Application of landmark recognition on iPhone

LIN Chen-hao, FAN Meng-tian

(Electrical Engineering Department of Columbia University, New York, USA)

摘 要: Landmark recognition on iPhone was proposed in this paper, so as to enable the user to take photos of some famous landmarks in the world and return the result of what this landmark is as well as provide some useful information about this landmark such as travel guide, history and interesting activities. The approach to recognize the landmark is by extracting features, using bag-of-words model and support vector machine to train the dataset, hence to get the classification model. Then this model is used to predict photos the user has taken. This article compares the different algorithm like SIFT, Gist and Hog to extract features and build a classification model. This article uses PC as the server to recognize landmark photos and return results to users.

关键词: landmark recognition; iPhone; SIFT; gist; SVM; BoW

中图分类号: TN 929.5 **文献标志码**: A **文章编号**: 1006-7930(2013)05-0755-06

1 Introduction

With thephenomenal growth in the use of smartphones, people become more and more reliable on their phones since the smartphone is not only the calling and texting device, but also contains the function of browsing website, text editing, music player, etc. Besides, most of the smartphones often have cameras and basic image processing ability that have led to a huge explosion of the application in computer vision area. Among tons of different mobile devices, iPhone is no doubt in the top of the trends. In this way, we propose to develop an application on iPhone and gain some cool experience from this.

Recognition of the object on the images is a popular topic in the image processing and computer vision research. If it can be automatically known what it is in the images, it will have a pretty broad range of application fields. For example, it can tag a big amount of images online automatically and make us search the images with keyword more accurately. The landmark recognition is also a useful functionality that can improve the user experience a lot when they are travelling. They can get the information and travel guide of where they are by simply taking a photo.

1.1 Proposed idea

Based on the search we have done in this area, and consideration of its function, we want to implement. We propose an idea of doing an iPhone Application for Landmark recognition.

The user takes a photo of the landmark from where they are currently standing. Then the application sends the image to the server. And the server runs the program that we have done on the computer and returns the result of the recognizing progress. And it can also send back the related information about the landmark, so they will have such information like history, tourist guide and some activities.

收稿日期: 2013-03-20 **修改稿日期**: 2013-10-08

基金项目: Project supported by the National Natural Science Foundation of China (Grant No. 13BF053)

作者简介: LIN Chen-hao (1989-), Master, New York, USA, E-mail: linchenhao1989@gmail.com

1.2 Related work

A lot of researchers have introduced some good algorithm for Landmark recognition. Research like B. Yamauchi and P. Langley[1], have developed a technique for place learning and place recognition in dynamic environments, associating evidence grids with places in the world and using hill climbing to find the best alignment between current perceptions and learned evidence grids. A. Bosch and A. Zisserman and X. Munoz [2] have proposed an approach using pLSA, a generative model from the statistical text literature here applied to a bag of visual words representation for each image and trained a multi-class classifier on the topic distribution vector for each image. A. Torralba et al [3] present a context-based vision system for place and object recognition. They present a low-dimensional global image representation that provides relevant information for place recognition and categorization. And the algorithm has been integrated into a mobile system that provides real-time feedback to the user.

2 Landmark recognition approach

To build this landmark recognition system on iPhone, we should use the training data to build the classifier first, which is supposed to be the core of our application. Our recognition roughly includes several components as follows:

2.1 Features extraction

At first,we need to extract features before we are able to train the classifier. How to select features plays an incredible significant role in the performance of our recognition. There are mainly two kinds of features: global feature and local feature [4]. We will discuss this topic in detail as following:

(1) Global feature: The global feature usually includes color, edge, texture and so on. The advantage of the global features is that it can be computed easily and efficiently. However, it doesn't contain any position information of the object in the image. And it is sensitive to the illumination and contrast. So the global feature is not enough for us to describe a photo in order to recognize it.

At thebeginning of our project, we propose to use color histogram as the global feature since it is pretty straightforward and easy to realize. But we found that the photo of the landmark is often diverse with light condition, and contrast. And most of buildings often have similar colors. So it may not help us recognize the landmark properly, while it may confuse us with the result.

Gist feature is awell-known global feature as well, which characterizes several important statistics about a scene [5]. The Gist feature can encode the amount or strength of vertical or horizontal lines in an image, which for example can help to match scenes with similar horizon lines, textures, or building in them [6]. So Gist feature can potentially improve the drawback of the histogram. We may put it in the features data if it shows a good influence on the result.

(2) Local feature: The local feature is just the suitable compensation for the global feature. It can describe the interesting area and show the properties of it. And local feature is robust to the rotation, scale, illumination, weather and clutter which is very common in different landmark photos. After some investigation and based on our past experience, we decide to choose SIFT feature.

