Noname manuscript No.
(will be inserted by the editor)

Real-time Food Recognition System on a Smartphone

Yoshiyuki Kawano · Keiji Yanai

Received: date / Accepted: date

Abstract We propose a mobile food recognition system the purposes of which are estimating calorie and nutritious of foods and recording a user's eating habits. Since we adopt image recognition methods which are suitable for mobile devices and all the processes on image recognition is performed on a smartphone, the system does not need to send images to a server and runs on an ordinary smartphone in a real-time way.

To recognize food items, a user draws bounding boxes by touching the screen first, and then the system starts food item recognition within the indicated bounding boxes. To recognize them more accurately, we segment each food item region by GrubCut, extract image features and finally classify it into one of the one hundred food categories with a linear SVM. As image features, we adopt two kinds of features: one is the combination of the standard bag-of-features and color histograms with χ^2 kernel feature maps, and the other is a HOG patch descriptor and a color patch descriptor with the state-of-the-art Fisher Vector representation. In addition, the system estimates the direction of food regions where the higher SVM output score is expected to be obtained, and it shows the estimated direction in an arrow on the screen in order to ask a user to move a smartphone camera. This recognition process is performed repeatedly and continuously. We implemented this system as an Android smartphone application so as to use multiple CPU cores effectively for real-time recognition.

In the experiments, we have achieved the 79.2% classification rate for the top 5 category candidates for a 100-category food dataset with the ground-truth bounding boxes when we used HOG and color patches with the Fisher Vector coding as image features. In addition, we obtained positive evaluation by a user study compared to the food recording system without object recognition.

Keywords Food Recognition · Dietary Recording · Smartphone · Fisher Vector · Mobile Image Recognition

The University of Electro-Communications, Tokyo
1-5-1 Chofugaoka, Chofu-shi, Tokyo, 182-8585 Japan
Tel.: +81-42-440-7365
Fax: +81-42-443-5334
E-mail: {kawano-y,yanai}@mm.inf.uec.ac.jp

1 Introduction

In recent years, food habit recording services for smartphones such as iPhone and Android phones have become popular. They can awake users' food habit problems such as bad food balance and unhealthy food trend, which is useful for disease prevention and diet. However, most of such services require selecting eaten food items from hierarchical menus by hand, which is too time-consuming and troublesome for most of the people to continue using such services for a long period.

Due to recent rapid progress of smartphones, they have obtained enough computational power for real-time image recognition. Currently, a quad-core CPU is common as a smartphone's CPU, which is almost equivalent to a PC's CPU released several years ago in terms of performance. Old-style mobile systems with image recognition need to send images to high-performance servers, which must makes communication delay, requires communication costs, and the availability of which depends on network conditions. In addition, in proportion to increase number of users, more computational resources of servers is also required, which makes it difficult to recognize objects in a real-time way.

On the other hand, image recognition on the client side, that is, on a smartphone is much more promising in terms of availability, communication cost, delay, and server costs. It needs no wireless connection and no commutation delay. Then, by taking advantage of rich computational power of recent smartphones as well as recent advanced object recognition techniques, in this paper, we propose a real-time food recognition system which runs on a common smartphone.

To do that, we adopt two kinds of image recognition methods: one is the combination of the standard bag-of-features(BoF) and color histograms with χ^2 kernel feature maps, and the other is a HOG patch descriptor and a color patch descriptor with the state-of-the-art Fisher Vector representation.

In the first method, we adopt a bag-of-feature representation with SURF local features and a standard color histogram as image features, and we use a linear SVM and a fast χ^2 kernel based on kernel feature maps [Vedaldi and Zisserman(2012)] as an image classifier.

In the second method, we adopt the more sophisticate method than the first one. Some effective image representations have been proposed recently, which can boost recognition performance in case of using a linear SVM. Especially, Fisher Vector [Perronnin and Dance(2007), Perronnin et al.(2010)] is known as a high performance method among the recent image representations. It is suitable for a linear classifier, and was turned out that it can improved recognition accuracy more than popular combination of bag-of-features (BoF) and a non-linear SVM. Moreover, while BoF needs larger dictionary to improve recognition accuracy, larger dictionary brings increase of computational cost for searching nearest visual words. On the other hand, Fisher Vector is able to achieve high recognition accuracy with even small dictionary, and low computational complexity. This is an advantage for mobile devices.

Thus adopting Fisher Vector as encoding method is better in terms of recognition accuracy and processing time for a mobile object recognition. However, a system on a smartphone does not exist so far that carries out rapid and high precision image recognition with Fisher Vector. Then we propose a recognition method for a smartphone which is rapid and accurate by making good use of computational resource of the smartphone with Fisher Vector.

In the experiments, we have achieved 79.2% classification rate within the top five candidates for a 100-category food dataset with ground-truth bounding boxes using the second method. This results outperformed the existing food recognition method running on the server-side regarding classification accuracy on the same 100 food categories. The process-

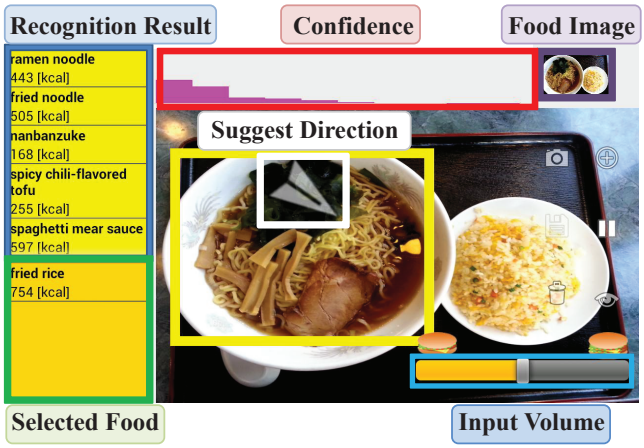

Fig. 1 The screen-shot of the main screen of the proposed system.

ing time by the second method for 100 kinds of food image recognition takes only 0.065 second.

Here, we explain the implemented system briefly. Figure 1 shows the screen-shot of the proposed system which runs as an Android smartphone application. First a user point a smartphone camera to foods, draws a bounding box (represented in the yellow rectangular in the figure) by dragging on the screen, and then food image recognition is activated for the given bounding box. The top five candidates for the yellow bounding box are shown on the left side of the screen. If a user touches one of the candidate items, the food category name and the photo are recorded as a daily food record in the system. In addition, the proposed system has functions on automatic adjustment of bounding boxes based on the segmentation result by GrubCut [Rother et al.(2004)], and estimation of the direction of the expected food regions based on Efficient Sub-window Search (ESS) [Lampert et al.(2008)]. Since this recognition process is performed repeatedly a user can search for good position of a smartphone camera to recognize foods accurately by moving it continuously without pushing a camera shutter button.

To summarize our contribution in this paper, it consists of five folds: (1) implementing an interactive and real-time food recognition and recording system running on a consumer smartphone, (2) using a liner SVM with a fast χ^2 kernel for fast and accurate food recognition as the first method, (3) using a Fisher Vector coding with a HOG and color patches as the second method, (4) adjustment of the given bounding box, and (5) estimation of the direction of the expected food region automatically.

