Cafeteria Vision

Identification and Amount Measurement of Foods in a Plate

Ting-Fan Wu

Abstract

We present a prototype of automatic dish recognition system, intended to ease the checkout process in self-serve cafeterias, where the price is based on what food and how much a customer takes. When a plate with several dishes in it goes though the checkout counter, a snapshot will be taken and then sent to a computer for analysis. A variety of features, including color and texture are extracted. Color texton histogram is empirically proved to be the best effective feature. Finally, support vector machines are used to classify the dishes against the pre-trained dish image bank. A preliminary system recognizing 24 dishes is constructed and several major difficulties are also identified.

1 Introduction

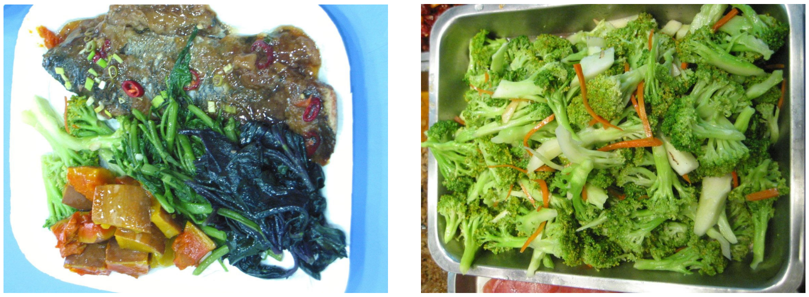
Cafeterias are extremely popular in Taiwan (see Appendix A for background.). They offer varieties of dishes sold at grocery store price at the speed like McDonald's. Eating in a cafeteria is a pipeline. First, a customer gets a plate at the entrance. Then he goes though a food-bar where dozens of dishes are served in bulk in big trays. he can pickup any dish in arbitrary amount as he wishes. Finally, the customer take the plate to checkout counter and a checker will calculate the total price base on unit prices and the amount of each dish in the plate.

As you can see, checkout is the most complicated part during the cafeteria food purchasing process. The clerk must memorize the floating[1] unit-prices (table lookup is time consuming.) and examining the amount of dish fairly while operating a cashier machine at same time. Due to the complication, only few employees are capable of being a checkout clerk and thus there is usually only one checkout counter in a cafeteria. As a result, the checkout counter is always the bottleneck. In addition, different clerks often have different personal scales of food amount measurement which usually results in customer complaints.

In this paper, we analyze the problem and design a prototype of the automatic checkout machine in order to speed up and standardize the checkout process. Furthermore, in addition to the total price, other bonus function can be added. For example, some people on diet might also want to know the total calories of food in her/his plate.

We organize the remaining of this paper as follows. First of all, in Section 2, some related researches are discussed. Next in Section 3 we describe the working flow for this application and investigate the nature of the problem. Then the two core components, texture

[1] Since the market price of fresh produces changes daily so the unit price will change accordingly.



(a) A **plate** with 5 dishes (b) A **tray** for holding food in a cafeteria

Figure 1: Sample images taken from a cafeteria in Taiwan.

recognition and segmentation, are described in Section 4 and 5. Several concept-proof and performance comparison experiment results are presented as well to support our decision of the design. The overall integrated results are presented in Section 6. Finally, Section 7 gives the conclusion of this work and provides future directions.

2 Related Work

Bolle et. al. [1] has developed a "Veggie Vision" system recognizing produces in a grocery store or a supermarket to ease the checkout process. In their scenario, there is only one produce in each image to recognize but the number of produces in their database is much larger (150) in comparison to ours (24).

Recently, such vision application like dish or produce recognition researches are getting interesting to the ubiquitous computing community. They use vision as a kind of sensors to build great human computer interface for "smart kitchen" and "smart dinning table". For example, [3] gives a scenario that dish recognition can apply to. In their original design, RFID and weight sensors are installed under a table to distinguish dishes on a table, so that they know who eats what. However, the correspondence between dish and RFID must be entered first manually, so that the computer knows which dish is placed on table by reading the RFID signal. Using dish recognition instead, we can setup a camera on top of a dining table to recognize the dishes directly and achieve similar purpose.

