Multimed Tools Appl
DOI 10.1007/s11042-014-2000-8

FoodCam: A real-time food recognition system on a smartphone

Yoshiyuki Kawano · Keiji Yanai

Abstract We propose a mobile food recognition system, FoodCam, the purposes of which are estimating calorie and nutrition of foods and recording a user's eating habits. In this paper, we propose image recognition methods which are suitable for mobile devices. The proposed method enables real-time food image recognition on a consumer smartphone. This characteristic is completely different from the existing systems which require to send images to an image recognition server. 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 a standalone mobile application for Android smartphones 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.

Y. Kawano · K. Yanai (✉)
The University of Electro-Communications, Tokyo 1-5-1 Chofugaoka, Chofu-shi, Tokyo, 182-8585 Japan
e-mail: yanai@cs.uec.ac.jp

Y. Kawano
e-mail: kawano-y@mm.inf.uec.ac.jp

Published online: 12 April 2014

Keywords Food recognition · Dietary recording · Smartphone application · Fisher vector · Mobile image recognition

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 have explored image recognition methods suitable for mobile devices. In this paper, we propose two kinds of mobile real-time 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 both method, as a classifier, we use linear SVMs in the one-vs-rest manner.

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 [24] as an image classifier.

In the second method, we adopt the more sophisticate method than the first one. Some effective image representations for image classification have been proposed recently, which can boost recognition performance in case of using a linear SVM. Especially, Fisher Vector [20, 21] is known as a high performance method among the recent image representations. It is suitable for a linear classifier, and has been turned out that it can improved recognition accuracy compared to popular combination of bag-of-features (BoF) and a non-linear SVM. Moreover, while BoF needs a larger-size dictionary to improve recognition accuracy, a larger-size dictionary brings increase of computational cost for searching nearest visual codewords from the visual word dictionary. On the other hand, Fisher Vector is able to achieve high recognition accuracy with even a small-size dictionary, and low computational complexity. This property of the Fisher Vector is a big advantage for mobile devices.

Thus, Fisher Vector helps improve recognition accuracy and processing time for a mobile object recognition. However, no mobile image recognition system employing Fisher Vector has existed so far. Then, in this paper, we propose a mobile recognition method employing Fisher Vector which makes effective use of computational resource of a smartphone, and we will show the effectiveness of the proposed method by the experiments.

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 processing 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 a stand-alone 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 [23], and estimation of the direction of the expected food regions based on Efficient Sub-window Search (ESS) [14]. Since this recognition process is performed repeatedly, a user can search for good position of a smartphone camera to recognize foods more accurately by moving it continuously without pushing a camera shutter button.

To summarize our contribution in this paper, it consists of four folds: (1) proposing two food recognition methods suitable for mobile devices in terms of processing time, memory efficiency, and classification accuracy (2) implementing an interactive and real-time food recognition system for a consumer Android-based smartphone by utilizing multiple CPU cores (3) validating the effectiveness of the proposed system by objective and subjective

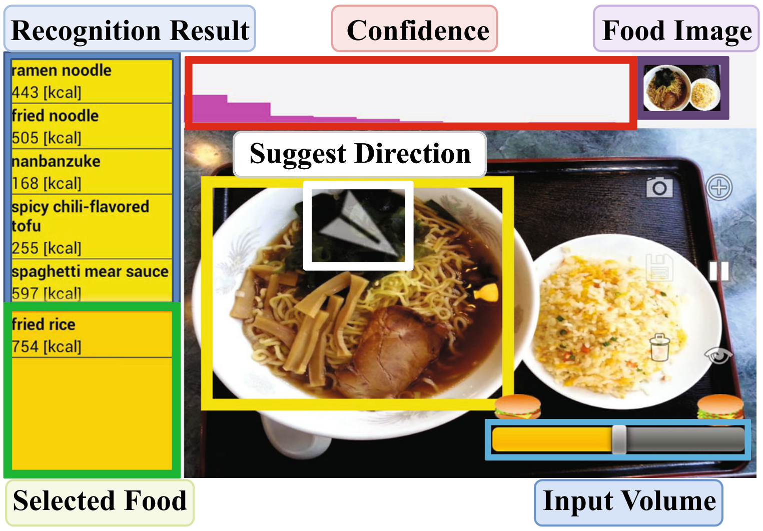


Fig. 1 A screen-shot of the main screen of the proposed system

experiments, and (4) proposing two additional functions: adjustment of the given bounding box, and estimation of the direction to the expected food region.

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. Section 6 describes the experimental results on processing times and classification accuracy 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 very various even within the same category. This is a kind of a fine-grained image categorization problem. Several works on food recognition have been published so far. Yang et al. [26] proposed pairwise local features which exploit positional relation of eight basic food ingredients. Matsuda et al. [19] proposed a method for recognition multiple-food images which detected food regions by several detectors including circle detector, JSEG [6] and Deformable Part Model (DPM) [8], and 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 [11, 12] estimates food balance. It divides the food image into 300 blocks, from each blocks extracts color and DCT coefficients, and classifies to five groups such as staple, main dish, side dish, fruit, and non-food.

The TADA dietary assessment system [17] 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 [2, 9].

In all the above-mentioned systems image recognition processes perform on the server-side, 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 recognizes only food categories and not food volumes, 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

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

recognition applications which recognizes specific objects 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 appearances of which always stays unchanged, which is different from this paper the target of which is generic objects.

Kumar et al. [13] 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. [18] 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 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. [27] 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.

