*Journal of Information Hiding and Multimedia Signal Processing* ©2016 ISSN 2073-4212
**Ubiquitous International** Volume **7**, Number **2**, March **2016**

# Extremely Randomized Clustering Forest Based Scene Recognition Algorithm in Mobile Devices

Xiangdong Yin

School of Electronics and Information Engineering
Hunan University of Science and Engineering
Yongzhou City, Hunan Province, 425100, China
yinxiangdongxz@163.com

Linping Tang

College of Mathematics and Econometrics
Hunan University
Changsha City, Hunan Province, China
274405911@qq.com

Na Wang

College of Media
Hunan University of Science and Engineering
Yongzhou City, Hunan Province, 425100, China
164492803@qq.com

Received October, 2015; revised November, 2015

Abstract. *With the development of embedded systems and image processing techniques, more and more users apply mobile devices to take photos. In order to recognize scenes with these mobile photos, we proposed an extremely randomized clustering forest based scene recognition algorithm. Firstly, we introduced a gravity-aware feature in traditional SIFT feature descriptor. After extracting feature descriptors for all images, we used extremely randomized clustering trees to classify feature descriptors into categories. While generating trees, we proposed an improved leaf nodes spliting algorithm. Finally, we used support vector classifier and geometric calibration for supervised learning. The experiments show that, our proposed method is not only more efficient but also more effective than the SIFT and SURF descriptors.*
**Keywords:** Mobile computing, Scene recognition, Extremely randomized clustering forest, Support vector machine.

1. **Introduction.** Recently, with the development of hardware and mobile computing techniques, the price of mobile devices, e.g. smart phones, decreases sharply. More and more users take images (photos) with their mobile devices, and upload these images to Internet according to the wireless communication techniques. There are a lot of image sharing Websites, such as Flickr and Facebook, and each of them has a huge image resource database. Therefore, how to recognize objects with these images efficiently and effectively is an important research topic [1, 2]. However, the images photographed by mobile devices are easily affected by lighting, scale, visual angle, noises of background, occlusion and so on, and this poses a huge challenge for recognizing objects in mobile environment [3].

There are two kinds of image recognizing methods for mobile devices. The first method is based on the idea of sensors and Client/Server [2]. Here, some simple image processing processes, such as image capturing [4], feature description [5, 6], feature extraction [7, 8] and location [9], are processed on mobile devices with all kinds of sensors, and complex processing processes, that require massive computation and memory, are executed on a server. This kind of method is easily affected by wireless communication [10], and has very poor performance when the wireless network congests. The other method decreases the complexity of computation and storage of feature descriptors based on the idea of compression [11, 12]. Here, every image is described as a feature vector, and the Hamming distance is applied to compute the similarity between two feature vectors. With the increase of compression ratio, the recognition precision of this method decreases, and thus some auxiliary algorithms, such as space and geometry consistencies, are required to improve the precision.

As mobile devices are usually integrated with gravimeter and/or gravimeter, in this paper we introduced a gravity-aware feature into traditional SIFT feature descriptor, and proposed an extremely randomized clustering forest based scene recognition algorithm with the gravity-aware feature descriptors.

The rest of the paper is organized as follows. In section 2, we reviewed related works about feature descriptors in mobile devices. In section 3, we presented our extremely randomized clustering forest based scene recognition algorithm. Experiments and conclusion are given in sections 4 and 5 respectively.

## 2. **Related works.**

In this section, we will review related works about scene or object recognition in image processing. Object recognition can be divided into feature description, feature extraction and feature matching.

### 2.1. **Feature description.**

The purpose of feature description is to represent an image as a series of features, which are invariant to rotation, scaling, lighting, transformation and so on. There are several local feature description methods, such as scale invariant feature transform (SIFT) [13], speed up robust features (SURF) [14] and binary robust independent elementary features (BRIEF) [15]. The SIFT descriptor is invariant to many image operations, e.g. rotation, scaling, lighting and transformation, but it needs longer execution time and more memory. The SURF descriptor has good performances on both time and space usages, and is also robust to many image transformations, but the object recognition rate is lower than the BRIEF descriptor. In addition, both efficiency and object recognition rate of the BRIEF descriptor are better than SIFT and SURF. However, all of these feature description methods needs massive computing and storage resources, and thus cannot be applicable to mobile devices.