3 System Framework

First, we require that each bulk dish in a big trays (Figure 1(b)) to be *pre-scanned* before being available to customers, so the system can be trained against these images. Next in the checkout process, the customer plate images (Figure 1(a)) will be *captured* as well at the checkout counter for content analysis. Finally, the system looks up the price database, calculates the total price (and calories) and then sends the result to the cashier machine.

3.1 Hardware

An inexpensive CMOS cam[2] is setup on the top of a checkout counter to capture the image of dishes in a plate on the counter. Depending on different local environment, additional

[2] One can also use additional modified cam to capture IR-band image. It might provide valuable information for classifying some certain type of food types. Multi-spectral image has been proven benefit classification.

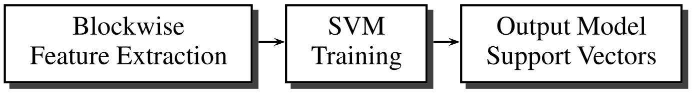


Figure 2: Procedure in training phase.

illumination might be required. Additionally, an off-the-shelf computer is used to run the image recognition algorithm and output the result. To speed up the image processing, a DSP card can be used as well.

3.2 The Nature of the Problem

- Image capturing
  Since the plates and trays will be place at the same counter and captured by the same camera. There will be no *scaling* and *shearing*, *minor illumination change*[3], and the plate/table surface are always the *known background.*
- Dish Type
  There are usually two types of pricing policies for two type of food respectively.
  - Countable food: $\text{price} = (\text{number of food}) * (\text{per-item price})$. Such kind of food, fish and chicken drumsticks for example, is usually rigid. Therefore, object recognition techniques might be useful.
  - Uncountable food: $\text{price} = (\text{amount of food}) * (\text{per unit price})$.
    In my case, we only have to distinguish “large” serving from “small’ serving. Such kind of food, vegetables for example, usually consists of very small deformable pieces. Therefore, it is not suitable for object recognition.

  For simplicity, we treat countable dishes as uncountable so that we can focus on one algorithm. But we keep in mind that we can apply object recognition techniques such as `SIFT` to further improve the performance of countable dishes.
- Unified Approach
  In our design, we treat all the dishes as well as the “counter table surface” and “paper plate” as textures. Then, this problem is reduced to “texture recognition and segmentation”. In this way, we eliminated the overheads of segmenting the dishes from the background (i.e. table surface and plate).

4 Texture Recognition

In this section, our goal is to find the best set of features and then train our model against the training tray images with the selected set of features.

4.1 Feature Selection: Cross-Validation and Two-Halves Test

Before we started to discuss the features, a baseline training/testing system must be setup to evaluate the performance of the features for comparison and avoid overfitting.

Since we do not have enough **plate** images for getting the true testing accuracy, splitted **tray** images are used instead. For each **tray** image, one half is used for training while the other is used for testing. To create numerous subsamples from a single **tray** image, a sliding window is used to walk through (blockscan) the image and generate the feature vectors based on the windowed image. The window must be wide enough to contain most

[3] We are not allowed to place the tray to the counter during the data collection process, so the illumination of the tray image is greatly different from the plate image.

Accuracy(%)	RGB	Lab	YCbCr	graylevel
Cross Validation	88.23	69.60	**91.17**	66.67
Two-halves Test	63.72	53.43	**63.73**	45.10

Table 1: The performance of `color histogram` under various colorspaces.

Accuracy(%)	RGB	Lab	YCbCr	graylevel
Cross Validation	39.22	30.88	25.49	**40.19**
Two-halves Test	27.45	23.53	21.08	**32.84**

Table 2: The performance of `gradient histogram` under various colorspaces.

of ingredients of the dish. Take the dish "carrot with stirred egg" for example, if the window is too small to cover pieces of both carrot and egg, it is very likely that the dish will be misclassified to be "pure carrot" or "pure stirred egg". Besides, two adjacent windows are not overlapped with each other to keep the independence and reduce the redundancy between blocks. Such sliding window subsampling is applied to both the training and testing halves.

As soon as the feature vectors are generated, we use support vector machines [2] to learn the from the training data. Radial basis function kernel (RBF) is used and the best kernel parameters are determined by grid search for the best **cross-validation** accuracy. Then a model trained with best parameter against the whole training data is used for testing to estimate a close-to-ground-truth testing accuracy (named **Two-Halves Test**).