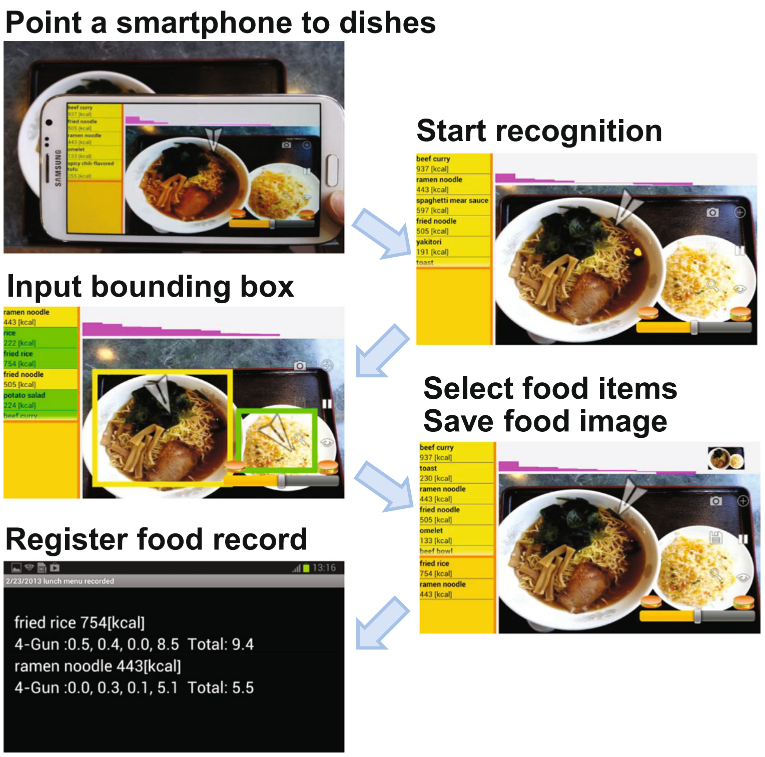
3 System overview

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 propose a mobile system which has function of food image recognition designed for mobile devices. 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 Fig. 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 more reliable region is estimated regarding each given 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 a correct food item name is 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 Fig. 3a. Meal records can be sent to the server, and a user can see them on the Web (Fig. 3b).



2/23/2013 lunch menu recorded

fried rice 754[kcal]
4-Gun :0.5, 0.4, 0.0, 8.5 Total: 9.4
ramen noodle 443[kcal]
4-Gun :0.0, 0.3, 0.1, 5.1 Total: 5.5

Fig. 2 The flow of the typical usage of the proposed mobile application system

4 Methods

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

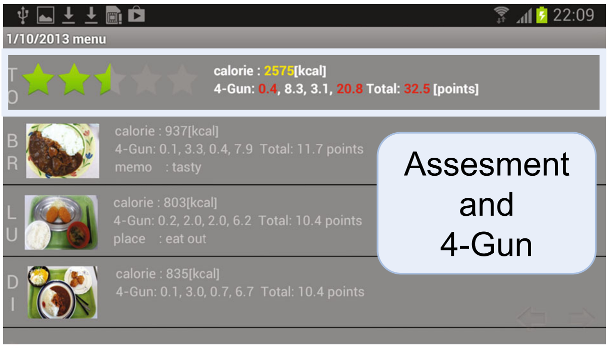
1. Adjustment of bounding boxes.
2. Recognition of food items within the given bounding boxes.
3. Estimation of the direction of the possible food region.

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

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 [23], 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



1/10/2013 menu

T O — calorie : 2575[kcal]
4-Gun: 0.4, 8.3, 3.1, 20.8 Total: 32.5 [points]

B R — calorie : 937[kcal]
4-Gun: 0.1, 3.3, 0.4, 7.9 Total: 11.7 points
memo : tasty

L U — calorie : 803[kcal]
4-Gun: 0.2, 2.0, 2.0, 6.2 Total: 10.4 points
place : eat out

D I — calorie : 835[kcal]
4-Gun: 0.1, 3.0, 0.7, 6.7 Total: 10.4 points

a Meal records on the mobile screen.

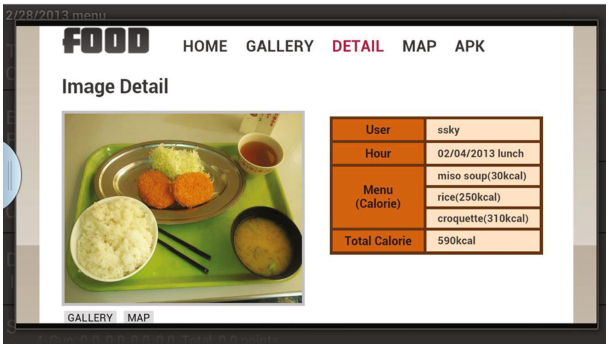


Image Detail

User	ssky
Hour	02/04/2013 lunch
Menu (Calorie)	miso soup(30kcal)
	rice(250kcal)
	croquette(310kcal)
Total Calorie	590kcal

b Meal records shown on the Web.

Fig. 3 Meal records

doubly-extended boxes of the original bounding boxes as background. Since the computational cost of GrabCut is relatively high in the real-time recognition system, 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 achieved the best five SVM scores over 100 categories are shown on the screen as food item candidates for the given bounding boxes.

This recognition is carried out continuously and repeatedly. The five top candidates on the screen is updated every second. If the name of the food item a user try to recognize is not shown in the top five candidates, a user can obtain different candidates by moving smartphone camera slightly to change camera position and direction. This is one of characteristics of real-time interactive recognition system. Note that we intend that food recognition makes it easier for a user to input names of familiar food items within the given bounding boxes rather than teaching names of unknown food items to a user. That is, we expected food recognition makes the user interface of a food-habit recording system easier to use for everyone compared to selecting food item names from 100 candidates only by hand without food recognition.

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 in the following subsections, and compare them in terms of processing time and classification accuracy in Section 6.