### 2.2. **Feature extraction.**

Feature extraction (or called feature detection) exacts feature pattern from massive images. Mair et al. [16] proposed an adaptive and generic accelerated segment test algorithm, short for AGAST, for detecting multi-scale corners in images. AGAST detects corners by building pixel-similar binary decision tree, and is repeatable for detecting corners while images are given. Leutenegger et al. [17] proposed a binary robust invariant scalable keypoints feature detecting algorithm, short for BRISK, and this algorithm detected changes of grayscale between neighbour regions in a continuous space, and let the obvious changes of grayscale be the local features. Rosten and Drummond [18] proposed the features from accelerated segment test, short for FAST. FAST ascertains feature points via the lighting relationships between the center of a cycle with its neighbour points, and has the advantages of quicker detecting speed and lower memory usage, so it is applicable to many practical situations.

2.3. **Feature matching.** Classical feature matching methods include brutal force algorithm, approximating nearest neighbours by clustering images, e.g. KD-Tree [19] and K-means [20], and Bayesian posterior based methods, e.g. R-Tree [21] and Ferns [22]. Both R-Tree [21] and Ferns [22] classify feature points by maximizing the posterior of training feature points. In order to build a more accurate decision tree, R-Tree and Ferns need massive training samples, and consume lots of memory while training. They can only be applicable to single scene recognition, and cannot be applicable to object recognition for multiple scenes in mobile devices because of the constraints of computing ability and storage. Both KD-Tree [19] and K-means [20] build decision trees via clustering methods. According to clustering, they can reduce the number of comparison between samples, need less training samples, and thus is much more applicable to mobile devices.

2.4. **Algorithms on mobile devices.** Object recognition algorithms on mobile devices are usually based on the above algorithms. Wagner et al. [23] studied the recognition and tracking for natural scenes and objects on smart phones by revising the SIFT and Ferns algorithms. Although their algorithm is real time, but only supports several fixed scenes. Chen et al. [24] implemented a SURF based system for matching images on smart phones. Their system is simple, and can be applicable to many scenes, but the matching speed is very low for high resolution images. Gu et al. [25] developed a mobile application of augmented reality based on J2ME. This application can be used to recognize scene in real time and augment 3-dimension display of overlapped objects, but the disadvantage is that, it is based on markers and depends on natural environment.

3. **Extremely randomized clustering forest object recognition algorithm.** In this section, we introduce how to recognize objects in mobile devices with extremely randomized clustering forests. We first build feature descriptors for images with the gravity aligned feature descriptor, then propose an extremely randomized algorithm for clustering feature descriptors of objects, and finally apply support vector classifier and geometric calibration to learn parameters of model.

3.1. **Building feature descriptor.** Kurz et al. [26] proposed the gravity aligned feature descriptor, and this kind of descriptor is an improved feature descriptor of SIFT by adding gravity. If we introduce an angle $\theta$, then the gravity-aware SIFT feature vector is $\boldsymbol{V} = (v_1, v_2, \cdots, v_{128}, \theta)$. While comparing between two gravity-aware SIFT feature vectors, we compute $|\theta_1 - \theta_2|$ first; and if it doesn't exceed the threshold, then we compare feature descriptors.

In mobile devices, acceleration sensors provide the direction of gravity for each image, and this can be described as $\boldsymbol{g} = [g_x, g_y, g_z]^{\mathrm{T}}$. Usually, $\boldsymbol{g}$ is normalized with $\|\boldsymbol{g}\| = 1$, and the gravity direction $\boldsymbol{d} = [d_u, d_v, 0]$ of $\boldsymbol{p} = [u, v, 1]$ can be computed as follows:

$$\boldsymbol{d} = \boldsymbol{p}' - \boldsymbol{p}, \tag{1}$$

where $\boldsymbol{p}' = [u', v', 1]$, and the parameter $K$ in a camera is a $3 \times 3$ matrix, and satisfies

$$[wu', wv', w] = \boldsymbol{p} + K\boldsymbol{g}. \tag{2}$$

In equation 2, $\boldsymbol{g} = [g_x, g_y, g_z]^{\mathrm{T}}$ can be acquired with acceleration sensor of a mobile phone, and $K = \begin{bmatrix} f_u & 0 & p_u \\ 0 & f_v & p_v \\ 0 & 0 & 1 \end{bmatrix}$, so the direction of gravity of $\boldsymbol{p}$ can be approximated with the following equation.