It is observed that the cross-validation accuracy is always higher than the testing accuracy. Because the "intra-half variance" between blocks in the same half of images are lower than the "inter-halves variance". By investigating the source image, we found that sometime half of the tray is well-illuminated by the spotlight like lamp while the other half is darker. Therefore, the testing accuracy gives better estimation of the power of features under different illuminations.

4.2 Color Space

The images captured by a CMOS cam is originally in RGB colorspace which is designed for human vision system. Many other colorspaces can be found in literature including `YCbCr`, `Lab` and graylevel. Since we have no analytical clue which is better to use. We evaluate the performance of color histograms (explained in next subsection) under different colorspaces to find out the best colorspace suitable for our application. The result is shown in Table 1. `YCbCr` beats all other colorspaces. So we decided to use `YCbCr` for all color-related features throughout our experiment.

4.3 Features

The feature representation for a dish should be rotational and translational invariant, and tolerant to illumination change. Since dish textures are extremely random in spatial distribution, histogram-like features are the best choice to exploit this property. We tried 3 different type of histograms:

- Concatenated Color Histograms: We first calculate the marginal histogram for each channel of the colorspace, and then concatenate these histograms together forming a feature vector. This features works the best in the `YCbCr` colorspace (see Table 1).

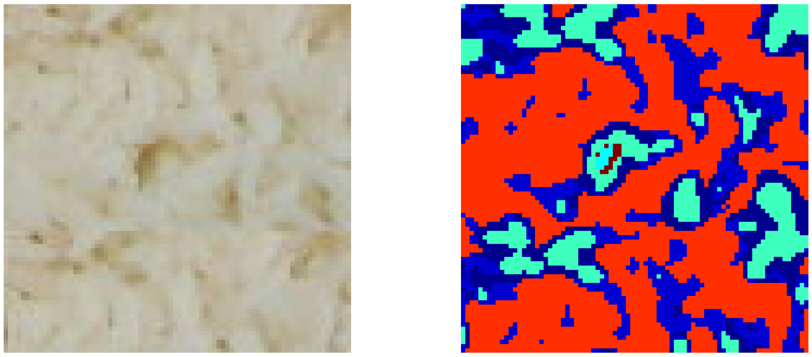


Figure 3: One block of "steamed rice" image in a sliding window (left) and its texton map (right)

Accuracy(%)	color histogram	gradient histogram	color+gradient histogram	texton histogram
Cross Validation	91.17	40.19	85.01	**92.65**
Two-halves Test	63.73	32.84	65.69	**69.12**

Table 3: The performances comparison of features.

- Gradient Histograms: This is a `SIFT` like feature. We first calculate the gradient of pixels in a window and then calculate the histogram of gradient directions for those pixels with magnitude greater than average. We figured out the best colorspace for gradient histograms empirically (see Table 2). To our surprise, `graylevel` yields the best performance.

- Texton Histograms: texton-based approach [5] is first developed to generate texture descriptor for texture classification and segmentation. It is also reported to be useful for object categorization [6]. Since the dishes are somewhat between textures and objects, textons are best suitable for our application. To maintain the rotational invariant property, we chose the rotational invariant filter bank used in [6] which are 3 Gaussians, 4 Laplacian of Gaussians (LoG) and 4 first-order derivatives of Gaussian (DoG). The filter bank is applied to all channels in current color space separately.

  Figure 3 gives an example of texton map of "steamed rice" image. Although our textons are obtained by clustering both edge information and color information, they are greatly dependent on the color of the source image. It is because that major differences between dishes are dominated by color. The texton clustering algorithm automatically learned this property in the clustering process.

  Theoretically speaking, color textons combines the feature of color and texture information. and thus should be self-contained and yield best performance over the other two features. We conducted experiments (see Table 3) and verified the hypothesis. Color texton histograms works even better than the combination of color and gradient histograms. Therefore, we decide to use texton histograms as the only feature for our application.