The rest of this paper is organized as follows: Section 2 describes related work. In Section 3, we explain the overview of the proposed system. In Section 4, we explain the detailed method for food recognition. In Section 5, we describe the detail of implementation of the proposed system as a smartphone application. In Section 6 describes the experimental results and user study, and in Section 7 we conclude this paper.

2 Related Work

2.1 Food Recognition

Food recognition is difficult task, since appearances of food items are various even within the same category. This is a kind of fine-grained image categorization problem. As food recognition, Yang *et al.* [Yang et al.(2010)] proposed pairwise local features which exploit positional relation of eight basic food ingredients. Matsuda *et al.* [Matsuda et al.(2012)] proposed a method for recognition multiple-food images which first detected food regions by several detectors including circle detector, JSEG [Deng and Manjunath(2001)] and Deformable Part Model(DPM) [Felzenszwalb et al.(2010)], next recognized food categories with the extracted color, texture, gradient, and SIFT using multiple kernel learning (MKL).

As a food recording system with food recognition, Web application FoodLog [Kitamura et al.(2008), Kitamura et al.(2009)] estimates food balance. It divides the food image into 300 blocks, from each blocks extracts color and DCT coefficients, next classifies to five groups such as staple, main dish, side dish, fruit, and non-food.

The TADA dietary assessment system [Mariappan et al.(2009)] has food identification and quantity estimation, although it has some restriction that food must be put on white dishes and food photos must be taken with a checkerboard to food quantity estimation. Recently, the same research group proposed a method to estimate volume of foods which was based on a 3D template matching [Chae et al.(2011), He et al.(2013)].

In all the above-mentioned systems image recognition processes perform on servers, which prevents systems from being interactive due to communication delays. On the other hand, our system can recognize food items on a client side in a real-time way, which requires no communication to outside computational resources and enables user to use it interactively. Note that our current system recognize only 100 food categories, and it requires user's assistances to estimate food volumes by touching a slider on the system screen. However, user's assistances are very easy, since our system is an interactive system on a smartphone which has a touching screen.

2.2 Mobile Device and Computer Vision

With the spread of smartphones, some mobile applications exploiting computer vision technique have been proposed. Google Goggles [1] is one of the most well-known mobile image recognition applications which recognize specific object such as famous landmarks, product logos, famous art and so on in photos taken by users, and return the names and some information on the recognized objects to users. However, recognition targets are limited to specific objects the appearance of which stays unchanged, which is different from this paper the target of which is generic objects.

Kumar *et al.* [Kumar et al.(2012)] proposed recognize 184 species application "Leafsnap". For a leaf image on the solid light-colored background they segment and extract curvature-based shape features. Maruyama *et al.* [Maruyama et al.(2012)] proposed a mobile recipe recommendation system which extracted color feature, recognized 30 kinds of food ingredients and recommended food recipes based on the recognized ingredients on a mobile device. In the above-mentioned mobile vision systems, generic object recognition

[1] http://www.google.com/mobile/goggles/

for leaves and food ingredients was performed, while we tackle category-level object recognition on foods on a mobile device.

As an interactive mobile vision system, Yu *et al.* [Yu et al.(2011)] proposed Active Query Sensing (AQS) the objective of which is localization of the current position by matching of street-view photos. When the system fails in location search, it suggests the best viewing direction for the next reference photo to a user based on the pre-computed saliency on each location. Our system is also built as an interactive system which detects object regions based on the bounding boxes a user draws and suggests the direction of food regions where the higher evaluation output score is expected.

3 System Overview

The final objective of the proposed system is to support users to record daily foods and check their food eating habits. To do that easily, we built-in food image recognition technique on the proposed system. In this paper, we mainly describe a food image recognition part of the proposed system.

Processing flow of typical usage of the proposed system is as follows (See Figure 2 as well):

1. A user points a smartphone camera toward food items before eating them. The system is continuously acquiring frame images from the camera device in the background.
2. A user draws bounding boxes over food items on the screen. The bounding boxes will be automatically adjusted to the food regions. More than two bounding boxes can be drawn at the same time. .
3. Food recognition is carried out for each of the regions within the bounding boxes. At the same time, the direction of the region having the higher evaluation score is estimated for each bounding box.
4. As results of food recognition and direction estimation, the top five food item candidates and the direction arrows are shown on the screen.
5. A user selects food items from the food candidate list by touching on the screen, if found. Before selecting food items, a user can indicate relative rough volume of selected food item by the slider on the right bottom on the screen. If not, user moves a smartphone slightly and go back to 3.
6. The calorie and nutrition of each of the recognized food items are shown on the screen.

In addition, a user can see his/her own meal record and its detail including calories and proteins of each food items on the screen as shown in Figure 3(a). Meal records can be sent to the server, and a user can see them on the Web (Figure 3(b)).

4 Methods

In this section, we explain the following three processing steps:

1. Adjustment of bounding boxes.
2. Recognition of food items within the given bounding boxes.
3. Estimation of the direction of the possible food region.

Regarding recognition of food items, we propose two kinds of the methods. In this section, we will explain both of them.

Fig. 2 System process flow

(a) Meal records on the screen.

(b) Meal records on the Web.

Fig. 3 Meal records.

4.1 Bounding Box Adjustment

Before starting to recognize food items for the frame images taken by a smartphone camera, at first the system requires that a user draws bounding boxes which bounds food items on the screen by dragging along the diagonal lines of the boxes.

First, a user draws bounding boxes roughly on the screen by dragging. The bounding boxes a user draws are not always accurate and sometimes they are too large for actual food regions. Therefore, we apply well-known graph-cut-based segmentation algorithm GrabCut [Rother et al.(2004)], and then modify the bounding boxes so as to fit them to food regions segmented by GrabCut. GrabCut needs initial foreground and background regions as seeds. Here, we provide GrubCut the regions within bounding boxes as foreground and the areas out of the doubly-extended boxes of the original bounding boxes as background. Since the computational cost of GrabCut is relatively high in the real-time recognition sys-

tem, the bounding box adjustment is performed only once after the original bounding box was drawn.

4.2 Food Item Recognition

Food recognition is performed for each of the window images within the given bounding boxes. Firstly, image features are extracted from each window, secondly feature vectors are built based on the pre-computed codebook, and finally we evaluate the feature vectors with the trained linear SVMs on 100 food categories. The top five candidates which have the top five SVM scores over 100 categories are shown on the screen as food item candidates for the given bounding boxes.

As image features, we adopt two kinds of features: one is the combination of the standard bag-of-features and color histograms with χ^2 kernel feature maps, and the other is a HOG patch descriptor and a color patch descriptor with the state-of-the-art Fisher Vector representation. We explain both of the methods and compare them in Section 6.