$$d = [g_z(p_u - u) + f_u g_x, g_z(p_v - v) + f_v g_y, 0] \tag{3}$$

In equation 3, $[p_u, p_v]$ is the center of camera, $f_u$ and $f_v$ are the focal lengths of horizontal and vertical axis. So, the angle $\theta$ can be computed as follows:

$$\theta = \arctan(\frac{d_v}{d_u}). \tag{4}$$

3.2. **Building forests with extremely randomized clustering.** After building feature descriptors for all images, we build extremely randomized forests [27] for category attributes of feature descriptors, and this process is a clustering process.

3.2.1. *Generating trees.* During the generation of trees, we use the Shannon entropy to decide whether or not to split a node. The computation of Shannon entropy is in equation 5.

$$S_C(L,T) = \frac{2 \times I_{C,T}(L)}{H_C(L) + H_T(L)} \tag{5}$$

where $H_C(L)$ is the entropy of category distribution in training data $L$, and can be computed as follows:

$$H_C(L) = -\sum_{c \in C} \frac{n_c}{n} \log_2 \frac{n_c}{n}, \tag{6}$$

where $L$ is the labeled feature descriptor set, $n$ is the number of feature descriptors in $L$, $n_c$ is the number of feature descriptors belong to category $c$. At the same time, the split entropy $H_T(L)$ is

$$H_T(L) = -\sum_{P=1}^{2} \frac{n_c}{n} \log_2 \frac{n_c}{n}. \tag{7}$$

Based on the entropy of a set, the information gain of a split can be computed as follows:

$$I_{C,T}(L) = H_C(L) - \sum_{P=1}^{2} \frac{n_p}{n} H_C(L_P). \tag{8}$$

When the uncertainty of each subset $L_i$ is 0, then the information gain of entropy $I_{C,T}(L)$ is maximum.

3.2.2. *Splitting nodes.* During the generation of extremely randomized trees, each node is randomly selected to split, so that all trees and all nodes in a tree are independent. Therefore, the clustering results of extremely randomized trees can be reinforced. At the same time, in order to keep the clustering accuracy of each tree, we must set a threshold for splitting. When the entropy of a split exceeds the threshold, we do the split. However, the number of candidate splits is limited, and in some nodes the entropy of all splits may be less than the threshold, so we need also split the nodes whose splitting results are not so good, and the direct effect of this phenomenon is longer training time and lower clustering accuracy.

In this paper, in order to split the nodes uniformly and randomly, we add balance factors in the splitting process. The addition of balance factors doesn't increase the computing complexity. We randomly select the numbers of both left and right sub-trees by statistics, and use the absolute value of their difference as balance factor. The aim of selecting balance factor is minimizing the differences of numbers of left and right sub-trees, and this can be described with equations 9 and 10.

$$balance_i = |count(R_{left}) - count(R_{right})|, i = 1, \cdots, MaxIter, \tag{9}$$

where $count(R_{left})$ and $count(R_{right})$ are the descriptors of left and right sub-trees splitted respectively, $i$ is the number of tried splits, $MaxIter$ is the maximum number of tried splits, and $balance_i$ is the balance factor of the $i$-th tried split.

$$\begin{aligned} &\forall i, i = 1, \cdots, MaxIter, \\ &\begin{cases} \text{if } S_i < S_{min}, & find(balance_{all}), \\ \text{if } S_i > S_{min}, & find(balance_c), \end{cases} \end{aligned} \tag{10}$$

where $S_i$ is the entropy of the $i$-th tried split, $S_{min}$ is the threshold of minimum entropy of splits, $balance_{all}$ are all balance factors, $balance_c$ is the balance factor that exceeds $S_{min}$, and $find(\cdot)$ is the splitting condition on finding minimum balance factor.

If the entropies of all randomly splits are less than the threshold, we select the splitting condition that has the minimum balance factor; if there are several entropies that are bigger than the threshold, we also select the splitting condition that has the minimum balance factor; and if there is only one splitting condition whose entropy is bigger than the threshold, then we select this splitting condition. After selecting the splitting condition, we split node under the selected condition, and make sure that the distribution of the splitted nodes is an approximately normal distribution.