Scale-invariant feature transform (SIFT)[7] is a state-of-the-art feature often used in the computer vision area, like gesture recognition and video tracking. It transforms an image into a large collection of feature vectors, each of which is invariant to image translation, scaling, and rotation, particularly invariant to illumination changes and robust to local geometric distortion. Here are the main steps of using SIFT.

· SIFT keypoints of objects are first extracted from a set of reference images and stored in a data-

base. An object is recognized in a new image by individually comparing each feature from the new image to this database and finding candidate matching features based on. Euclidean distance of their feature vectors.

• From the full set of matches, subsets of keypoints that agree on the object and its location, scale, and orientation in the new image are identified to filter out good matches.

• The determination of consistent clusters is performed rapidly by using an efficient hash table implementation of the generalized Hough transform. Each cluster of 3 or more features that agree on an object and its pose is then subject to further detailed model verification and subsequently outliers are discarded. Finally the probability that a particular set of features indicates the presence of an object is computed, given the accuracy of fit and number of probable false matches.

• Object matches that pass all these tests can be identified as correct with high confidence.

2.2 Classification model building

Having got all the features of the patch in the images, we use Bag-of-words model to build a codebook to describe every image in the dataset. The bag-of-words model is a simplifying representation used in natural language processing and information retrieval. Also it is commonly used in methods of document classification, where the (frequency of) occurrence of each word is used as a feature for training a classifier. In this model, a text (such as a sentence or a document) is represented as an unordered collection of words, disregarding grammar and even word order. Recently, the bag-of-words model has also been used for computer vision.

We use K-means clustering to perform vector quantization of the patch descriptors from all the classes to form the codebook. Then we use the histogram of the code words to represent the image's dense patches. By several testing, we choose 300 as the number of the visual words.

After we represent the image by the BoW model, we use Support Vector Machine (SVM) to train these data. In machine learning, it is supervised learning models with associated learning algorithms that analyze data and recognize patterns, that are used for classification and regression analysis. The basic SVM takes a set of input data and predicts, for each given input, which of two possible classes forms the output, making it a non-probabilistic binary linear classifier. Given a set of training examples, each is marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or another. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap they fall on.

We try two ways to build our classifier. Firstly, we directly use multi-class SVM classifier, which means we set 15 different labels of the test and train data and then calculate the accuracy of this classifier. Secondly, for each category, we use 10 positive images and 10 negative images as the input. The negative images are selected randomly among all the other categories. When we are doing the training, we build 15 bi-class classifiers, and for each test group, we calculate the accuracy of these 15 classifiers and make comparisons.

2.3 Client-server architecture

Since calculating the features and doing the classification takes a long time and the processor of the iPhone is much slower than the computer, we decide to build a client-server system. The iPhone is set to be the client. The user uses iPhone to take photos and sends the photo to the server. The server receives the photo and returns the result of the program. It runs much more sufficiently than putting all the programs on the iPhone.

3 Implementation and result

In this part we introduce the implementation scenario of our application including the developing platform and dataset. Then, we show the results of our testing system and analysis of the performance.

3.1 Developing platform

Since we use the client-server architecture for the application, we can implement the classifier model to do the match on the computer instead of the iPhone. For the implementation of building the classification model and doing queries from the user, we use Matlab because we are both familiar with it and it always does a great job in realizing classification.

On the iPhone side, we use Objective-C to build the user interface, and there is no choice for us. It is a totally different language with special grammar compared with C and Java. Because we will not focus on the design of the user interface part, we have just built a simple interface for the interaction with the server.

On the server side, we use Django 1.4[8] to implement. Django is a high-level Python Web framework that can help us to build application more easily. It is easy sending and receiving data between Matlab program and iPhone application. For the iPhone, we use AFNetworking[9] framework to connect with the sever. AFNetworking is a delightful networking library for iOS. It has a modular architecture with well-designed, feature-rich APIs that are easy for us to handle.

3.2 Dataset

We tried toexplore available dataset on the Internet but they are not perfectly suitable for our project. So we created a landmark dataset by downloading the images online. It consists of 300 images of landmarks—15 categories and 20 images per category(Fig. 1). For every category, we selected images taken from different angels, illumination and other condition in order to cover different kinds of photos the user may take.

3.3 Resultand analysis

For the experiment, we used four methods including SIFT feature matching, Gist feature matching, using SVM on the Gist feature, and BoW (Bag of words) model. The train set has 150 images, 10 images per class and so is the test set. The result is basically satisfying.