4.2.1 Image Feature: Bag-of-features and Color Histogram

As the first image features, we adopt bag-of-features [Csurka et al.(2004)] and color histograms both of which are standard image features for generic object recognition. It is shown that fusion of many kinds of image features is effective for food recognition [Matsuda et al.(2012)]. Since we aims for implementing a mobile recognition system which can run in the real-time way, image features to be extracted should be minimum requisites. Then we evaluated and compared performance and computational costs of various kinds of common standard image features including global and local features such as color histogram, color moment, color auto correlogram, Gabor texture feature, HoG (Histogram of Oriented Gradient) [Dalal and Triggs(2005)], Pyramid HoG, and Bag-of-features with SURF features [Bay et al.(2008)]. Finally we chose the combination of color histogram and Bag-of-features with SURF. To save computational cost, if the longer side of a bounding box is more than 200, the image extracted from the bounding box is resized so that the longer side becomes 200 preserving its aspect ratio.

Color Histogram: We divide a window image into 3×3 blocks, and extract a 64-bin RGB, color histogram from each block. Totally, we extract a 576-dim color histogram. Note that we examined HSV and La*b* color histograms as well, and RGB color histogram achieved the best among them.

Bag-of-fetures with SURF: As local features, we use dense-sampled SURF features. SURF [Bay et al.(2008)] is an invariant 64-dim local descriptor for scale, orientation and illumination change. We sample points by dense sampling in scale 12 and 16 with every 8 pixel with the window. To convert bag-of-features vectors, we prepared a codebook where the number of codeword was 500 by k-means clustering in advance. We apply soft assignment [Philbin et al.(2008)] by 3 nearest-neighbor assigned reciprocal number of Euclid distance to the codeword, also we use fast approximated-nearest-neighbor search based kd-tree to search the codebook for the corresponding codeword for each sampled point. Finally, we create a 500-dim bag-of-SURF vector.

Feature Embedding: As a classifier, we use a linear SVM because of its efficiency on computation and memory. Although a linear SVM is fast and can save memory, classification accuracy is not as good as a non-linear kernel SVM such as a SVM with χ^2-RBF kernel. To compensate weakness of a liner SVM, we use explicit embedding technique. In this work, we adopt kernel feature maps. Vedaldi *et al.* [Vedaldi and Zisserman(2012)] proposed homogeneous kernel maps for χ^2, intersection, Jansen-Shanon's and Hellinger's kernels, which are represented in the closed-form expression. We choose mapping for χ^2 kernel and we set a parameter so that the dimension of mapped feature vectors are 3 times as many as the dimension of original feature vectors as shown in the following equation:

$$\phi(x) = \sqrt{x} \begin{bmatrix} 0.8 \\ 0.6\cos(0.6\log x) \\ 0.6\sin(0.6\log x) \end{bmatrix} \tag{1}$$

This mapping can be applied for L1-normalized histogram [Vedaldi and Zisserman(2012)]. Then we apply this to L1-normalized color histograms and Bag-of-SURF vectors. Finally we obtained a 1728-dim vector as a color feature, and a 1500-dim vector as a BoF vectors.

4.2.2 Image Feature: HOG and Color Patch with Fisher Vector

As the second image features, we adopt a HOG patch and a Color patch as local descriptor, and Fisher Vector as representation of local descriptors.

HOG Patch: Histogram of Oriented Gradients(HOG) was proposed by N.Dalal *et al.*[Dalal and Triggs(2005)]. It is similar to SIFT is terms of how to describe local patterns which is based on gradient histogram. The characteristics of HOG is no invariant for scale and rotation, which is different from the standard local descriptors such as SIFT [Lowe(2004)] and SURF [Bay et al.(2008)]. Since HOG description is very simple, it is able to describe much faster than the common local descriptors such as SIFT and SURF. This is important characteristic to carry out real-time recognition on a smartphone. In addition, it is able to extract local feature more densely to compensate no invariance on scale change and rotation. As a result, it improves recognition accuracy.

We extract HOG features as local features. We divide a local patch into 2×2 blocks (totally four blocks), and extract gradient histogram regarding eight orientations from each block. Totally, we extract 32-dim HOG Patch features. Then the 32-dim HOG Patch is L2 normalized to an L2 unit length. Note that we did not adopt HOG-specific normalization by sliding fusion. The HOG used here is a simpler-version of the original HOG. After extracting 32-dim HOG patch, PCA is applied each HOG patch to reduce dimensions from 32 to 24.

Color Patch: We use mean and variance of RGB value of pixels within a local patch as a Color Patch feature. We divide a local patch into 2×2 blocks, and extract mean and variance of RGB value of each pixel within each block. Totally, we extract 24-dim Color Patch features. PCA is applied without dimension reduction. The dimension of a Color Patch feature are kept to 24-dim.

Fisher Vector: Recently, Fisher Vector [Perronnin and Dance(2007),Perronnin et al.(2010)]

Recently, Fisher Vector [Perronnin and Dance(2007),Perronnin et al.(2010)] becomes known as a high performance method to represent a set of local features. It can decrease quantization error than bag-of-features [Csurka et al.(2004)] by using of a high order statistic. It was turned out that it can improved recognition accuracy more than bag-of-features (BoF) and other recent representations such as Locality-constrained Linear Coding (LLC) [Wang et al.(2010)][Chatfield et al.(2011)], and most of the higher rank teams in the large-scale visual recognition challenge (LSVRC) used Fisher Vector [Jia et al.(2012)].

Moreover, while BoF needs larger dictionary to improve recognition accuracy, larger dictionary brings increase of computational cost for searching nearest visual words. On the other hand, Fisher Vector is able to achieve high recognition accuracy with even small dictionary, and low computational complexity. This is an advantage for mobile devices.

According to [Perronnin et al.(2010)], we encode local descriptors into Fisher Vector. We choice probability density functions (pdf) as Gaussian mixture model (GMM). Then pdf is given as follows.

$$p(x|\theta) = \sum_{i=1}^{K} \pi_i \mathcal{N}(x|\mu_i, \Sigma_i) \tag{2}$$

where x is a local descriptor, K is number of component of Gaussian, $\theta = \{\pi_i, \mu_i, \Sigma_i, i = 1, ..., K\}$ is a parameter of GMM. π_i is the mixing coefficient, μ_i is a mean vector and Σ_i is a covariance matrix. At this point, we assume that the covariance matrix is diagonal and diagonal elements are presented variance vector σ^2.

The probability of x_t is belong component i (estimated posterior probability) is given as follows.

$$\gamma_t(i) = \frac{\pi_i \mathcal{N}(x_t|\mu_i, \Sigma_i)}{\sum_{j=1}^{N} \pi_j \mathcal{N}(x_t|\mu_j, \Sigma_j)} \tag{3}$$

Then the gradient with respect to the mean and variance is defined as follows,

$$\mathcal{G}^X_{\mu,i} = \frac{1}{T\sqrt{\pi_i}} \sum_{t=1}^{T} \gamma_t(i) \left(\frac{x_t - \mu_i}{\sigma_i} \right) \tag{4}$$

$$\mathcal{G}^X_{\sigma,i} = \frac{1}{T\sqrt{2\pi_i}} \sum_{t=1}^{T} \gamma_t(i) \left[\frac{(x_t - \mu_i)^2}{\sigma_i^2} - 1 \right] \tag{5}$$

Finally, gradient $\mathcal{G}^X_{\mu,i}$ and $\mathcal{G}^X_{\sigma,i}$ are calculated for all the Gaussian. Fisher Vector $\mathcal{G}^X_\theta$ is their concatenation. Therefore Fisher Vector is $2KD$-dimensional.

