers from scratch. This pairwise framework is easy to adapt to new classes and to new instances. Suppose there exist N pre-trained classes, updating a class can be done easily by re-training $(N-1)$ classifiers in the upper triangular matrix; adding a new class, named as f_{N+1}, is equivalent to adding a new column (N) of classifiers $\{C_{i,N+1} : i = 1, \cdots, N\}$.

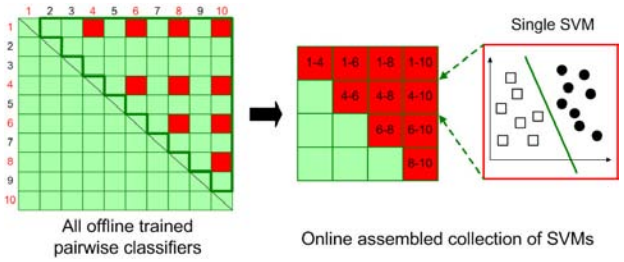


Figure 3. Illustration of the pairwise framework.

3.2. Bootstrap with Feature Selection

3.2.1 Color and Texture Features

For our food recognition problem, we focus on color and texture features, as they are generic for all kinds of food. Figure 2 shows some typical food types in our database. We can see that shape is not a persistent feature for these food types. To deal with varying lighting conditions, we place a color pattern in the image for photometric calibration. Fourteen colors (12 from the color pattern and 2 from the checker-board) have been used to solve the 3×3 color transformation matrix using a least squares solution. As texture features may vary with changes in scale, normalization of scale is necessary. For this purpose, a scaling factor is determined to map the checker-board to a predetermined size (75×75 pixel). The plate is located by using a contour based circle detection method proposed in [16]. We regard plate as one label in the classification and we annotate plate regions as well in the training set. Figure 4 shows an example of image before and after color/scale normalization. The detected color pattern, checker-board and the plate have been overlaid on the first image.

(a) before normalization (b) after normalization

Figure 4. Color and scale normalization.

To compute the label map, we apply the classifiers densely (*i.e.* every patch) on the color and scale normalized images. To train such a classifier, we manually annotate the training set to obtain the segmentation, in the form of label masks, of the food. We use texton histograms as our features, which essentially translate to a bag-of-word approach. Texton histograms have been widely used for texture recognition and encode the distribution of occurrence of textons (words). There are many approaches that have been proposed to create textons such as, spatial-frequency based textons [17], MRF textons [8] and gradient orientation based textons [4]. A detailed survey and comparison of local image descriptors can be found in [6].

It is important to choose the right texton as it directly determines the discriminative power of texton histograms. The current features used in our system include color (RGB and LAB) neighborhood features [8] and Maximum Re-

sponse (MR) features [17]. The color neighborhood feature is a vector that concatenates the color pixels within an $L \times L$ patch. Note that for the case $L = 1$ this feature is close to a color histogram. An MR feature is computed using a set of edge, bar and block filters along 6 orientations and 3 scales. Each feature consists of eight dimensions by taking maximum along each orientation [17]. Note that as the convolution window is large, we directly apply convolution on the image instead of patches, which is different from the way in [17]. We first compute filter responses and then form the feature vector according to a sampled patch. Both color neighborhood and MR features can be computed densely in an image and the computational cost is relatively low. Moreover, these two types of features contain complementary information: the first one contains color information but cannot carry edge information at a large scale, which is represented in MR features; on the other hand, the MR feature does not encode color information, which is useful to separate foods. We observed that by using only one type of feature at one scale we cannot achieve a satisfactory result over all pairwise classifiers. In the next subsection, we show how to use feature selection to create a strong classifier from a set of weak features.

3.2.2 Feature Selection in Bootstrap

A pair of foods may be more separable using some type of feature or at a particular scale than using other features or at other scales. In training a pairwise classifier, we could simply choose all possible types and scales of features and concatenate all of them into one feature vector. This, however, put too much burden on the classifier by confusing it with non-discriminative features. Moreover, this is not computationally efficient. Thus, we create a rich set of local feature options (color, texture, scale) and adopt a process of feature selection to automatically determine the best combination of heterogenous features. The types and scales of features used in current system are shown in Table 1.