3.2.3. *Improving differentiation of leaf nodes.* In original extremely randomized trees, when all feature vectors of a node belong to the same category, then the splitting is stopped and a leaf node is created. In this paper, by introducing the angle $\theta$ between direction of gravity with local direction of an image we fine-tune the feature descriptor, and thus improve the ability to classification.

For feature descriptors from different images, even though they have the same feature vector, their angle between direction of gravity with local direction of an image is different; and for feature descriptors belong to the same category, they not only have different feature vectors, but also have different angles. So, we re-cluster leaf nodes after adding the angle $\theta$, and this can make sure that similar angles are classified into the same sub-node. The clustering standards are described as equations 11 and 12.

$$splitAngle = R_{mid}(\theta), mid = \frac{count(R)}{2}, \text{and } R = sort(L(\theta)), \tag{11}$$

where $L$ is denoted as all feature descriptors of a leaf node, $sort(L(\theta))$ is the descending order of leaf nodes according to their angles, $R$ is the feature descriptor of leaf node after sorting, $count(\cdot)$ is the number of feature descriptors of a leaf node, $mid$ is the number of the median, and $R_{mid}(\theta)$ is the angle of feature descriptor in the median of a leaf node.

$$\begin{aligned} &n = 1, \cdots, count(S), \\ &split(L) = \begin{cases} R_n \rightarrow R_{left} | R_n(\theta) \leq splitAngle \\ R_n \rightarrow R_{right} | R_n(\theta) \leq splitAngle \end{cases} \end{aligned} \tag{12}$$

where $R_{left}$ and $R_{right}$ are the left and right sub-trees of leaf node respectively, $R_n(\theta)$ is the angle of the $n$-th node in all leaf nodes after sorting.

When the leaf nodes are sorted, if the angle of a node feature descriptor is less than $splitAngle$, we add the node to its left sub-tree; and add the node to its right sub-tree, otherwise. Figure 1 illustrates the splitting flowchart of a node. In the figure, the cycles are leaf nodes generated by extremely randomized clustering forests, the part below dashed line describes details about generation of new leaf nodes.

When processing queries about an image, we generate all feature descriptors with extremely randomized clustering forests, count the number of occurrences for each leaf node,

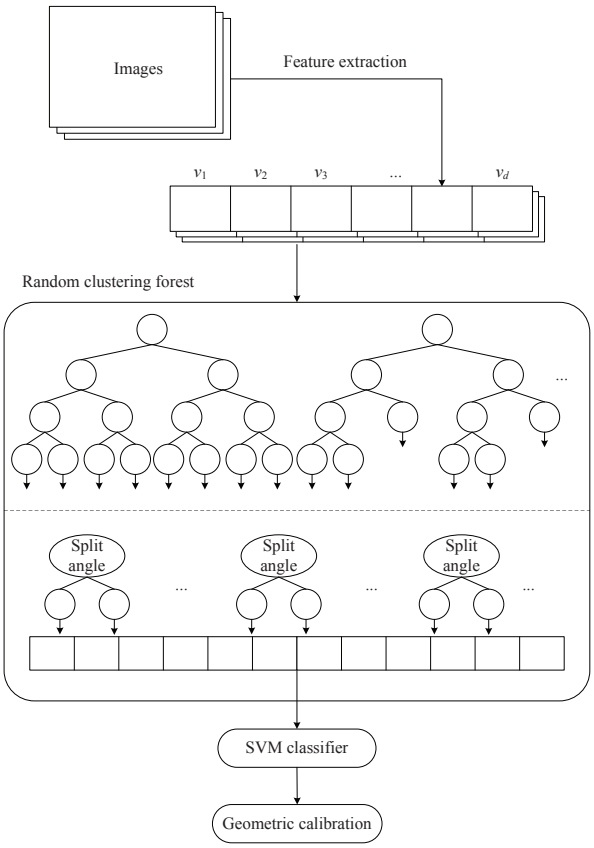


FIGURE 1. The flowchart of node splitting

and build a statistical histogram. After that, we classify these histograms with the support vector classifier. Details about how to split leaf nodes are in algorithm 1.