In this paper, the number of component of Gaussian is 32 and local descriptors reduced to 24 dimensions by PCA. Thus each feature vector is 1536-dimensional. To improve recognition accuracy, we apply power normalization ($\alpha = 0.5$) and L2 normalization [Perronnin et al.(2010)].

4.2.3 Classification

As a classifier, we use a linear kernel SVM, and we adopt the one-vs-rest strategy for multi-class classification. In the experiment, since we prepared 100 food categories, we trained 100 linear SVM classifiers.

Linear kernel is defined as the inner product of two vectors. In advance, we computed the inner product of a support vector and the weight of the corresponding support vector, then Linear SVM can be written as follows:

$$f(\mathbf{x}) = \sum_{i=1}^{M} y_i \alpha_i K(\mathbf{x}, \mathbf{x_i}) + b = \sum_{i=1}^{M} y_i \alpha_i \langle \mathbf{x}, \mathbf{x_i} \rangle + b$$
$$= \langle \sum_{i=1}^{M} y_i \alpha_i \mathbf{x_i}, \mathbf{x} \rangle + b = \langle \mathbf{w}, \mathbf{x} \rangle + b \quad (6)$$

where $\mathbf{x}$ is an input vector, $f(\mathbf{x})$ is an output SVM score, $\mathbf{x_i}$ is support vector, $y_i \in \{+1, -1\}$ is a class label, α_i is a weight of the corresponding support vector, b is a bias vector, and M is the number of support vector. By this transformation, we can save memory to store support vectors as well as calculation of kernels. Therefore, when N is the dimension of feature vector, calculation of a SVM score requires $\mathcal{O}(N)$ operations and $\mathcal{O}(N)$ memory space. We train SVMs with LIBLINEAR [Fan et al.(2008)] in off-line.

For both cases of the first and the second image features, we combine the output values of both linear SVMs for gradient-based features and color-based features in the late fusion manner. The weight to combine two SVM output values is estimated by cross-validation in the experiments.

4.3 Estimation of the more reliable direction

In case that no categories with high SVM scores are obtained, camera position and viewing direction should be changed slightly to obtain more reliable SVM evaluation scores. To encourage a user to move a smartphone camera, the proposed system has a function to estimate the direction to get the more reliable SVM score and show the direction as an arrow on the screen as shown in Figure 1.

To estimate the direction of the food regions with more reliable SVM score, we adopt an effective window search method for object detection, Efficient Sub-window Search (ESS) [Lampert et al.(2008)], which can be directly applied for a combination of a liner SVM and bag-of-features. Note that estimation of the more reliable direction can be carried out for the first type of the image feature, because ESS assumes image features is represented by bag-of-features.

The weighting factor $\mathbf{w}$ of an input vector of the SVM classifier represented in Equation 6 can be decomposed into a vector $\mathbf{w}^+$ including positive elements and a vector $\mathbf{w}^-$ including negative elements.

$$\mathbf{w} = \mathbf{w}^+ + \mathbf{w}^- \quad (7)$$

Therefore, an SVM output score for one rectangular region can be calculated in $\mathcal{O}(1)$ operation by making use of $\mathbf{w}^+$ and $\mathbf{w}^-$ integral images according to the ESS method [Lampert et al.(2008)]. In case of the above-mentioned soft assignments to codewords, the output score can be calculated by a product of w and assigned values to codewords. We search for the window which achieves the maximum SVM score by Efficient Sliding Windows search so as to keep more than 50% area being overlapped with the original window. Finally the relative direction to the window with the maximum score from the current window are shown as an arrow on the screen (See Figure 1).

5 Implementation on a Smartphone

We implemented two version of the systems as Android applications for an Android smartphone which has a quad-core CPU. The first version uses bag-of-features and color histogram, while the second version uses a HOG and Color patch with Fisher Vector. We implemented both systems as multi-threaded systems for using multiple CPU cores effectively.

In both systems, we first apply GrabCut to adjust the size of the given bounding box after a bounding box is drawn. Because the computational cost of GrabCut is relatively high, bounding box adjustment is carried out once for one bounding box. After the adjustment was finished, feature extraction begins. The ways to extract features are different for both systems.

For the first version, high-cost SURF descriptor extraction, assignment to codewords, evaluation of 50 kinds of fast χ^2 linear SVMs and direction estimation are carried out over four cores in parallel, while low-cost color histogram extraction is performed on a single core. Figure 4 shows the flow of the processing steps and usage of four CPU cores in the first version system.

For the second system, we parallelized extraction for HOG Patch and Color Patch feature. Both extractions are carried out over two cores in parallel. Extract descriptor, reduce to dimension by PCA, encoding into Fisher Vector, power normalization, L2 normalization and classify with SVMs are carried out over 2 cores in parallel for each feature, totally over 4 cores in parallel. Figure 5 shows the flow of the processing steps and usage of four CPU cores in the second version system. Note that estimation of the direction of foods is not implemented in the second version system, since the ESS method which the method on direction estimation is based on assumes that the feature representation is bag-of-features.

For the second system, we devised speeding-up of computation by pre-computation. The gradient of Fisher Vector with respect to the mean Eq.(4) is deformation as follows to decrease the number of operation. The number of local descriptors T is much bigger than the number of GMM component K and the dimension of local descriptor D, and calculate Eq.(4) for each component, so effectively.

$$\mathcal{G}^X_{\mu,i} = \frac{1}{\sqrt{\pi_i}\sigma_i}\frac{1}{T}\sum_{t=1}^{T}\gamma_t(i)(x_t - \mu_i) \tag{8}$$

Moreover in advance, we computed the term of calculate posterior probability by GMM and the gradient with respect to the mean and sigma in off-line, and we create the lookup table for acceleration (using for calculate posterior probability $\log \pi_i - 0.5 \times \log |\Sigma_i|$, $1/\sqrt{2\pi_i}$ and $1/\sigma^2$ of Eq.(5) and $1/\sqrt{\pi_i}\sigma_i$ of Eq.(8)).

We trained SVMs in off-line. And all the parameter values using recognition steps are loaded on main memory (eigenvalue and eigenvector for PCA, created lookup table, mean of GMM, weight vectors of SVMs). Although all the values can be stored on main memory in advance, Fisher Vector is able to bring better recognition result with even smaller dictionary than that of BoF. We also set the dimension of feature vectors smaller reduced by PCA. As a result, memory space required for Fisher Vector is smaller than the space for codebook for conventional BoF, and in respect of memory Fisher Vector is also superior to BoF.