4.2.1 Image feature: bag-of-features and color histogram

As the first image features, we adopt bag-of-features [4] 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 [19]. Since we aims for implementing a mobile recognition system which can run in the real-time way, the number of kinds of image features to be extracted should be minimum. 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) [5], Pyramid HoG, and Bag-of-features with SURF features [1]. 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 an 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 [1] 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 [22] 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 linear SVM, we use explicit embedding technique. In this work, we adopt kernel feature maps. Vedaldi et al. [24] 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 [24]. 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. [5]. 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 [16] and SURF [1]. Since HOG description is very simple, it is able to describe local patches 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. 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 to each HOG patch to reduce its dimension from 32 to 24.

Color patch We use the mean and the variance of RGB values 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. Note that applying PCA to local feature vectors before encoding them to Fisher vector is crucial to obtain good classification performance [21]. This is the reason why we apply PCA without dimension reduction.

Fisher vector Recently, Fisher Vector [20, 21] becomes known as a high performance method to represent a set of local features. It can decrease quantization error than bag-of-features [4] 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) [3, 25]. In fact, most of the higher rank teams in the large-scale visual recognition challenge (LSVRC) used Fisher Vector [10].

Moreover, while BoF needs a larger dictionary of visual words to improve recognition accuracy, a 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 a small dictionary, and low computational complexity. This is a prominent advantage for mobile devices.

For the number of local descriptors T, given a set of local descriptors $X = \{x_t\}, t = 1, ..., T$, Fisher Vector is defined as follows:

$$\mathcal{G}_\theta^X = \frac{L_\theta}{T} \nabla_\theta \log p(X|\theta) \tag{2}$$

where $p(X|\theta)$ is a probability density function (pdf), $\nabla_\theta \log p(X|\theta)$ is a gradient of log likelihood, F_θ is the Fisher information matrix, and F_θ is decomposed into $F_\theta^{-1} = L'_\theta L_\theta$.

Fisher Kernel $K(X, Y) = G_\theta^{X'} F_\theta^{-1} G_\theta^Y$ is an inner product of Fisher Vector. Thus it is able to classify images efficiently with a linear classifier.

According to [21], we encode local descriptors into Fisher Vector. We use a probability density functions as Gaussian mixture model (GMM). Then, pdf is given as follows:

$$p(x|\theta) = \sum_{i=1}^{K} \pi_i \mathcal{N}(x|\mu_i, \Sigma_i) \tag{3}$$

where x is a local descriptor, K is a number of Gaussian components, $\theta = \{\pi_i, \mu_i, \Sigma_i\}, i = 1, ..., K$ is a parameter of GMM. π_i is the mixing coefficient, μ_i is 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 in a variance vector σ^2.

The probability that x_t is belong to the i-th component (estimated posterior probability) is given as follows:

$$\gamma_t(i) = \frac{\pi_i \mathcal{N}(x_t|\mu_i, \Sigma_i)}{\sum_{j=1}^{N} \pi_j \mathcal{N}(x_t|\mu_j, \Sigma_j)} \tag{4}$$

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

$$\mathcal{G}_{\mu,i}^X = \frac{1}{T\sqrt{\pi_i}} \sum_{t=1}^{T} \gamma_t(i) \left(\frac{x_t - \mu_i}{\sigma_i} \right) \tag{5}$$

$$\mathcal{G}_{\sigma,i}^X = \frac{1}{T\sqrt{2\pi_i}} \sum_{t=1}^{T} \gamma_t(i) \left[\frac{(x_t - \mu_i)^2}{\sigma_i^2} - 1 \right] \tag{6}$$

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

In this paper, the number of components 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 [21].

4.2.3 Training and classification

As a classifier, we use a linear kernel SVM, and we adopt the one-vs-rest strategy for multi-class classification.

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:

$$\begin{aligned} f(\mathbf{x}) &= \sum_{i=1}^{M} y_i \alpha_i K(\mathbf{x}, \mathbf{x_i}) + b = \sum_{i=1}^{M} y_i \alpha_i \langle \mathbf{x}, \mathbf{x_i} \rangle + b \\ &= \left\langle \sum_{i=1}^{M} y_i \alpha_i \mathbf{x_i}, \mathbf{x} \right\rangle + b = \langle \mathbf{w}, \mathbf{x} \rangle + b \end{aligned} \tag{7}$$

where $\mathbf{x}$ is an input vector, $f(\mathbf{x})$ is an output SVM score, $\mathbf{x_i}$ is a 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 value, 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.

To implement a mobile system, we prepared one hundred categories food image dataset which has more than 100 images per category, and all the food items in which are marked with bounding boxes. The total number of food images in the dataset is 12,905. The detail will be explained in Section 6. With this dataset, we train 100 linear SVMs with LIBLINEAR [7] for 100-class multi-class classification in the one-vs-rest manner in off-line. After extracting gradient-based features such as SURF-BoF and HOG-FV and color-based feature such as color histogram and Color-FV from each window image surrounded by the given bounding boxes, we train each SVM using all the image features in one certain category as positive samples and all the rest image features as negative samples. Note that we train SVMs for gradient-based features and SVMs for color-based features independently, because we adopt the late fusion strategy to fuse two different kinds of features. In the late fusion, the SVM output value of gradient-based features $f_{gradient}(\mathbf{x})$ and the SVM value of color-based features $f_{color}(\mathbf{x})$ are combined by linear weighting to obtain the combined score $f_{combined}(\mathbf{x})$ as follows:

$$f_{combined}(\mathbf{x}) = \lambda f_{gradient}(\mathbf{x}) + (1 - \lambda) f_{color}(\mathbf{x}) \tag{8}$$

The weight λ to combine two SVM scores is estimated by cross-validation when training. In our system, the names of the food categories the SVM scores of which are ranked within the top five among 100 categories are shown as food candidates on the screen.