3.3. **Classification and geometric calibration.** Support vector machine (SVM) [28] is an efficient and effectively statistical method for classification and regression tasks. In this paper, we use support vector classifier to classify the extremely randomized clustering forests, and details about support vector machine can be seen in [28].

After classifying all images, we do geometric calibration [29] for each category. We compare the gravity-aware SIFT vectors between the query image with the centroid of each image category, and match the two images by computing their Euclid distance. If the Euclid distance is less than a fixed threshold, we assume that they match; and otherwise, we assume that they don't match. During geometric calibration of the matched feature descriptor, we delete the wrongly matched images, and further match the results. By setting the minimum matching threshold, we compare the remaining feature descriptors with the minimum matching threshold. If the comparing result is bigger than zero, then we think this classification is correct; and otherwise, we think it isn't correct, and then there is no matching image.

## 4. Experiments.

4.1. **Experimental setup.** The mobile platform that we use in our experiments is a smart phone, which includes a 1024MHz CPU, 1.5GB memory and a 32GB MicroSD running on Android 2.3. In the experiments, we use the public UKBench [30] dataset. UKBench includes 2550 different scenes, and each scene is photographed from 4 different viewpoint. The total number of images in the UKBench dataset is 10200.

**Algorithm 1:** Algorithm for splitting leaf nodes

Let $L_t$ be descriptors of all training samples, $LeafNum = 0$, $leaf_T$ be number of leaf nodes, $S_{min} = 0.5$, $T_{max} = 50$;
**repeat**
  $Tree(L_t)$; //build a tree for each $L_t$;
  $Select(i)$; //select a dimension randomly;
  $Sample(P_t)$; //sampling from a normal distribution;
  **if** *(stopsplitting*$(L_t) = true$*)* **then**
    $createLeafNode(L_t)$; //build a middle node with equation 12, let the angle be *splitAngle*;
    $leafNum = leafNum + 2$;
    **if** *(descriptors in* $L_t$ *belong the same category)* **then**
      **return**;
  **else**
    let $tries = 0$, and try split;
  **repeat**
    $tries = tries + 1$;
    $\Gamma = \{f_i \leq P_t\}$;
    $score = S_c(L, H)$;
    splitting $L_t$ according to $\Gamma$ and *score*;
  **until** $(score \geq S_{min})$ *or* $(tries \geq T_{max})$;
  **if** $(score_{all} \leq S_{min})$ *or* $(\exists score_j \geq S_{min})$ **then**
    $i = find(balance)$; //find minimum balance factor;
    $createDecisionNode(i, P_t, Tree(P_t), Tree(L_r))$;
  **else**
    $i = \max(score_n)$, for $n = (1, 2, \cdots, T_{max})$;
    $createDecisionNode(i, P_t, Tree(P_t), Tree(L_r))$;
    **return**;
**until** $(leafNum \geq leaf_T)$;

In order to validate the performance of the proposed method, we compare it with SIFT [13] and SURF [14] two classical feature descriptors. The metrics we use in the experiments are training time, training memory usage, recognition time and recognition accuracy.

4.2. **Experimental results.** In each experiment, we randomly select 20 scenes to train, and all images are converted to a $320 \times 320$ pixels. Each experiment is implemented 10 times, and we compute the average.

Firstly, we randomly choose 20 images from different scenes, run SIFT, SURF and our proposed method to extract features. We compute the average number of features, executing time and memory usage, and the result is in table 1. As can be seen from the figure that, our proposed method has the the most features with the least running time and memory usage.

Secondly, we compare the memory usage and running time of the three methods while training, and the results are in figures 2 and 3 respectively. During the comparison of training memory usage in figure 2, our proposed method is a little better than SURF, and

TABLE 1. Performance of different feature descriptors

| Algorithms | Feature No. | Running time (ms) | Memory usage (KB) |
|---|---|---|---|
| Ours | 146 | 272 | 16852 |
| SURF | 126 | 1281 | 17232 |
| SIFT | 119 | 1936 | 42632 |

both of them are obviously better than SIFT. During the comparison of training time, we can see from figure 3 that, our method is the fastest, SURF is the second, and both of them are faster than SIFT while training.