Regarding memory space, the first-version system adopts 1500-dim SURF-BoF and 1728-dim color histogram, while the second-version system adopts 1536-dim HOG Patch-FV and 1536-dim Color Patch-FV. The feature vector of the second one is more compact but more dense. And then many values are loaded on memory to encode Fisher Vector faster, sum of these require only less one fifth in case of HOG Patch and less one fourth in case of

Table 1 Classification rate within the top and top 5 candidates by the proposed methods, the server-side method proposed by [Matsuda et al.(2012)], and the extended version of the proposed method.

method	top1	top5
SURF-BoF+ColorHistogram	42.0	68.2
HOG Patch-FV+Color Patch-FV	49.7	77.6
HOG Patch-FV+Color Patch-FV(flip)	51.9	79.2
MKL[Matsuda et al.(2012)]	51.6	76.8
Extended HOG Patch-FV+Color Patch-FV(flip)	59.6	82.9

Color Patch than memory space of codebook for BoF, respectively. Java heap (mainly except image processing) and native heap (mainly image processing) the implemented application required were about 16MB and 3MB. We realize the mobile system with low memory space. Therefore we are able to increase the number of recognition target and feature dimension.

6 Experiments

In this section, we describe experimental results regarding recognition accuracy and processing time. In addition, we also explain the evaluation result by user study.

In the experiments, we prepared one hundred categories food image dataset which has more than 100 images per categories and all the food item in which are marked with bounding boxes. The total number of food images in the dataset is 12,905. Figure 6 shows all the category names and their sample photos.

We set the validation data and the test data for each category as 20 images. The rest is the train data, and evaluated classification rate 5 trials, randomly changing the images in the five-fold cross validation manner. We used Samsung Galaxy Note II (1.6GHz 4 cores, 4 threads, Android 4.1) for measuring the processing time of image recognition.

6.1 Evaluation on classification accuracy

In this experiments, we compare the two types of the proposed systems with server-side recognition system by Matsuda *et al* [Matsuda et al.(2012)].

Figure 7 shows classification rate of each recognition method and Table 1 shows classification rate within top1 and top 5. SURF-BoF+Color Hist. achieved 42.0% and 68.2% within the top 1 and the top 5 candidate, HOG Patch-FV+Color Patch-FV achieved 49.7% and 77.6% classification rate, respectively. In case of adding flipped images as training data, the results were slightly improved, and it achieved 51.9% and 79.2% classification rate within top1 and top5. Adding flipped training images makes variability of training data increase, which is expected to be effective for HOG patch, since HOG is not invariant to rotation.

Our second approach is better than [Matsuda et al.(2012)] which is the server-side high cost recognition system. This shows food recognition on a smartphone is equivalent to the conventional server system in terms of recognition accuracy.

The bottom row in Table 1 shows the classification rate of the extended version of the second approach. In this extended version, the number of GMM components are doubled (64 Gaussians), the dimension of HOG Patches are 32 (original was 24), and spatial pyramid [Lazebnik et al.(2006)] was applied. Although the dimension of image features, processing time and memory requirements increased greatly, the classification rate was im-

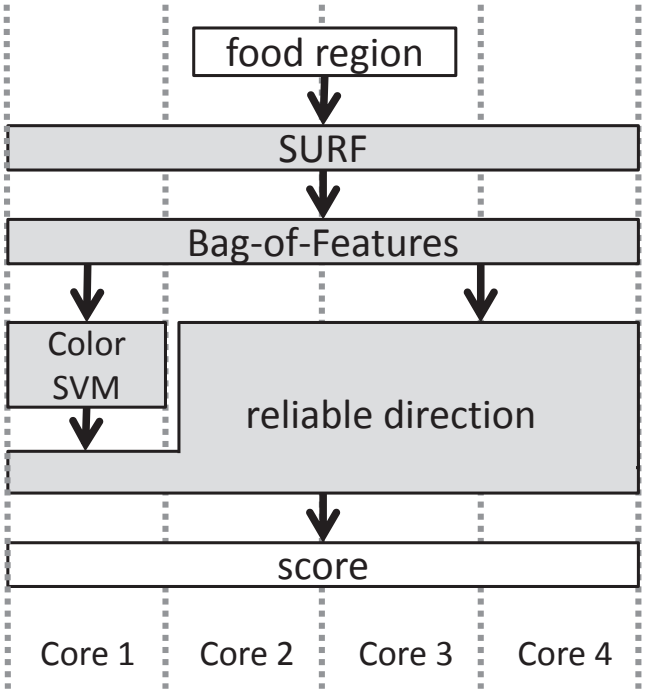

Fig. 4 Processing flow and assignment on CPU cores for the system using bag-of-features and color histogram.

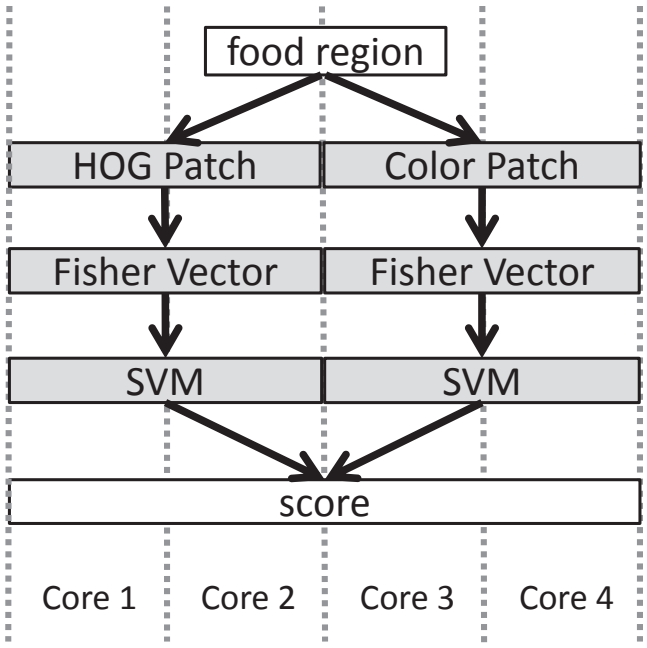

Fig. 5 Processing flow and assignment on CPU cores for the system using a HOG patch and a Color patch with Fisher Vector

Fig. 6 100 kinds of food images which are recognition targets in the paper

proved much. This version can be carried out on not a smartphone but a server, since the dimension of Color-FV and HOG-FV are 15,360 and 20,480, respectively. We show these results for reference. Figure 8 shows the results by the original second method and the extended method.

As the next evaluations, we compare the results by single features. Firstly, we compare the results by gradient-based local features which are not only SURF-BoF and HOG Fisher Vector but also HOG-BoF. Secondly, we compare the results by color features which are color histogram, Color-patch Fisher Vector and Color-patch BoF.