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 Fig. 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) [14], which can be directly applied for a combination of a linear 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 (7) can be decomposed into a vector $\mathbf{w}^+$ including positive elements and a vector $\mathbf{w}^-$ including negative elements.

$$\mathbf{w} = \mathbf{w}^+ + \mathbf{w}^- \tag{9}$$

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 [14]. 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 Fig. 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.

5.1 The first version system

In the first version system, as shown in Fig. 4, firstly the bounding box adjustments employing GrabCut are carried out in two cores soon after a bounding box is drawn by a user on the screen, In this first step, we parallelize estimation of two Gaussian Mixture Models (GMMs) for foreground pixels and background pixels, and evaluation of probability that pixels belong to foreground or background, respectively. Regarding graph cut, it is calculated on a single core. GrabCut processing is relatively high, bounding box adjustment is carried out only once for one bounding box. After the adjustment was finished, the loop of

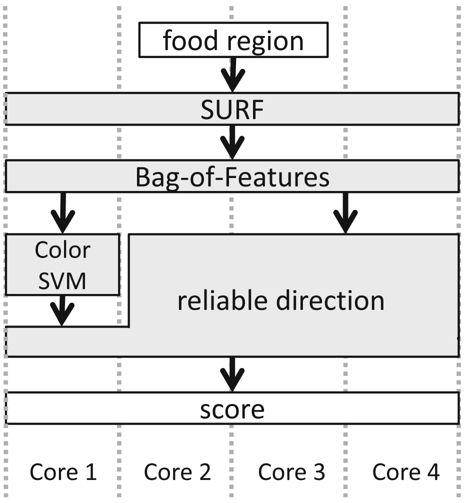


Fig. 4 Processing flow and assignment on CPU cores for the system using bag-of-features and color histogram

feature extraction and score calculation is started and it is continuously repeated once per second.

After computation of GrabCut, secondly SURF features are extracted in the dense sampling way on four cores. The given image is divided into 2x2 blocks, and SURF features are calculated on each block in parallel.

Thirdly, in the step of generation of BoF vectors, regarding each block, BoF vectors are built on each CPU core independently. After that, all the BoF vectors are summed in a single core, and then L1-normalization and fast χ^2 feature maps are calculated.

Fourthly, extraction of color histogram and SVM evaluation on both BoF and color vectors are carried out on a single core, while estimation of the direction of more reliable region on the rest three cores. Since color histogram extraction and SVM evaluation are very light computation, it must be done before reliable direction estimation is completed. Then, the core used for color histogram calculation helps 10 % of processing of reliable direction estimation, while each of three cores processes 30 % of total processing of reliable direction estimation. Because basically SURF extraction, BoF generation and reliable direction estimation are computationally costly, they are parallelized in the first version system.

5.2 The second version system

In the second system, as shown in Fig. 5, at first bounding box adjustment are carried out on two cores. This is the same as the first version system. Bounding box adjustment is carried out only once for one drawn bounding box.

After bounding box adjustment, the processing on HOG patch and the processing on color patch from local feature extraction to SVM score evaluation are carried out in parallel by assigning two CPU cores to each processing, since computational cost of both features are almost the same. After each step of local patch extraction and Fisher vector encoding, synchronization is carried out within each of two core pairs. After SVM evaluation, all the obtained SVM scores are integrated and sorted in the descending order in a single core.

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.

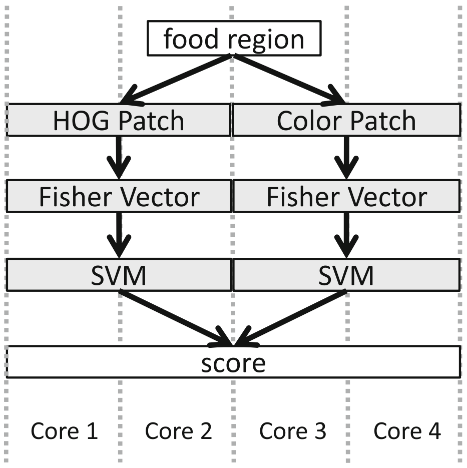


Fig. 5 Processing flow and assignment on CPU cores for the system using a HOG patch and a Color patch with Fisher Vector

For the second system, we devised speeding-up of computation by pre-computation. The gradient of Fisher Vector with respect to the mean (5) is transformed 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 (10) for each component, so effectively.

$$\mathcal{G}^X_{\mu,i} = \frac{1}{\sqrt{\pi_i}\sigma_i}\frac{1}{T}\sum_{t=1}^{T}\gamma_t(i)(x_t - \mu_i) \tag{10}$$

Moreover, we computed the term of calculate posterior probability by GMM and the gradient with respect to the mean and sigma in off-line in advance, 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 (6) and $1/\sqrt{\pi_i}\sigma_i$ of (10)).

While we trained SVMs in off-line on a PC, in the on-line mode all the processing steps are carried out on a smartphone. In on-line, all the parameter values used in recognition steps are pre-loaded on main memory (eigenvalue and eigenvector for PCA, created lookup table, mean of GMM, weight vectors of SVMs). When using Fisher Vector, we need to set the relatively smaller value to the number of codewords so that all the values can be stored on main memory in advance. However, 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 Fisher Vector is also superior to BoF in terms of memory space efficiency as well as classification accuracy and computational cost.

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 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, respectively. 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 processing time and recognition accuracy for food recognition, and evaluation on bounding box adjustment and reliable food direction. 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 category, and all the food items 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. This 100-class food image dataset is called "UEC-FOOD100" which can be downloaded from http://foodcam.mobi/dataset/.