Type	Scale
Color(RGB/LAB) Neighborhood [8]	$1, 3, 5, 7$
Maximum Responses [17]	$0.5, 1, 2$

Table 1. Features options

Our feature selection algorithm is based on Adaboost [12], which is an iterative approach for building strong classifiers out of a collection of "weak" classifiers. In our approach, each weak classifier corresponds to one type of texton histogram. We adopt a χ^2 kernel SVM to train the weak classifier using one feature in the feature pool. A comparison of different kernels in [18] shows that the χ^2 kernel outperforms the rest.

Let's denote the feature set $\{f_1, \cdots, f_n\}$ by $\mathcal{F}$. A strong classifier based on a subset of features by $F \subseteq \mathcal{F}$ is obtained by linear combination of selected weak SVM classifiers, $h : \mathcal{X} \to \mathbb{R}$,

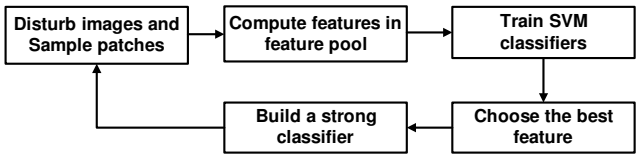


Figure 5. Bootstrap training procedure

$$h_F(x) = sign\left(\sum_{f_i \in F} \alpha_{f_i} h_{f_i}(\mathbf{x})\right) \tag{1}$$

where $\alpha_{f_i} = \frac{1}{2}\ln(\frac{1-\varepsilon_{f_i}}{\varepsilon_{f_i}})$ and ε_{f_i} is the weighed error rate of the weak classifier f_i. For a sample $\mathbf{x}$, denote its true class label by $y(= \pm 1)$. The classification margin of h on $\mathbf{x}$ is defined by $y \times h(\mathbf{x})$. The classification margin represents the discriminative power of the classifier. Larger margins imply better generalization power. Adaboost is an approach to iteratively select the feature in the feature pool which has the largest margin according to current distribution (weights) of samples.

$$h_{k+1} = \arg\max_{h \in \mathcal{F}} M(H_k + h) \tag{2}$$

where H_k is the strong classifier learned in the k^{th} round and $M(\cdot)$ is the expected margin on $\mathcal{X}$.

As each h is a SVM, this margin can be evaluated by N-fold validation (in our case, we use $N = 2$). Instead of comparing the absolute margin of each SVM, we adopt a normalized margin, as $M(h, x) = \frac{yh(\mathbf{x})}{\|h\|}$, where $\|h\|$ denotes the number of support vectors. This criterion actually measures the discriminative power per support vector. This criterion avoids choosing a large-margin weak classifier that is built with many support vectors and possibly overfits the training data. Also, this criterion tends to produce a smaller number of support vectors to ensure low complexity.

Another issue in our application is how to make full use of training data. Given annotated training images and a patch scale, we can extract a large number of patches by rotating and shifting the sampling windows.[3] Instead of using a fixed number of training samples or using all possible training patches, we adopt a bootstrap procedure, shown in Figure 5, to sample training data and select features simultaneously. Initially, we randomly sample a set of training data, compute all features in feature pool and train individual SVM classifiers. We use a 2-fold validation process to evaluate the expected normalized margin for each feature and choose the best one to update the strong classifier with weighted classification error. Given the current strong classifier, we apply it to densely sampled patches in the annotated images, and add wrongly classified patches (plus the

[3]Note that we do not pursue scale-invariant features in this application since scale information is also helpful to discriminate some types of food, *e.g.* blueberries and grapes.

ones close to the decision boundary) as new samples, then update the weights of all training samples. Note that we perturb the training images in the LAB color space before bootstrapping. The training is stopped if the number of wrongly classified patches in the training images falls below a strict threshold.