Figure 9 shows the comparison with SURF-BoF, HOG Patch-BoF and HOG Patch-FV. First, we set that a step of dense grid sampling is every 8 pixels. The difference top1 and top5 classification rate from SURF-BoF and HOG Patch-BoF is only 0.62% and 2.1%, respectively, which means that SURF is slight better. However, HOG Patch-FV achieved higher performance than SURF-BoF, differ is 2.7% and 3.22%. And then HOG Patch extraction is very fast than SURF extraction. In this paper, we set that a step of dense grid sampling is every 6 pixels. Then classification rate was improved. In case of adding horizontally flipped images for training data, we achieved 36.3% and 63.2% classification rate with top1 and top5 candidates. The classification rate is 7.52% and 6.9% higher than SURF-BoF.

Next, we evaluated Color Patch feature. Figure 10 shows the color histogram, Color Patch-BoF and Color Patch-FV. Color histogram divides a given image into 3×3 blocks, and extract a 64-bin RGB color histogram from each block. Totally, we extract a 576-dim color histogram. Then we apply χ^2 kernel feature map. Finally we build a 1728-dim color

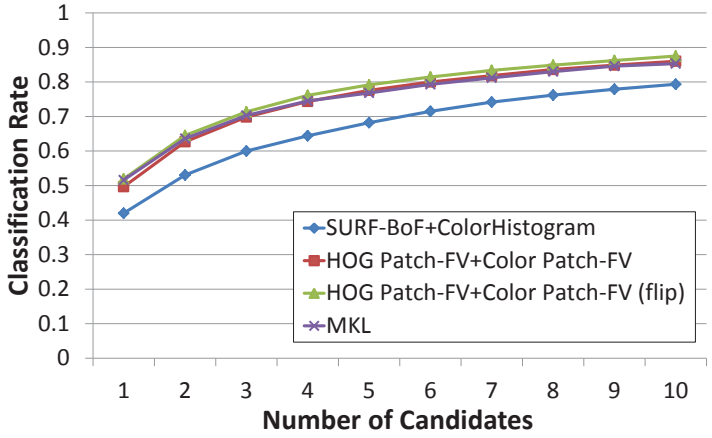

Fig. 7 Classification rate by the two proposed methods and the server-side method [Matsuda et al.(2012)].

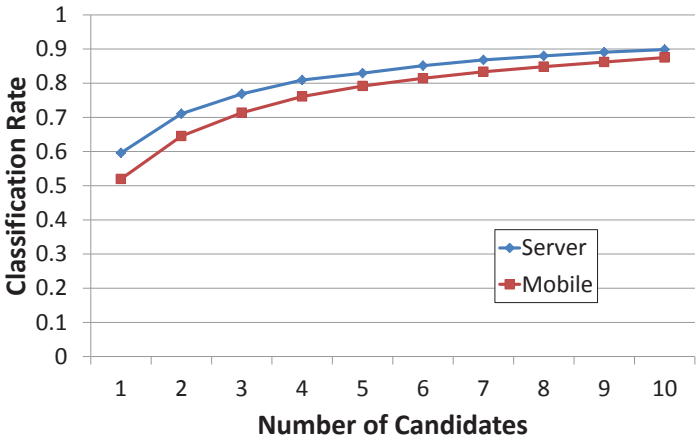

Fig. 8 Classification rate of the mobile version and the extended server version of the second proposed method (HOG and Color patch with Fisher Vector).

histogram. The difference top1 and top5 classification rate from Color Patch-BoF and color histogram is only 0.76% and 2.28%, Color Patch-BoF is slight better. However, in case of Color Patch-FV, classification rate is much improved, and top-1 and top-5 classification rate is 13.0% and 18.4% higher than color histogram. In case of adding horizontally flipped images for training data, we achieved 43.0% and 70.6% classification rate. In case of using only Color Patch feature, we achieved much better classification rate than the first-version system, which means that Fisher Vector representation for Color Patch feature is very effective for food recognition.

According to the experiments of recognition accuracy, we could successfully show effectiveness of our proposed method. Moreover, we achieved better results than the server-side result which was obtained by very high cost recognition method [Matsuda et al.(2012)].

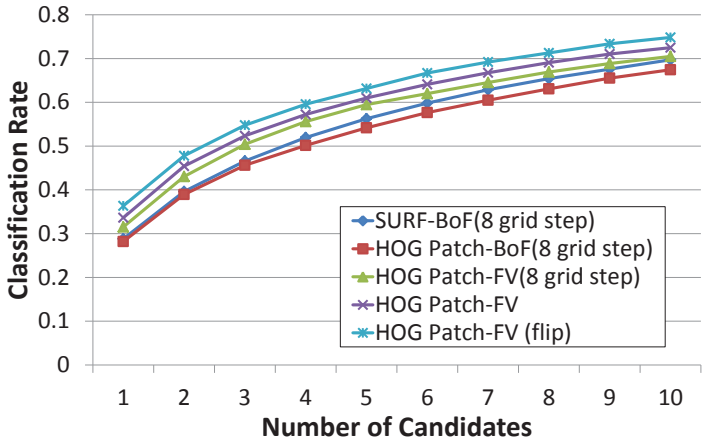

Fig. 9 Classification rate by SURF-BoF, HOG-Patch-BoF and HOG-Patch-Fisher-Vector with 6 or 8 grid step

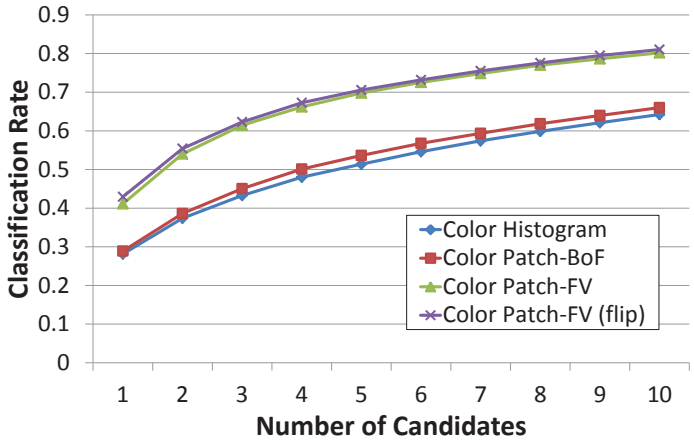

Fig. 10 Classification rate by Color Histogram, Color-Patch-BoF and Color-Patch-Fisher-Vector

Therefore, we have proved that rapid image recognition and high precision is possible on a smartphone.