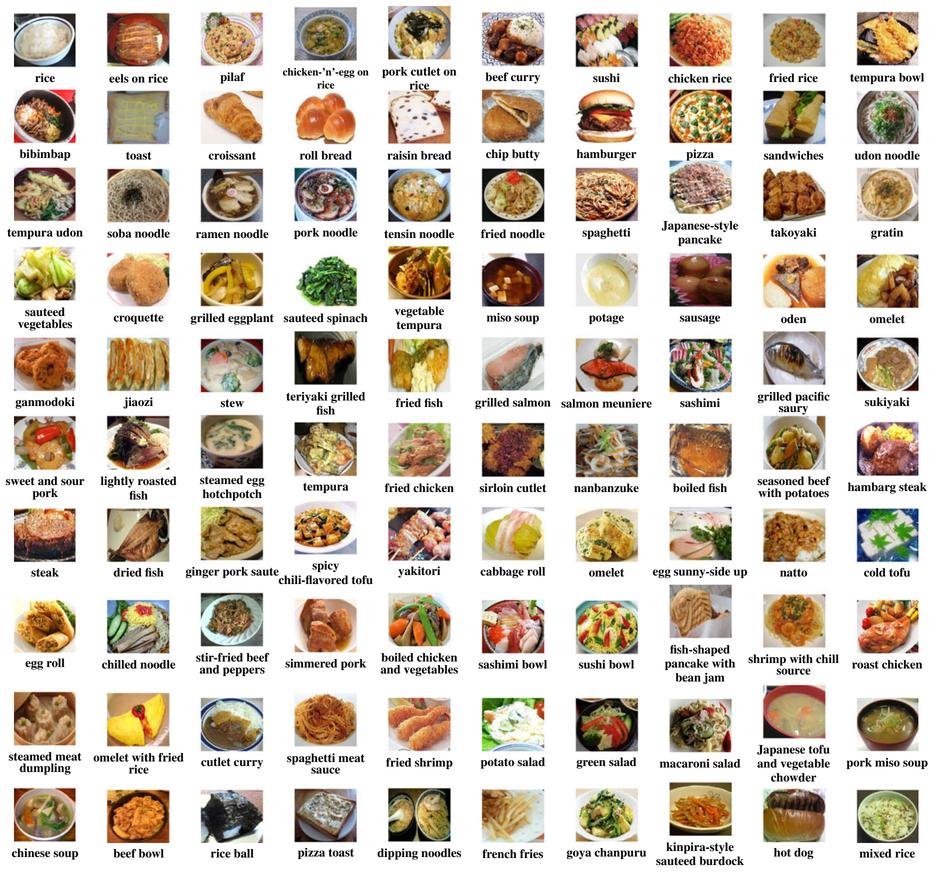


Fig. 6 100 kinds of food images in the "UEC-FOOD100" dataset

In the experiments on classification accuracy evaluation, 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 of processing time

We measured processing times on the latest smartphone, Samsung Galaxy Note Ⅱ (1.6GHz Quad Core with Android 4.1). In this subsection, we desribe the processing time on each step in the first version and the second version system, the time in case of changing the number of active CPU cores and comparison with other baseline methods.

6.1.1 Processing time on each step

We measured recognition time by repeating 20 times and averaging them. The results are shown in Table 1. Processing time for recognition by the first method and by the second

Table 1 Average processing time

	Average time [second]
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

Table 2 Processing time with different numbers of CPU cores

	time [second]
Quad-core	0.065
Dual-core	0.090
Single-core	0.15

method are 0.26 seconds and 0.065 seconds, respectively. 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.

6.1.2 Processing time in case of changing the number of CPU cores

We evaluated response times in case of using a single core and two cores for the second systems. Note that implementation with three cores is impossible since we used two threads for each of two kinds of the features. The results are shown in Table 2.

In case of dual core, we assigned each of two cores to each of two kinds of features, HoG-FV and Color-FV. Compared to single-core, the processing time was reduced by 40.0 % in case of dual-core and by 27.8 % in case of quad-core. For these observation, the quad-core implementation in our system is very effective for speeding-up of the processing time.

6.1.3 Comparison with baseline methods

We evaluated the processing time of the baseline methods used in Section 6.2 (evaluation on classification accuracy) on a PC (Intel Xeon X3230 2.66GHz). Firstly, we evaluated the time to extract local features, secondly evaluated the time for coding, and finally evaluated the time for evaluation of 100-class one-vs-rest SVMs.

Extraction of local features As local features, we compared HOG Patch with two different implementation of SIFT (OpenCV[2] and VL_FEAT[3]), SURF, and Color Histogram. To

[2]http://www.opencv.org/

[3]http://www.vlfeat.org/

Table 3 Processing time for local feature extraction

	time [second]
SIFT (OpenCV)	0.76
SIFT (VL_FEAT)	0.28
SURF (OpenCV)	0.069
HOG patch	0.0055
Color patch	0.0060
Color histogram	0.00012

extract local features, we sampled keypoints in two scales every eight pixels in the grid sampling manner. As a result, the number of keypoints extracted from one image is about seven hundreds. Table 3 shows the results.

Although SIFT(VL_FEAT) is a fast implementation which is optimized for grid sampling, SURF is much faster than SIFT(VL_FEAT). HOG patch is about twelve times faster than SURF. Although HOG patch is not invariant for scale and direction, its classification accuracy is equivalent or outperforms to one of SURF. This means HOG patch is very suitable for mobile recognition. In addition, Color patch is equivalent to HOG patch in terms of processing time. Although normal color histogram is much faster, as shown in the next section regarding the accuracy the combination of color patch and Fisher vector is much superior to color histogram.