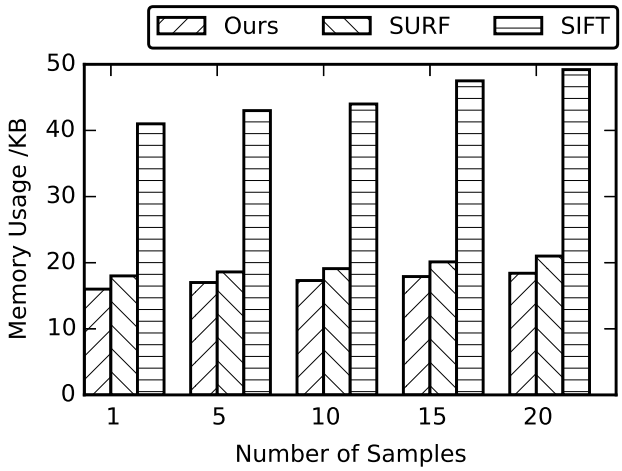


FIGURE 2. Comparison of training memory usage

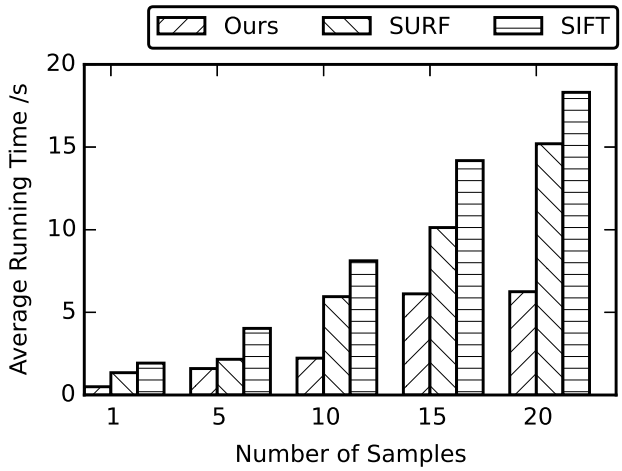


FIGURE 3. Comparison of training time

Nextly, we compare the running time while recognizing scenes with the trained models, and the results are in figures 4 and 5. Figure 4 illustrates the comparison of recognizing time between our method with SURF, and figure 5 illustrates the comparison of recognizing time between our method with SIFT. The recognizing time of our method is

slightly shorter than SURF, and both of them needs only dozens milliseconds, However, the recognizing time of SIFT is about several seconds from 0.5 to 12.