6.2 Evaluation on bounding box adjustment

Next, we made an experiments to examine effectiveness of bounding box adjustment. We magnified ground-truth bounding boxes of test images with 25% in terms of bounding box size. In fact, we used only 1912 food images as test images in the 5-fold cross-validation classification experiment, since for some food photos their size are almost the same as the size of attached bounding boxes and they do not includes 25% background regions. We

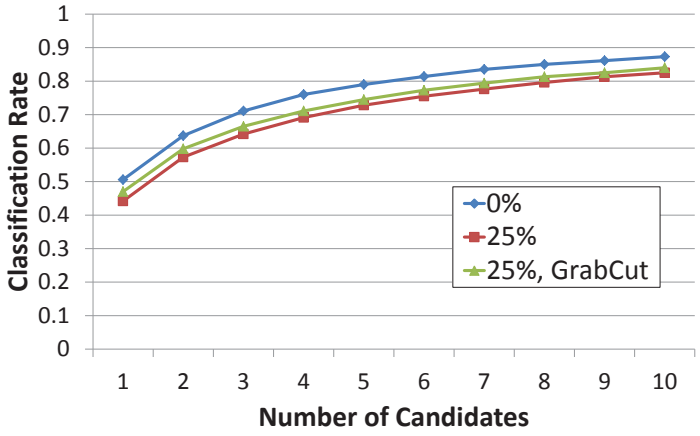

Fig. 11 Classification rates in case of the ground-truth bounding box (BB), 25% magnified bounding box and adjusted bounding box after 25% magnification (shown as "25%, GrabCut").

compared the groundtruth bounding boxes, 25%-magnified bounding boxes, and adjusted bounding boxes after 25% magnification in terms of classification rate under the same condition as the previous experiment. Figure 11 shows the results, which indicates that 25% magnification of groundtruth bounding boxes degraded the classification rate within the top five by 6.3%, while 25% magnification with bounding box adjustment degraded the rate by only 4.5%. From this results, GrubCut-based bounding box adjustments can be regarded as being effective.

6.3 Evaluation on estimation on food direction

Finally, we made an experiment on estimation of the direction of a food window. We evaluated error in the direction estimation in case of sifting the ground-truth bounding boxes by 10, 15, 20, and 25% to each of eight directions around the original boxes. Figure12 shows cumulative classification rates of the estimated direction with different shifts. The rates with less than $\pm 20^\circ$ error and $\pm 40^\circ$ error were 31.81% and 50.34% in case of 15% shift, and are 34.54%, and 54.16% in case of 25% shift. From these results, when the difference between the ground-truth and the given bounding box is small, estimation of the direction of the ground-truth bounding box is more difficult. This is because the difference of SVM scores between them is small in case that the difference in the location of the bounding boxes is small.

6.4 Evaluation of processing time

We measured processing times on the latest smartphone, Samsung Galaxy Note Ⅱ (1.6GHz Quad Core with Android 4.1). We measured recognition time by repeating 20 times and averaging them. The results are shown in Table 2. Processing time for recognition by the first method and by the second method are 0.26 seconds and 0.065 seconds, respectively.

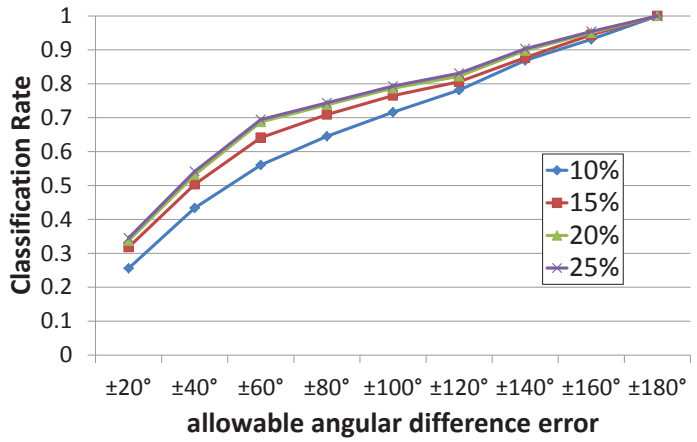

Fig. 12 Cumulative rate of the estimated orientation in case of 10, 15, 20, and 25% shifted bounding boxes.

Table 2 Average processing time.

	average time [sec]
Bounding Box Adjustment	0.70
[Recognition 1] SURF-BoF+Color Histogram	0.26
[Recognition 2] HOG Patch-FV+Color Patch-FV	0.065
Estimation of Food Direction	0.091

Direction estimation takes 0.09 seconds, while bounding box adjustment is relatively high-cost, which takes 0.70 seconds. This is why bounding box adjustment are carried out once after drawn.

Among 0.26 seconds for recognition by the first method and 0.065 seconds for recognition by the second methods, linear SVM classification takes only 0.003 seconds, while most of the time are taken for extraction of image features. From this results, extraction of HOG-FV and Color-FV are much faster than extraction of SURF-BoF and color histogram. In fact, extracting SURF features and voting each feature to one of the 500 codewords to create BoF vectors are most time-consuming processing.

6.5 User Study

We asked five student subjects to evaluate quality of the proposed system in five step evaluation regarding food recognition, how easy to use, quality of direction estimation, and comparison of the proposed system with the baseline which has no food recognition and requires selecting food names from hierarchical menus by touching. The evaluation score 5, 3 and 1 means good, so-so, and bad, respectively. At the same time, we measured time for selecting food items with food recognition, and compared it with the time for selecting food items from the hierarchical menu by hand.

Figure 13 shows the spent time for selecting each food item. The median time were 5.1 second with food recognition, and 5.7 second by hand. This means the proposed system can

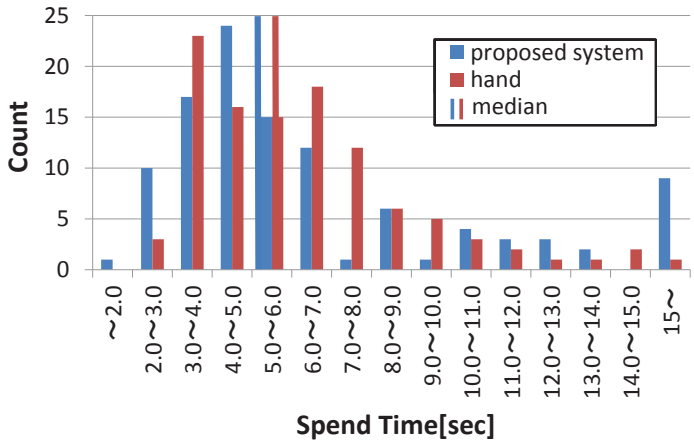

Fig. 13 Time comparison: food recognition vs. hierarchical menu.

Table 3 User study results which are the average of five-step evaluation scores.

Outcome Measure	average score
Recognition quality	3.4
Facility to use	4.2
Quality of Direction Suggestion	2.4
Proposed System Quality (compared with hand-selection)	3.8

help a user select food names faster than from a hierarchical menu by hand. However, for some food items which not able to be recognized, it spent long time to find food names using food recognition.

Table 3 shows the system evaluation by the five grade evaluation. Except for suggest direction, more than three points are obtained. Especially, usability of the system is good, since recognition is carried out in a real-time way. On the other hand, estimation of the expected food region is not evaluated as being effective, since the classification accuracy is not so good for practical use. We will improve it as a future work.