Fig. 1 Sample images from the dataset
Row 1: Liberty Status. Row 2: Effiel Tower

(1) Using SIFT feature and matching: First, we tried the basic SIFT extraction and used Euclidean distance to do the matching. The result is shown in the first column in Table I. We can see that the result is not so satisfying. Most of the classes only have 30%～40% accuracy. This result is not so surprising based on the observation of the images in the dataset. SIFT extracts all the interest points features and has no selection. However, just thinking about the photo of the landmark taken during our traveling, it always has blue sky, green grass and some unnamed buildings around the landmark. So when we are doing the matching, the one has the closest distance may not have the same landmark, while the most consistent one could have the same background and other useless parts.

(2) Using Gist feature and matching: As we expected, Gist should have better performance. In fact, it did do very well and most of the classes return 9 out of 10 right answers. The Gist feature can encode the amount or strength of vertical or horizontal lines in an image that can help to match scenes with similar horizon lines, textures, or building in them. And this kind of feature is suitable and reliable for our dataset, since it will focus on the main buildings in images instead of a background or other objects. Gist feature is perfectly and widely used in the scene classification and recognition.

(3) Using Gist feature and SVM classifier: After doing the Gist feature matching, we tried to use SVM, both multi-class classifiers and bi-class classifiers, to train the Gist feature data in order to get a better result. However, since the size of our dataset limits, the accuracy is already high leaving no space for improvement. We got a 6% improvement. In the future, if we use a larger dataset to build test our system, the accuracy will be improved.

Tab. 1　Result of recognition

Name of Landmarks	Accuracy			
	SIFT L_2-Dist/%	Gist L_2-Dist/%	Gist SVM/%	SIFT BoW SVM/%
Effiel Tower	30	60	60	30
Liberty Status	50	90	70	70
Congress of the United States	30	90	90	50
Triumphal Arch	60	80	100	40
Colosseum	80	100	100	70
Leaning Tower of Pisa	40	80	90	40
Sydney Opera House	30	80	100	40
Taj Mahal	70	50	60	80
The Great Sphinx	60	80	80	60
Forbidden City	30	90	90	40
Christ the Redeemer	80	80	90	30
Burj Al-Arab Hotel	40	80	90	70
Westerminster Church	100	100	100	90
Louvre Museum	30	60	70	80
Nest	100	90	100	60
Average	62	80.67	86	56.67

(4) Using BoW for SIFT feature and SVM classfier: Though this method is the most complicated one, it doesn't return the effort of our working. It has similarly result with SIFT feature. The reason is that BoW needs really large dataset to build the clustering and get codebook. But the size of dataset is not so large, and also a lot of unrelated components may be treated as a code, such as the similar background of the images. So the result is not so surprised. In the future, if the test is set up on a larger dataset and choose central part of dataset images to build codebook, we believe this function will produce a good result.

4　Conclusion and future work

Basically, wehave realized the recognition of the landmark on iPhone and got a pretty good result. And from the result of all the methods we used, we found that the Gist is the best. However, we still have some limitations.

Although nowadays smartphones are much more powerful than before, they are still limited compared to the computer. Basically it is the problem of the processor's power and low memory. It is much slower running a program like our app on iPhone than Macbook since the image contains a large number of data and the algorithm we used is complex. If the responding time is too long, it will be unsatisfying to the user. So we have to connect to the server that make user need to use mobile network.

This application has space to be improved. In the future, we want to add the return of useful information about the landmark the user queries and add some share function with Facebook, flicker and

so on.

The iPhone also has a good built-in accessory called magnetometer that can act as a digital compass. And it also has the GPS component. These two can play an important role when we are implementing landmark recognition. For time limitation, we cannot combine these into our system. Hope we can do it in the future work and give a better performance.

参考文献 References

[1] YAMAUCHI B, LANGLEY P. Place recognition in dynamic environments [J]. Robotic Systems, 1997, 14(2): 107-120.

[2] BOSCH A, ZISSERMAN A, MUÑOZ X. Scene classification using a hybrid generative/discriminative approach [J]. IEEE Trans. Pattern Analysis Machine Intelligence, 2008, 4(3): 712-726.

[3] TORRALBA A. Context-Based Vision System for Place and Object Recognition[C]//Proc. IEEE Int'l Conf. Computer Vision. IEEE Press, 2003, 1(2): 273-280.

[4] YAP K, CHEN T, LI Z et al. A Comparative study of mobile-based landmark recognition techiniques[J]. Intelligent Systems IEEE, 2010, 25(2): 48-57.

[5] OLIVE A, TORRALBA A. Building the gist of a scene: the role of global image features in recognition[J]. Visual Perception, Progress in Brain Research, 2006, 155(2): 59-63.