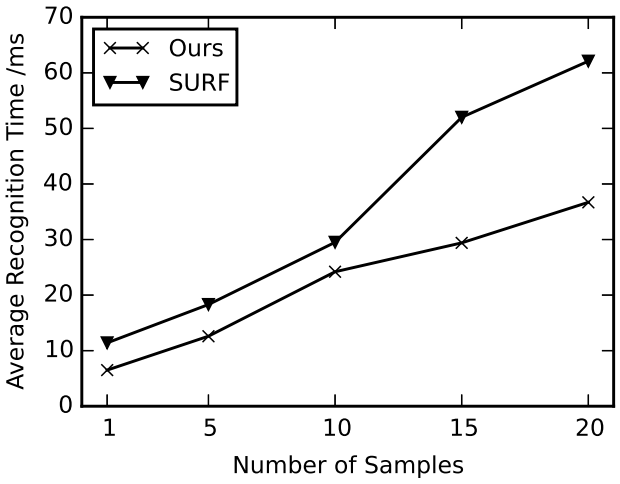


FIGURE 4. Comparison of average recognizing time, Ours vs. SURF

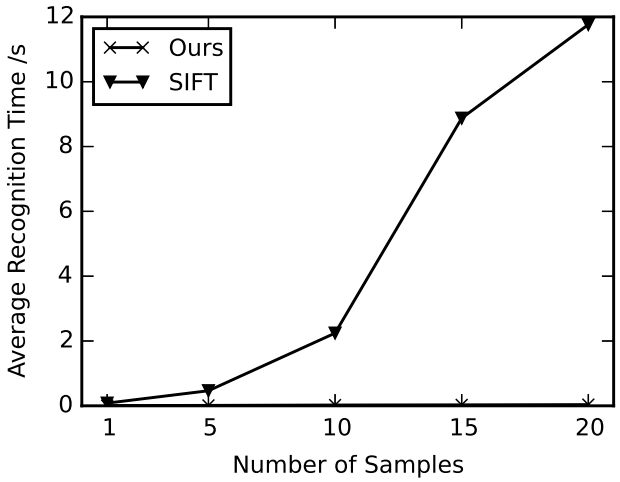


FIGURE 5. Comparison of average recognizing time, Ours vs. SIFT

Finally, we compare the average recognizing accuracy of the three methods in figure 6. From the figure we can see that, the recognizing accuracy decreases with the increase of number of samples. When we have small number of samples, all methods have good recognizing accuracy; when the number of samples reaches 10, the recognizing accuracies of all methods are above 80%; and when the number of samples reaches 20, the recognizing accuracy of SIFT drops quickly, and it is about 73%. So, the result shows that our method is stable in different number of samples while recognizing scenes.

5. **Conclusion.** Scene or object recognition in mobile devices is a hot research in both image processing and mobile computing fields. In this paper, we introduced the gravity feature into traditional SIFT feature descriptor, clustered the gravity-aware feature descriptors into extremely randomized trees, and classified the clustered tree with support vector classifier and geometric calibration. We validated the efficiency and effectiveness of our proposed method via massive experiments.

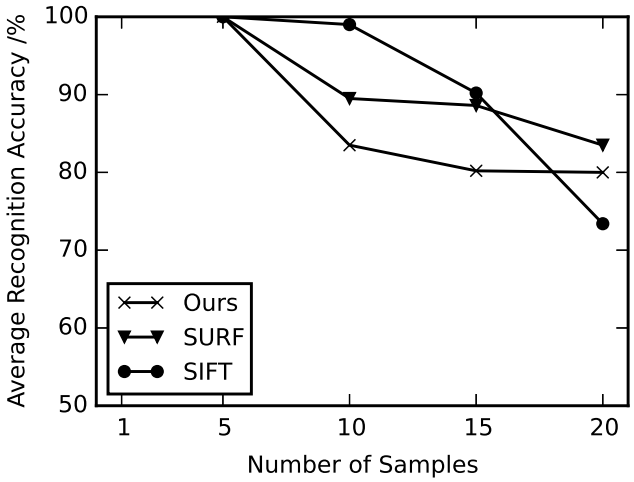


FIGURE 6. Comparison of average recognizing accuracy

**Acknowledgment.** The work was supported by the following funds, Science and Technology Project of Hunan Province of China (2014FJ6095) and Science Research Foundation for Distinguished Young Scholars of Hunan Province Education Department (14B070).

## REFERENCES

[1] P. Föckler, T. Zeidler, B. Brombach, E. Bruns and O. Bimber, Phoneguide: museum guidance supported by on-device object recognition on mobile phones. *Proceedings of the 4th International Conference on Mobile and Ubiquitous Multimedia*, pp.3-10, 2005.

[2] S. Gammeter, A. Gassmann, L. Bossard, T. Quack, L. Van Gool, Server-side object recognition and client-side object tracking for mobile augmented reality. *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference On*, pp.1-8, 2010.

[3] M. Satyanarayanan, Fundamental challenges in mobile computing. *Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing*, pp.1-7, 1996.

[4] H. Umudum, Smart phones and microscopic image capturing: Searching, Finding, capturing, talking about, as well as sharing microscopic images of pathologic conditions form another type of learning module. *Archives of Pathology and Laboratory Medicine*, vol.139, no.7, pp.847-847, 2015.

[5] F. Debole, C. Gennaro, P. Savino, Enriching image feature description supporting effective content-based retrieval and annotation. *Virtual Systems & Multimedia (VSMM), 2014 International Conference On*, pp.80-87, 2014.

[6] Y. Liu, X. Q. Lv, Y. Y. Qin, Z. Tang, J. B. Xu, Scale and color invariant image feature description. *Journal of Chinese Computer Systems*, vol.33, no.10, pp.2297-2302, 2012.

[7] P. Corke, Image feature extraction. *Robotics, Vision and Control*, pp.335-379, 2011.

[8] Z. Qin, J. Yan, K. Ren, C. W. Chen, C. Wang, Towards efficient privacy-preserving image feature extraction in cloud computing. *Proceedings of the ACM International Conference on Multimedia*, pp.497-506, 2014.