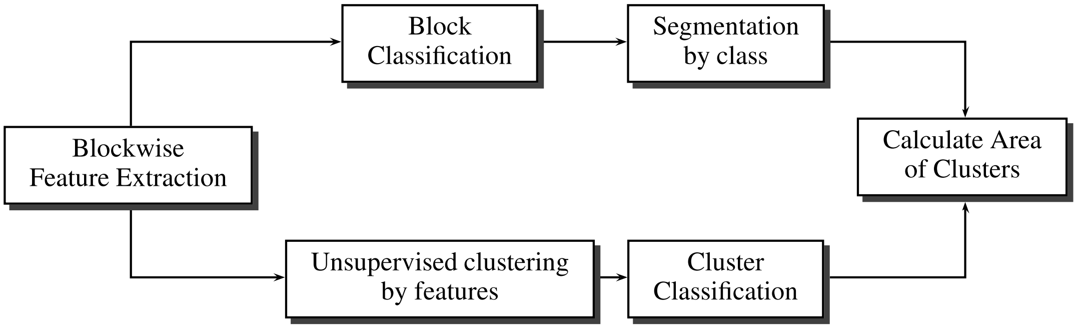


Figure 4: Two different procedures in testing phase: classification then clustering (upper path), and clustering then classification(bottom path).

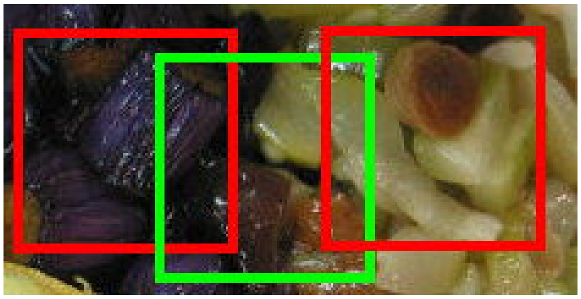


Figure 5: Two different cases of sliding windows on the source images: on one dish (red) and on the boundary(green).

5 Texture Segmentation

The "texture segmentation and classification" problem is a well-known hard problem. In general, there are two different ways to solve as shown in Figure 4: "classification then segmentation" or "segmentation then classification". There are no state-of-art methods that solve both problem all at once. Whether segmentation or clustering first must been determined by the nature of the problem to solve.

5.1 Classification then Segmentation

Similar to training data subsampling, sliding window (or blockscan) method is used again to generate features for each block of **plate** images. But overlapping is allowed here to get dense predictions. The predicted image is like a mosaic where each piece/block is a prediction result. We expect to get a clean predicted image with obvious boundary between dishes.

However, it turns out the predicted mosaic is quite noisy at the boundary between two adjacent dishes. There are couples of irrelevant dishes bumped out on the boundary. (Say dish C bumped out on the boundary of A and B). This is because when the sliding window is on the boundary (see green rectangle in Figure 5, where the histogram is essentially the mixture of histograms the two adjacent dishes. The mixture sometimes looks similar to irrelevant dishes and leads to wrong predictions. The situation is getting serious when the size of window is large because a large window takes more steps to cross the boundary. Reducing the size of window cause another problem: the smaller the size of the window is, the less stable the histogram will be.

It's hard to find a good tradeoff between the robustness and clear boundary. So we decided to abort this "classification then segmentation" method.

5.2 Segmentation then Classification

The **plate** image is first segmented using an unsupervised color texture segmentation algorithm called JSEG [4] with default parameters. Next, the histogram calculated in each segment is converted into a feature vector and then passing through the classification pipeline. Since the segment is usually large enough to produce robust histogram. The prediction result is more stable than small windows/blocks.

But there are still challenges when using this method. The unsupervised segmentation sometimes **over-segmented** or **under-segmented**. If a region is seriously over-segmented that each segment is too small to produce stable histogram features the prediction will fail. On the contrary, sometimes **under-segmented** happens. Two dishes are divided into the same segment. Then the mixture of histograms of both dish produces crappy results. For example, the segments of "rice" and the "table surface" (stainless iron) are often segmented into one region by JSEG because these two textures are both white and look alike in some sense.