4. 3D Food Volume Estimation

In order to estimate the volume of food items on a user's plate, a set of three single 2D images are taken at different positions above the plate.

Figure 6 illustrates the proposed 3D food volume estimation flowchart. As shown in the figure, once the three food images are taken, first the relative camera poses (comprising of six degrees of freedom with 3 rotations and 3 translations) among them are estimated. Then, the first two images are selected to form a stereo pair and 3D reconstruction is carried out on them to generate 3D point clouds of the food. Finally, from the reconstructed 3D point cloud, both the 3D scale and table plane are estimated to compute the 3D volume of each food item. The following sections explain details of the proposed algorithm.

4.1. Camera Pose Estimation

The first step of the algorithm involves extraction of multiple feature points in each image frame and finding matches between these frames. In the current implementation, Harris corners [3] have been used, however any other feature, which describes an image point in a distinctive manner, can be used in this step. Each feature correspondence establishes a feature track, which lasts as long as it is matched across the frames. These feature tracks are later sent into the pose estimation step which is carried out using the proposed preemptive RANSAC-based method [2], as explained in detail below.

The preemptive RANSAC algorithm starts with randomly selecting different sets of 5-point correspondences over three frames such that N number of pose hypotheses (by default N=500) are generated using the 5-point algorithm. Here, each pose hypothesis consists of the pose of the second and third view with respect to the first view. Then, starting with all the hypotheses, each one is evaluated on chunks of M data points based on trifocal Sampson error (by default M=100), every time dropping out half of the least scoring hypotheses. Thus, initially we start with 500 hypotheses, all of which are evaluated on a subset of 100-point correspondences. Then they are sorted according to their scores on this data set and the bottom half is removed. In the next step, another set of 100 data points is selected on which the remaining 250 hypotheses are evaluated and the least scoring half are pruned. This process continues until a single best-scoring pose hypothesis remains.

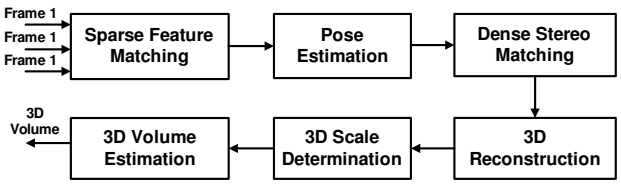


Figure 6. The flowchart of the 3D volume estimation from 3 single images.

In the next step, the best pose at the end of the preemptive RANSAC routine is passed to a pose refinement step where iterative minimization of a robust cost function (derived from Cauchy distribution) of the re-projection errors is performed through Levenberg-Marquardt method [5].

Using the above proposed algorithm, camera poses are estimated over three views such that poses for the second and third view are with respect to the camera coordinate frame in the first view. In order to stitch these poses, they are then placed in the coordinate system of the first camera position corresponding to the first frame in the image sequence. At this point, the scale factor for the new pose-set (poses corresponding to the second and third views in the current triple) is also estimated with another RANSAC scheme [2].

4.2. Dense Stereo Matching and 3D Reconstruction

Once the relative camera poses between the image frames are estimated, the first two frames are selected to form a stereo pair.

During the dense stereo reconstruction, for each pixel in the left image, its corresponding pixel in the right image is searched using a hierarchal pyramid matching scheme. Once the left-right correspondence is found, using the intrinsic parameters of the pre-calibrated camera, this match is projected in 3D using triangulation. At this stage, any bad matches are filtered out by validating them against the epipolar constraint. To gain speed, the reconstruction process is carried out for all non-zero pixels in the segmentation map provided by the food classification stage. Figure 7 shows the corresponding matches between left and right frames by a set of blue lines.