Encoding of BoF or FV We compared the time to encode feature vectors with bag-of-features (BoF) with/without χ^2 feature mapping and Fisher Vectors (FV). The dimension of both features are the same as mentioned in Section 4.2. The size of the codebook is 500, and the dimension of feature mapped BoF is 1500 since feature maps make the dimension of the original vector three times. On the other hand, regarding Fisher vectors, the number of GMM is 32, and the dimension of HOG patch is reduced to 24-dim by PCA. Total dimension becomes 1536. As a result, the dimension of both feature vectors are almost the same to each other. Table 4 shows the results.

Regarding BoF and chi2 feature map, the cost to search nearest visual words is relatively high, although feature mapping is very fast. Fisher vector is 1.8 times faster than BoF. Therefore, it has turned out that FV is more effective not only for classification accuracy but also for computational speed compared to BoF when the dimension of both feature vectors are equivalent. This means FV is more suitable for fast recognition. Note that the

Table 4 Processing time of Bag-of-features and Fisher vector encoding

	Time [second]
BoF	0.018
BoF + chi2 map	0.018
(only chi2 map)	(0.000024)
FV (including PCA)	0.0099
(only FV)	(0.0040)
(only PCA)	(0.0059)

Table 5 Processing time of evaluation of linear and non-linear SVM

	Time [second]
Linear SVM	0.00033
Non-linear SVM	0.12

processing time of FV (0.0099 sec.) includes the time to apply PCA to all the HOG patch vectors (0.0059 sec.). In fact, the time to calculate a Fisher vector is shorter than the time to apply PCA.

Evaluation of linear or non-linear SVMs Finally, we compare the processing time to evaluate linear and non-linear SVMs. We examined the processing time to evaluate a linear SVM with SIFT-FV (1532 dim) and a non-linear SVM with RBF kernel and SIFT-BoF (500dim). The results are shown in Table 5.

Note that the time shown in Table 5 represents the time to evaluate one SVM. In case of 100-class classification in one-vs-rest manner, evaluating 100 SVMs is required. Even if SVM evaluation is parallelized ideally over four CPU cores, 25 times as much as the time shown in the table is needed for 100-class food recognition.

Because evaluation of linear SVM needs only calculation of inner product, it is very fast and low memory requirement. On the other hand, since evaluation of non-linear SVM needs kernel computation between all the support vectors and the given image feature, it is very time-consuming. In addition, all the support vectors are needed for evaluation, it required much more memory than linear SVM. In the experiments, since the average number of support vectors over 100 SVMs was 456, 456 times as much memory area as linear SVM is needed for non-linear SVM. Therefore, using non-linear SVM on mobile devices is impractical. Especially, in case of 100 classes, it is impossible to put the all the support vectors on memory, because one application in Android smartphones can use 256 MBytes memory at most.

6.2 Evaluation on classification accuracy

In this experiments, we compare two types of the proposed systems with server-side recognition system by Matsuda et al. [19].

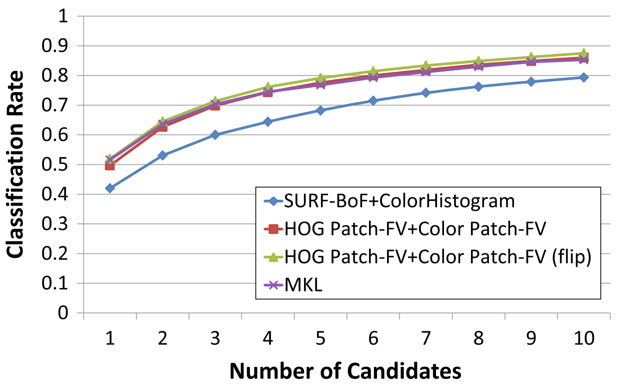


Fig. 7 Classification rate by the two proposed methods and the server-side method [19]

Table 6 Classification rate within the top-1 and top-5 candidates by the proposed methods, the server-side method proposed by [19], and the extended version of the proposed method

Method	top-1	top-5
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[19]	51.6	76.8
Extended HOG Patch-FV+Color Patch-FV(flip)	59.6	82.9

Figure 7 shows classification rate of each recognition method and Table 6 shows classification rate within top-1 and top-5. SURF-BoF+Color Hist. achieved 42.0 and 68.2 % within the top-1 and the top-5 candidates, 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 top-1 and top-5. 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 [19] 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 6 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 [15] was applied. Although the dimension of image features, processing time and memory requirements increased greatly, the classification rate was improved 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 evaluation, 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

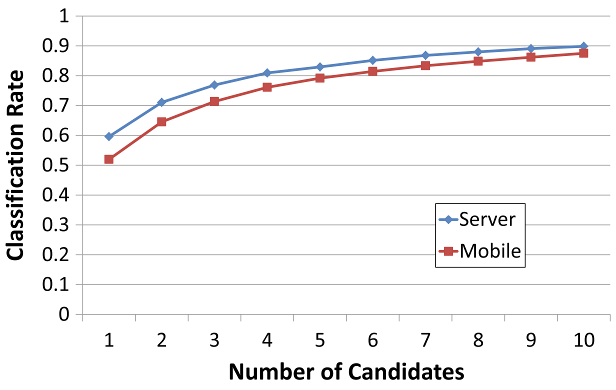


Fig. 8 Classification rate of the mobile version and the extended server version of the second proposed method (HOG and Color patch with Fisher Vector)