6 The Real-life Testing Result

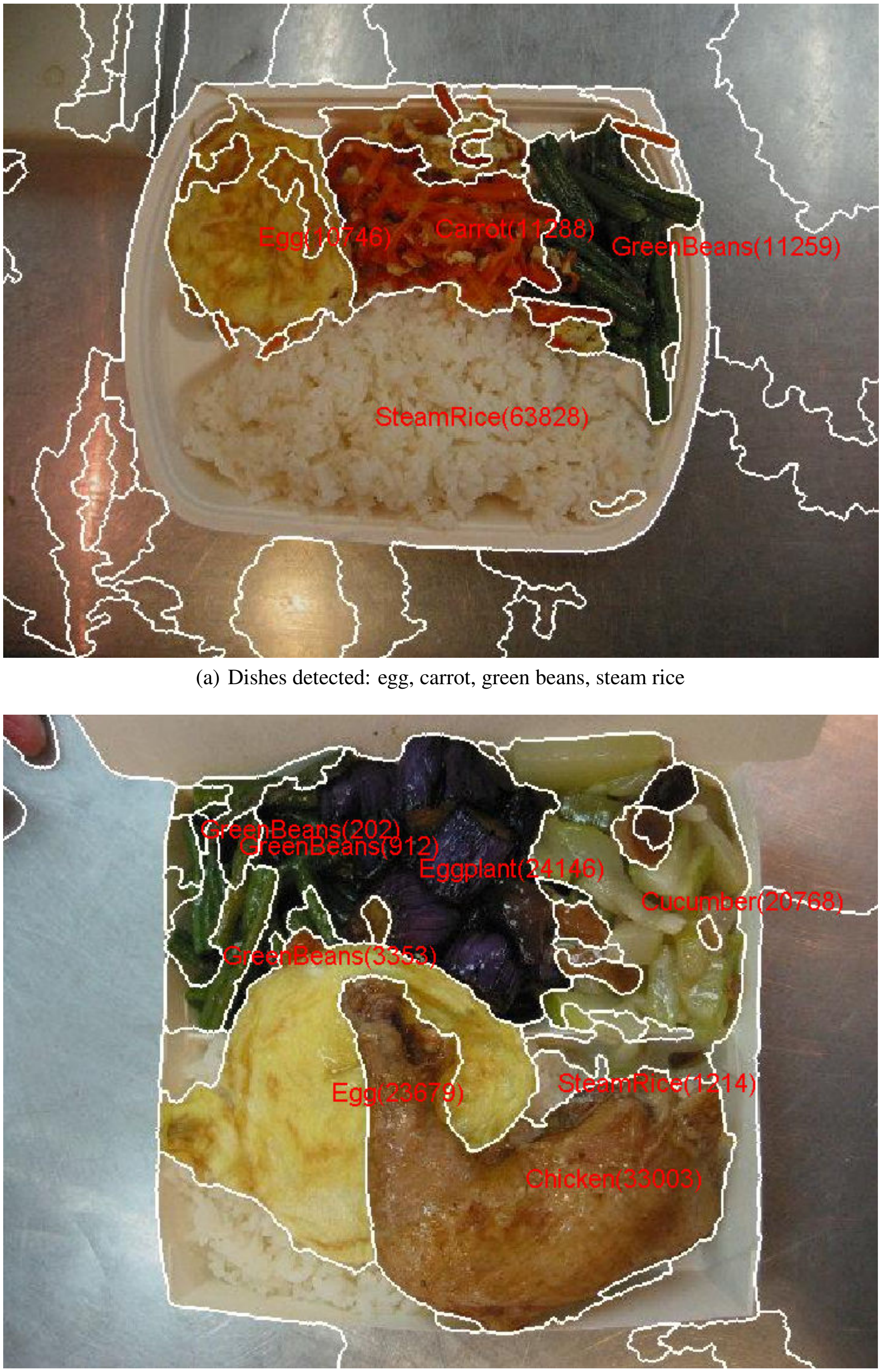
The final result using "texton histogram" and "segmentation then classification" is shown in Figure 6. Those segments with high prediction confidence (over a threshold) are marked with the predicted names of the dishes. And the area measurements (number of pixels) are shown in the parenthesis.

The overall results looks promising, but there are still some broken predictions/measurements. As you can see, the "green beans" in Figure 6(b) is over segmented. Fortunately, even it is over-segmented, we still get correct predictions and obtains multiple segments of "green beans". On the other hand, the case of "rice" is poor in both Figure 6(a) and 6(b). The boundary between "rice" and "table surface" is missing. So the area calculated for "rice" is much larger than it should be.