4.3. 3D Scale Determination

After the pose estimation step, there is still a scale ambiguity in the final pose of the frames. In order to recover the global scale factor, an object with known dimensions is placed and captured along with the plate of food in the image. For simplicity, a checker-board is utilized in the present case. In order to compute the 3D scale, the algorithm first detects each corner of the checker-board in the image followed by its reconstruction to obtain corresponding 3D coordinates. The size of each checker-board square is determined in 3D from its respective corners. Let d_{Ref} be the real size of checker-board as measured by ground

truth and d_{Est} be its size as obtained by estimation in 3D. Then, the 3D scale (S) is computed using equation 3. In the present case, we use a 3×3 checker-board, as shown in figure 4, with d_{Ref} = 3.14 cms.

$$S = d_{Ref}/d_{Est} \tag{3}$$

Once the 3D scale is computed using the checker-board, an overall scale correction is made to all the camera poses over the set of frames and they are mapped on a common coordinate system.

(a) Cropped left image with a heap of gram.

(b) Left-Right image-based matches.

Figure 7. Dense stereo matching

4.4. 3D Volume Estimation

Following the stage of stereo reconstruction, we obtain a dense 3D point cloud for all points on the plate. Figure 8 displays an example of a 3D point cloud for the image shown in Figure 7. Since the volume of each food item needs to be measured with respect to a reference surface, estimation of the table plane is carried out as a pre-requisite step. By inspection of the image, we know that apart from pixels corresponding to food on the plate, most lie on the table plane. Hence, table estimation is performed by employing RANSAC to fit a 3D plane equation on feature points earlier used for camera pose estimation. To obtain better accuracy, points falling on the plate are removed for the purpose of plane fitting by using the boundaries obtained from the plate detection step. Once the table plane has been estimated, it is used to slice the entire point cloud into two portions such that only 3D points above the plane are considered for the purpose of volume estimation.

The entire volume estimation process consists of two steps. First, Delaunay triangulation is performed to fit the surface of food. Second, total volume of the food (V_{Total}) is calculated as a sum of individual volumes (V_i) for each Delaunay triangles obtained from the previous step. Equation 4 shows computation of total food volume where K is the total number of triangles.

(a) Top View

(b) Side View

Figure 8. 3D reconstructed point cloud of gram.

$$V_{Total} = \sum_{i=1}^{K} V_i \tag{4}$$

The goal of the proposed FIVR system is to report volumes of each individual food item on a user's plate. This is done by using the binary label map obtained after food recognition. The label map for each food item consists of non-zero pixels that have been identified as belonging to the food item of interest and zero otherwise. Using this map, we select a subset of the 3D point cloud that corresponds to reconstruction of a particular food label and then feed it into the volume estimation process. This step is repeated for all food items on the plate to compute their respective volumes.

5. Experiments

In order to standardize our analysis of various foods, we refer to the USDA Food and Nutrient Database for Dietary Studies (FNDDS) [1] that contains more than 7,000 foods along with the information such as, typical portion size and nutrient value. Up to now, we have collected 400 sets of images containing 150 commonly occurring food types in the FNDDS. This data has been used to train the classifier. An independently collected data set with 26 types of foods has been used to evaluate the recognition accuracy. We randomly sample N(in this case, $N = 500$) patches from each type of food and evaluate the accuracy of classifiers trained in different ways:

- Using single MR feature ($\sigma_{x_1} = 0.5$);
- Using single RGB neighborhood features (at 3×3 scale);
- Using combined features with fixed number of training samples per food label;
- Using feature selection in the proposed bootstrap framework.

For comparison, we train all pairwise classifiers ($13 \times 25 = 325$) and sort the classification accuracy. As each pairwise classifier $c_{i,j}$ is evaluated over $2N$ patches (N patches in label i and N patches in label j), the pairwise classification accuracy is the ratio of correct instances over $2N$. Figure 9 shows the comparison of sorted pairwise accuracy. By applying the feature selection in the bootstrap procedure, we achieve significant improvement over using a single feature and using a fixed number of training samples.