[9] T. Prod'homme, B. Holl, L. Lindegren, A. G. Brown, The impact of CCD radiation damage on Gaia astrometry–I. Image location estimation in the presence of radiation damage. *Monthly Notices of the Royal Astronomical Society*, vol.419, no.4, pp.2995-3017, 2012.

[10] W. Zhuang, M. Ismail, Cooperation in wireless communication networks. *Wireless Communications*, vol.19, no.2, pp.10-20, 2012.

[11] P. Q. Le, F. Dong, K. Hirota, A exible representation of quantum images for polynomial preparation, image compression, and processing operations. *Quantum Information Processing*, vol.10, no.1, pp.63-84, 2011.

[12] P. Anandan, R. Sabeenian, Image compression techniques using curvelet, contourlet, ridgelet and wavelet transformsa review. *Biometrics and Bioinformatics*, vol.5, no.7, pp.267-270, 2013.

[13] D. G. Lowe, Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, vol.60, no.2, pp.91-110, 2004.

[14] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (surf). *Computer vision and image understanding*, vol.110,no.3, pp.346-359, 2008.
[15] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, P. Fua,Brief: Computing a local binary descriptor very fast. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol.34, no.7, pp.1281-1298, 2012.
[16] E. Mair, G. D. Hager, D. Burschka, M. Suppa, G. Hirzinger, Adaptive and generic corner detection based on the accelerated segment test. *Computer Vision-ECCV 2010*, pp.183-196. 2010.
[17] S. Leutenegger, M. Chli, R. Y. Siegwart, Brisk: Binary robust invariant scalable keypoints. *Computer Vision (ICCV), 2011 IEEE International Conference On*, pp.2548-2555, 2011.
[18] E. Rosten, T. Drummond, Machine learning for high-speed corner detection. *Computer Vision-ECCV 2006*, pp.430-443, 2006.
[19] Z. W. Yuan, Y. H. Wang, Research on k nearest neighbor non-parametric regression algorithm based on kd-tree and clustering analysis. *Computational and Information Sciences (ICCIS), 2012 Fourth International Conference On*, pp.298-301, 2012.
[20] C. H. Lin, C. C. Chen, H. L. Lee, J. R. Liao, Fast k-means algorithm based on a level histogram for image retrieval. *Expert Systems with Applications*, vol.41, no.7, pp.3276-3283, 2014.
[21] T. Yamashita, Y. Yamauchi, H. Fujiyoshi, Human detection for multiple pose by boosted randomized trees. *Pattern Recognition (ACPR), 2011 First Asian Conference On*, pp.229-233, 2011.
[22] M. Ozuysal, M. Calonder, V. Lepetit, P. Fua, Fast keypoint recognition using random ferns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol.32, no.3, pp.448-461, 2010.
[23] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, D. Schmalstieg, Real-time detection and tracking for augmented reality on mobile phones. *Visualization and Computer Graphics, IEEE Transactions on*, vol.16, no.3, pp.355-368, 2010.
[24] W. C. Chen, Y. Xiong, J. Gao, N. Gelfand, R. Grzeszczuk, Efficient extraction of robust image features on mobile devices. *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp.1-2, 2007.
[25] J. Gu, R. Mukundan, M. Billinghurst, Developing mobile phone ar applications using j2me. *Image and Vision Computing New Zealand, 2008. IVCNZ 2008. 23rd International Conference*, pp.1-6, 2008.
[26] D. Kurz, S. Ben Himane, Inertial sensor-aligned visual feature descriptors, *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference On*, pp.161-166, 2011.
[27] F. Moosmann, E. Nowak, F. Jurie, Randomized clustering forests for image classification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol.30, no.9, pp.1632-1646, 2008.
[28] Y. Bazi, F. Melgani, Toward an optimal svm classification system for hyperspectral remote sensing images, *Geoscience and Remote Sensing, IEEE Transactions on*, vol.44, no.11, pp.3374-3385, 2006.
[29] O. Chum, J. Matas, Matching with prosac-progressive sample consensus. *Computer Vision and Pattern Recognition, CVPR 2005. IEEE Computer Society Conference On*, vol.1, pp.220-226, 2005.
[30] D. Nister, H. Stewenius, Scalable recognition with a vocabulary tree. *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference On*, vol.2, pp.2161-2168, 2006.