Besides the segmentation problem, poor illumination change tolerance is also serious that deteriorate the final prediction accuracy by a large margin (this can be observed by gap between cross-validation and two-halves testing accuracy, too.). We further investigate this problem by comparing the RGB color histograms of the tray and plate image of the same dish. Figure 7 shows the comparison. As we can see, the red and green channels are drifted but the blue channels are identical. I finally realized that the tray images are illuminate by heating lamps which emit strong intensity in red and yellow channels (and IR as well). But the plate image taken on the checkout counter is illuminated with fluorescent light instead. Therefore we have two different histogram for the same dish. This problem can be solved hopefully by either take the **tray** image at the counter or using color calibration cards.

7 Summary

In this project, we successfully developed a dish recognition system targeted to ease the checkout process in a cafeteria. Texton histograms are empirically proved to be best effective over general color histograms and gradient histograms. We also identified that "segmentation then classification" is the better viable way for such application. Several difficulties including poor segmentation and illumination variation are also observed pro-



(a) Dishes detected: egg, carrot, green beans, steam rice

(b) Dishes detected: green beans*3, eggplant, cucumber, egg, chicken, steam rice

Figure 6: Recognized dish (the red label superimposed) on and their measured area (shown in the parenthesis).

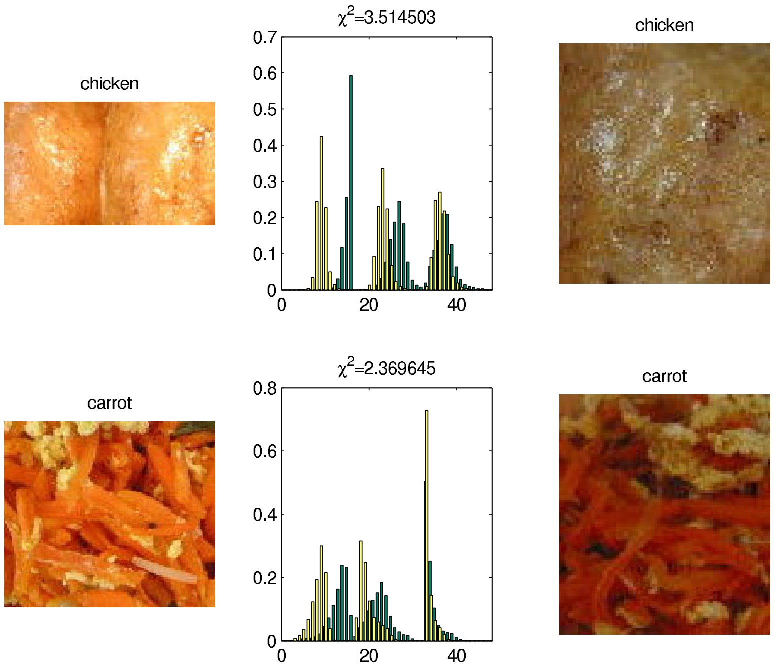


Figure 7: The color histogram differences under different illumination: heating lamp in **tray image**(images in the left column, yellow histogram) and fluorescent light in **plate image** (images in the right column, green histogram).

viding guidance for future work.

Appendix

A Background of Cafeteria

We, Taiwanese people, are always looking for ways making things fast, convenient and flexible. For most people living without families, they seldom cook themselves. Because spending an hour cooking for only one serving is not economic. Most double-income families do not cook often either, because the parents are tired after working the whole day. Those people just want to fulfill their stomach as soon as possible rather than spending couple of hours in a romantic restaurant. The (Taiwanese) cafeterias are born in such scenario.

References

[1] R. Bolle. *VeggieVision: A Produce Recognition System*. IBM TJ Watson Research Center, 1996.

[2] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[3] K. Chang, S. Liu, H. Chu, J. Hsu, C. Chen, T. Lin, C. Chen, and P. Huang. The Diet-Aware Dining Table: Observing Dietary Behaviors over a Tabletop Surface. *LECTURE NOTES IN COMPUTER SCIENCE*, 3968:366, 2006.

[4] Y. Deng and b. s. Manjunath. Unsupervised segmentation of color-texture regions in images and video. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(8):800–810, 2001.

[5] M. Varma and A. Zisserman. A Statistical Approach to Texture Classification from Single Images. *International Journal of Computer Vision*, 62(1):61–81, 2005.

[6] J. Winn, A. Criminisi, and T. Minka. Object Categorization by Learned Universal Visual Dictionary. *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, 2, 2005.