In order to evaluate the multi-class classifiers assembled online based on the user input, we randomly add K confusing labels to each ground truth label in the test set. Hence, the multi-class classifier has $K+1$ candidates. The accuracy of the multi-class classifier is shown in Figure 10. As we

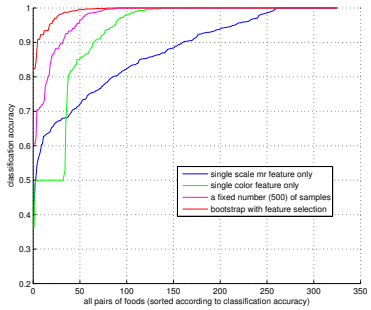


Figure 9. Comparison of pairwise classification accuracy

can see, the accuracy drops as the number of candidates increases. This is understandable since the larger the number of candidate, the more likely the confusion between them. However, the number of foods in a meal is rarely greater than 6, for which we can still achieve about 90% accuracy.

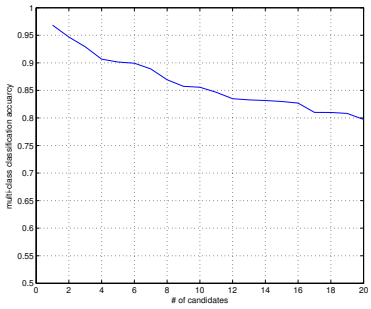


Figure 10. Effectiveness of using candidates

Qualitative results of classification and 3D volume estimation are shown in Table 2: the first column shows the images after scale and color normalization; the second column shows the classification results and the last column shows the reconstructed 3D surface obtained using Delaunay triangulation and the estimated table plane, which are used for computing the volume. Table 3 shows the quantitative evaluation of these sets. In the present system, volume is returned in milliliter units. This value can be converted to calories by indexing into the FNDDS.

To test the accuracy and repeatability of volume estimation under different capturing conditions, an object with a known ground truth volume is given as input to the system. For this evaluation, we captured 35 image sets of the object taken at different viewpoints and heights. Figure 11 shows a plot of error rate per image set. The average error in volume is 5.75 ($\pm$ 3.75) % over all the sets.

The present system is running on a Intel Xeon workstation with 3 GHz CPU and 4 GB of RAM. The total turnaround time is 52 seconds (19 seconds for classification and 33 seconds for dense stere reconstruction and volume estimation on a 1600x1200 pixel image). The current implementation has not been optimized and runs on a single core.

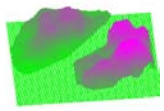
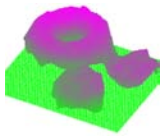
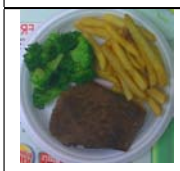
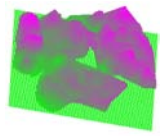
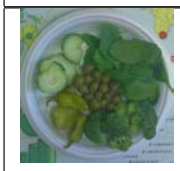
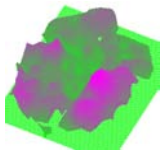
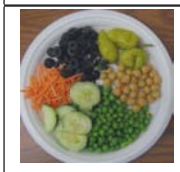
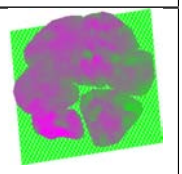
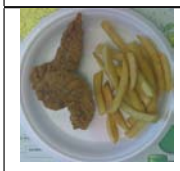
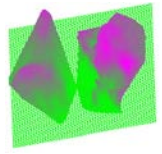
Image	Classification	3D Surface
		