7 Conclusions

We proposed a mobile food image recognition system and two types of food recognition methods. One is the combination of the standard bag-of-features and color histograms with χ^2 kernel feature maps, and the other is a HOG patch descriptor and a color patch descriptor with the state-of-the-art Fisher Vector representation. In both cases, we used a liner SVM as a classifier, which is fast and memory-efficient.

In the experiments, we have achieved the 79.2% classification rate for the top 5 category candidates for a 100-category food dataset with the ground-truth bounding boxes when we used HOG and color patches with the Fisher Vector coding as image features. It is 11.0 point higher than the result by color histogram and SURF-BoF with χ^2 kernel feature maps. In

addition, it is superior to very high cost server side recognition method. Regarding processing time it is only 0.065 second for 100 kinds of targets. It is about four times faster than the processing time by color histogram and SURF-BoF.

As feature works, we plan to extend the system regarding the following issues:

- Touch just a point instead of drawing bounding boxes to specify food regions.
- Use multiple images to improve accuracy of food item recognition.
- Improve accuracy of estimation of expected food regions or move the bounding boxes automatically instead of only showing the direction.
- Take into account additional information such as user's food history, GPS location data and time information.
- Increase the number of food categories to make the system more practical.

Note that Android application of the proposed mobile food recognition system can be downloaded from http://foodcam.mobi/ .

References

Bay et al.(2008). Bay H, Ess A, Tuytelaars T, Van Gool L (2008) Speeded-up robust features (SURF). Computer Vision and Image Understanding 110(3):346–359

Chae et al.(2011). Chae J, Woo I, Kim S, Maciejewski R, Zhu F, Delp E, Boushey C, Ebert D (2011) Volume estimation using food specific shape templates in mobile image-based dietary assessment. In: Proc. of the IS&T/SPIE Conference on Computational Imaging IX, vol 7873, p 78730K

Chatfield et al.(2011). Chatfield K, Lempitsky V, Vedaldi A, Zisserman A (2011) The devil is in the details: an evaluation of recent feature encoding methods. In: Proc. of British Machine Vision Conference

Csurka et al.(2004). Csurka G, Bray C, Dance C, Fan L (2004) Visual categorization with bags of keypoints. In: Proc.of ECCV Workshop on Statistical Learning in Computer Vision (SLCV), pp 59–74

Dalal and Triggs(2005). Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. In: Proc. of IEEE Computer Vision and Pattern Recognition

Deng and Manjunath(2001). Deng Y, Manjunath BS (2001) Unsupervised segmentation of color-texture regions in images and video. IEEE Transactions on Pattern Analysis and Machine Intelligence 23(8):800–810

Fan et al.(2008). Fan RE, Chang KW, Hsieh CJ, Wang XR, Lin CJ (2008) LIBLINEAR: A library for large linear classification. The Journal of Machine Learning Research 9:1871–1874

Felzenszwalb et al.(2010). Felzenszwalb P, Girshick R, McAllester D, Ramanan D (2010) Object detection with discriminatively trained part-based models. IEEE Transactions on Pattern Analysis and Machine Intelligence 32(9):1627–1645

He et al.(2013). He Y, Xu C, Khanna N, Boushey C, Delp E (2013) Food image analysis: Segmentation, identification and weight estimation. In: Proc. of IEEE International Conference on Multimedia and Expo

Jia et al.(2012). Jia D, Alex B, Sanjeev S, Hao S, Aditya K, Fei-Fei L (2012) Imagenet large scale visual recognition challenge 2012 (ILSVRC2012). http://www.image-net.org/challenges/LSVRC/2012/index

Kitamura et al.(2008). Kitamura K, Yamasaki T, Aizawa K (2008) Food log by analyzing food images. In: Proc. of ACM International Conference Multimedia, pp 999–1000

Kitamura et al.(2009). Kitamura K, Yamasaki T, Aizawa K (2009) Foodlog: Capture, analysis and retrieval of personal food images via web. In: Proc. of ACM Multimedia Workshop on Multimedia for Cooking and Eating Activities, pp 23–30

Kumar et al.(2012). Kumar N, Belhumeur P, Biswas A, Jacobs D, Kress W, Lopez I, Soares J (2012) Leafsnap: A computer vision system for automatic plant species identification. In: Proc. of European Conference on Computer Vision

Lampert et al.(2008). Lampert CH, Blaschko MB, Hofmann T (2008) Beyond sliding windows: Object localization by efficient subwindow search. In: Proc. of IEEE Computer Vision and Pattern Recognition

Lazebnik et al.(2006). Lazebnik S, Schmid C, Ponce J (2006) Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: Proc. of IEEE Computer Vision and Pattern Recognition, pp 2169–2178

Lowe(2004). Lowe DG (2004) Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2):91–110

Mariappan et al.(2009). Mariappan A, Bosch M, Zhu F, Boushey C, Kerr D, Ebert D, Delp E (2009) Personal dietary assessment using mobile devices. In: Proc. of the IS&T/SPIE Conference on Computational Imaging VII, vol 7246, pp 72,460Z–1–72,460Z–12

Maruyama et al.(2012). Maruyama T, Kawano Y, Yanai K (2012) Real-time mobile recipe recommendation system using food ingredient recognition. In: Proc. of ACM MM Workshop on Interactive Multimedia on Mobile and Portable Devices(IMMPD)

Matsuda et al.(2012). Matsuda Y, Hoashi H, Yanai K (2012) Recognition of multiple-food images by detecting candidate regions. In: Proc. of IEEE International Conference on Multimedia and Expo

Perronnin and Dance(2007). Perronnin F, Dance C (2007) Fisher kernels on visual vocabularies for image categorization. In: Proc. of IEEE Computer Vision and Pattern Recognition

Perronnin et al.(2010). Perronnin F, Sánchez J, Mensink T (2010) Improving the fisher kernel for large-scale image classification. In: Proc. of European Conference on Computer Vision

Philbin et al.(2008). Philbin J, Chum O, Isard M, Sivic J, Zisserman A (2008) Lost in quantization: Improving particular object retrieval in large scale image databases. In: Proc. of IEEE Computer Vision and Pattern Recognition

Rother et al.(2004). Rother C, Kolmogorov V, Blake A (2004) Grabcut: Interactive foreground extraction using iterated graph cuts. In: ACM SIGGRAPH, pp 309–314

Vedaldi and Zisserman(2012). Vedaldi A, Zisserman A (2012) Efficient additive kernels via explicit feature maps. IEEE Transactions on Pattern Analysis and Machine Intelligence

Wang et al.(2010). Wang J, Yang J, Yu K, Lv F, Huang T, Gong Y (2010) Locality-constrained linear coding for image classification. In: Proc. of IEEE Computer Vision and Pattern Recognition, pp 3360–3367

Yang et al.(2010). Yang S, Chen M, Pomerleau D, Sukthankar R (2010) Food recognition using statistics of pairwise local features. In: Proc. of IEEE Computer Vision and Pattern Recognition

Yu et al.(2011). Yu F, Ji R, Chang S (2011) Active query sensing for mobile location search. In: Proc. of ACM International Conference Multimedia