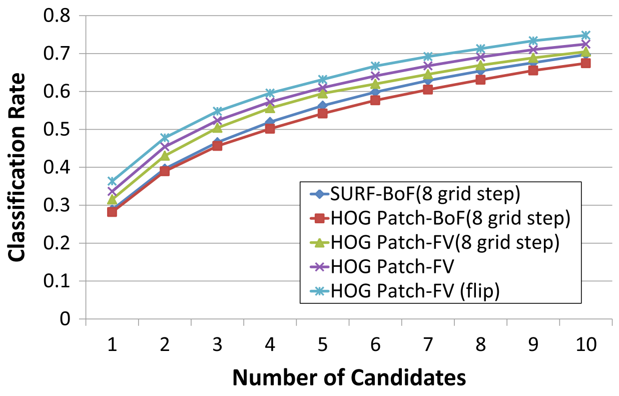


Fig. 9 Classification rate by SURF-BoF, HOG-Patch-BoF and HOG-Patch-Fisher-Vector with 6 or 8 grid step

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 on top-1 and top-5 classification rate between SURF-BoF and HOG Patch-BoF are only 0.62 and 2.1 %, respectively, which means that SURF is slight better. However, HOG Patch-FV achieved higher performance than SURF-BoF, the difference are 2.7 and 3.22 %. In addition, HOG Patch extraction is very fast than SURF extraction. In this experiment, we change a step of dense grid sampling as every 6 pixels to sample points more densely. 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 top-1 and top-5 candidates. The classification rate are 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

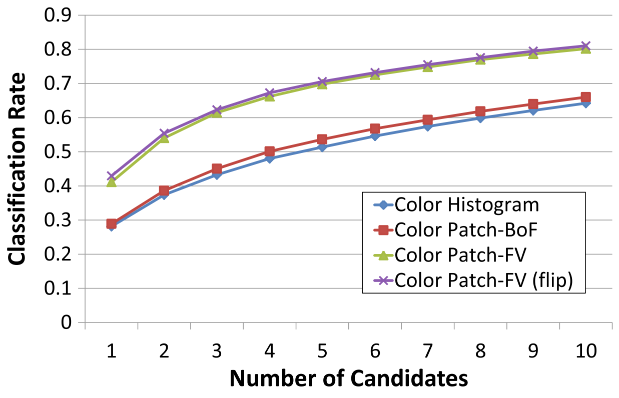


Fig. 10 Classification rate by Color Histogram, Color-Patch-BoF and Color-Patch-Fisher-Vector

histogram. The difference on top-1 and top-5 classification rate between Color Patch-BoF and color histogram are 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 are higher than color histogram by 13.0 and 18.4 %, respectively. 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 the effectiveness of our proposed method. Moreover, we achieved better results than the server-side result which is obtained by very high cost recognition method [19]. Therefore, we showed that rapid image recognition and high precision is possible on a smartphone.

6.3 Evaluation on bounding box adjustment

Next, we made an experiment 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 % backgound regions. We compared 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.

6.4 Evaluation on estimation on food direction

Finally, we made an experiment on estimation of the more reliable direction of a food window. We evaluated error in the direction estimation in case of sifting the ground-truth

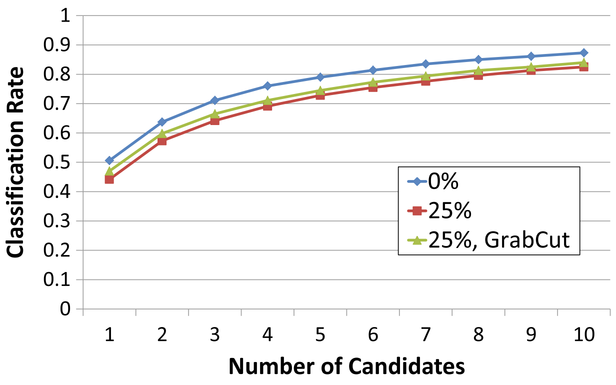


Fig. 11 Classification rates in case of the ground-truth bounding box (BB), 25 % magnified bounding box and adjusted bounding box after 25 % magnification (shown as “25 %, GrabCut”)

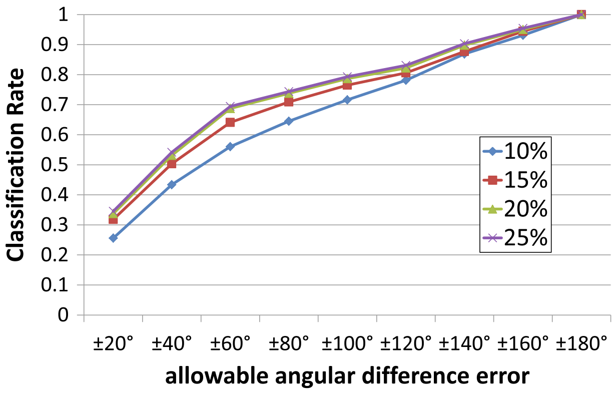


Fig. 12 Cumulative rate of the estimated orientation in case of 10, 15, 20, and 25 % shifted bounding boxes

bounding boxes by 10, 15, 20, and 25 % to each of eight directions around the original boxes. Figure 12 shows cumulative classification rates of the estimated direction with different shifts. The rates with less than ±20° error and ±40° error were 31.81 and 50.34 % in case of 15 % shift, and are 34.54 %, and 54.16 % in case of 25 % shift, respectively. 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.

6.5 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

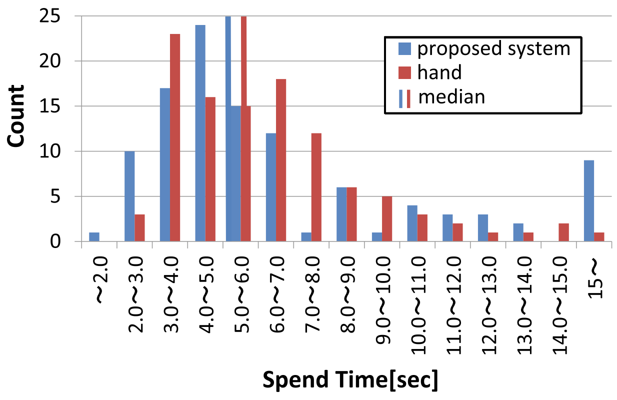


Fig. 13 Time comparison: food recognition vs. hierarchical menu

Table 7 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

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 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 7 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 linear SVM as a classifier, which is fast and memory-efficient.