Set1		
		
Set2		
		
Set3		
		
Set4		
		
Set5		
		
Set6		

Table 2. Qualitative classification and 3D surface reconstruction results

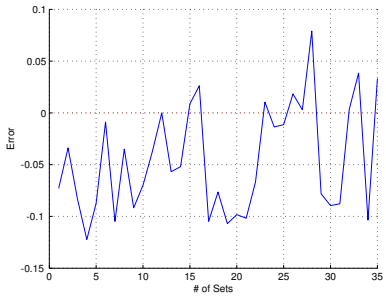


Figure 11. Accuracy and repeatability of volume estimation

Set #	Food	Ground truth (in ml)	Estimate (in ml)	Error (%)
1	Broccoli	150	143.5	4.3
	Carrots	120	112.3	6.4
2	Orange	195	189.4	2.9
	Bagel	300	310.5	3.5
3	Fries	200	194.8	2.6
	Steak	190	203.9	7.3
	Broccoli	180	186.3	3.5
4	Spinach	160	151.2	5.5
	Cucumber	100	98.2	1.5
	Olives	100	104.8	4.8
	Broccoli	120	114.2	4.8
	Peppers	80	82.7	3.4
5	Olives	100	98.4	1.6
	Carrots	90	82.7	8.1
	Peas	120	123.8	3.2
	Chickpeas	100	103.1	3.1
	Cucumber	140	144.2	3.0
	Peppers	90	84.1	6.6
6	Chicken	130	121.2	6.8
	Fries	150	133.6	10.9

Table 3. Quantitative classification and 3D volume results

Our extensive experiments indicate the textureless foods may cause erroneous 3D volume estimation. This can be improved by using better dense stereo matching algorithms. With the use of a color pattern, our system can deal with diverse lighting conditions to some extent. However, non-uniform lighting might still cause recognition to fail, and thus lead to wrong volume estimates. This problem can be alleviated by enlarging the training set to include more samples under different illumination settings.

6. Summary and Future work

We have presented a system that given a set of three images along with speech of a user's meal, performs object recognition and 3D reconstruction to estimate food volumes, which are reported back to the user. Our proposed pairwise classification framework, which is particularly designed for this application, makes full use of the user's speech input to improve both accuracy and computational efficiency of the system. Also, our framework is easy to extend to include new types of foods. Quantitative results show our food recognition and 3D volume estimation provide an automated and convenient solution to carry out studies on dietary intakes of individuals.

Currently there are 150 types of common food in our dataset and we are actively working on expanding it by including more food types. Besides a mobile device, in the future, we also plan to deploy this application on the Internet and build a user-centric database. Meanwhile, we intend to improve the robustness and usability of the entire system to make it work in diverse lighting conditions.

Acknowledgements

The research was supported by the grant U01HL091738 from the National Institutes of Health (NIH) through a subcontract with Viocare, Inc. Thanks to Rick Weiss of Viocare for many helpful discussions. Its contents are solely the responsibility of the authors and do not necessarily represent the official views of NIH.

References

[1] FNDDS. http://www.ars.usda.gov.

[2] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. In *CVPR*, 2004.

[3] C. Harris and M. Stephens. A combined corner and edge detector. In *the 4th Alvey Vision Conference*, 1988.

[4] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, pages 91–110, 2004.

[5] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[6] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *PAMI*, pages 1615–1630, 2005.

[7] M. Varma and D. Ray. Learning the discriminative power invariance trade-off. In *ICCV*, 2007.

[8] M. Varma and A. Zisserman. Texture classification: Are filter banks necessary? In *CVPR*, pages 691–698, 2003.

[9] Veggie Vision http://domino.watson.ibm.com/comm/wwwr_thinkresearch.nsf/pages/machine399.html.

[10] G. Csurka and F. Perronnin. A simple high performance approach to semantic segmentation. In *BMVC*, 2008.

[11] M. Everingham, L. V. Gool., C. K. I. Williams., J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results. http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html.

[12] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, pages 1651–1686, 1998.

[13] F. Zhu, A. Mariappan, C. J. Boushey, K. D. Lutes, D. S. Ebert, and E. J. Delp. Technology-assisted dietary assessment. In *SPIE*, 2008.

[14] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *ECCV*, 2006.

[15] N. Kumar, P. N. Belhumeur, and S. K. Nayar. Facetracer: A search engine for large collections of images with faces. In *ECCV*, 2008.

[16] W. Cai, Q. Yu, H. Wang, and J. Zheng. A fast contour-based approach to circle and ellipse detection. In *In: 5th IEEE World Congress on Intelligent Control and Automation (WCICA) 2004*, 2004.

[17] M. Varma and A. Zisserman. Classify images of materials: Achieving viewpoint and illumination independence. In *ECCV*, pages 255–271, 2002.

[18] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *IJCV*, pages 213–238, 2007.