Especially, in this paper, we have turned out that the combination of a HOG patch, a color patch and a linear SVM is very effective for mobile image classification regarding processing speed and memory efficiency as well as classification accuracy.

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, or automatically detect food regions without any user operation.
– 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.
- Introduce on-line learning on a smartphone for users to be able to add new training images or new food categories by themselves.

Note that Android application of the proposed mobile food recognition system and 100-class food dataset "UEC-FOOD100" can be downloaded from http://foodcam.mobi/.

References

1. Bay H, Ess A, Tuytelaars T, Van Gool L (2008) Speeded-up robust features (SURF). Comput Vis Image Underst 110(3):346–359
2. 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: Proceedings of the IS&T/SPIE conference on computational imaging IX
3. Chatfield K, Lempitsky V, Vedaldi A, Zisserman A (2011) The devil is in the details: an evaluation of recent feature encoding methods. In: Proceedings of British machine vision conference
4. Csurka G, Bray C, Dance C, Fan L (2004) Visual categorization with bags of keypoints. In: Proceedings of ECCV workshop on statistical learning in computer vision (SLCV), pp 59–74
5. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. In: Proceedings of IEEE computer vision and pattern recognition
6. Deng Y, Manjunath BS (2001) Unsupervised segmentation of color-texture regions in images and video. IEEE Trans Pattern Anal Mach Intell 23(8):800–810
7. Fan RE, Chang KW, Hsieh CJ, Wang XR, Lin CJ (2008) LIBLINEAR: a library for large linear classification. J Mach Learn Res 9:1871–1874
8. Felzenszwalb P, Girshick R, McAllester D, Ramanan D (2010) Object detection with discriminatively trained part-based models. IEEE Trans Pattern Anal Mach Intell 32(9):1627–1645
9. He Y, Xu C, Khanna N, Boushey C, Delp E (2013) Food image analysis: segmentation identification and weight estimation. In: Proceedings of IEEE international conference on multimedia and expo
10. 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/
11. Kitamura K, Yamasaki T, Aizawa K (2008) Food log by analyzing food images. In: Proceedings of ACM international conference multimedia, pp 999–1000
12. Kitamura K, Yamasaki T, Aizawa K (2009) Foodlog: capture, analysis and retrieval of personal food images via web. In: Proceedings of ACM multimedia workshop on multimedia for cooking and eating activities, pp 23–30
13. 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: Proceedings of European conference on computer vision
14. Lampert CH, Blaschko MB, Hofmann T (2008) Beyond sliding windows: object localization by efficient subwindow search. In: Proceedings of IEEE computer vision and pattern recognition
15. Lazebnik S, Schmid C, Ponce J (2006) Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In: Proceedings of IEEE computer vision and pattern recognition, pp 2169–2178
16. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vis 60(2):91–110
17. Mariappan A, Bosch M, Zhu F, Boushey C, Kerr D, Ebert D, Delp E (2009) Personal dietary assessment using mobile devices. In: Proceedings of the IS&T/SPIE conference on computational imaging VII
18. Maruyama T, Kawano Y, Yanai K (2012) Real-time mobile recipe recommendation system using food ingredient recognition. In: Proceedings of ACM MM workshop on interactive multimedia on mobile and portable devices(IMMPD)
19. Matsuda Y, Hoashi H, Yanai K (2012) Recognition of multiple-food images by detecting candidate regions. In: Proceedings of IEEE international conference on multimedia and expo
20. Perronnin F, Dance C (2007) Fisher kernels on visual vocabularies for image categorization. In: Proceedings of IEEE computer vision and pattern recognition

21. Perronnin F, Sánchez J, Mensink T (2010) Improving the fisher kernel for large-scale image classification. In: Proceedings of European conference on computer vision
22. Philbin J, Chum O, Isard M, Sivic J, Zisserman A (2008) Lost in quantization: improving particular object retrieval in large scale image databases. In: Proceedings of IEEE computer vision and pattern recognition
23. Rother C, Kolmogorov V, Blake A (2004) Grabcut: interactive foreground extraction using iterated graph cuts. In: ACM SIGGRAPH, pp 309–314
24. Vedaldi A, Zisserman A (2012) Efficient additive kernels via explicit feature maps. IEEE Trans Pattern Anal Mach Intell
25. Wang J, Yang J, Yu K, Lv F, Huang T, Gong Y (2010) Locality-constrained linear coding for image classification. In: Proceedings of IEEE computer vision and pattern recognition, pp 3360–3367
26. Yang S, Chen M, Pomerleau D, Sukthankar R (2010) Food recognition using statistics of pairwise local features. In: Proceedings of IEEE computer vision and pattern recognition
27. Yu F, Ji R, Chang S (2011) Active query sensing for mobile location search. In: Proceedings of ACM international conference multimedia

Yoshiyuki Kawano received B.Eng from the University of Electro-Communications, Tokyo in 2013. Currently, he is a master-course student of the graduate school of the University of Electro-Communications, Tokyo. His research interest includes mobile image recognition and food recognition.

Keiji Yanai received B.Eng., M.Eng. and D.Eng degrees from the University of Tokyo in 1995, 1997 and 2003, respectively. From 1997 to 2006, he was a research associate at Department of Computer Science, the University of Electro-Communications, Tokyo. He is currently an associate professor at Department of Informatics, the University of Electro-Communications, Tokyo. His recent research interests include object recognition and Web multimedia processing. He is a member of IEEE Computer Society, ACM, JSAI and IPSJ.