Who is Here: Location Aware Face Recognition

Zixuan Wang
Electrical Engineering
Stanford University
zxwang@stanford.edu

Jinyun Yan
Computer Science
Rutgers, New Brunswick
jinyuny@cs.rutgers.edu

Cong Pang
Computer Science
National University of Singapore
pangcong@nus.edu.sg

David Chu
Microsoft Research
Redmond, WA
davidchu@microsoft.com

Hamid Aghajan
Electrical Engineering
Stanford University
aghajan@stanford.edu

Abstract

Face recognition has many challenges. For instance, the illumination, various facial expression and different viewpoints add difficulties to identify the same person from a bunch of images. Searching over a huge set of images will only amplify such difficulties. We introduce the location aware face recognition framework for mobile-taken photos to alleviate the hardness. With the help of location sensor on the mobile devices, we collect images with location information. We propose an algorithm to reduce the search space of face recognition and therefore achieve better accuracy. Photos are clustered by locations on the server. Each location is then associated with a face classifier. Every client can send a "Who is Here" type query to the server by uploading an image with the location. The algorithm on the server will search over the given location and identify the person on the image. Experiments are conducted on mobile devices. The results are quite promising that higher accuracy is achieved and the query can be answered in near real-time.

1 Introduction

Face recognition is a well studied area, due to its importance in security and intelligent social network applications. However, it still has many challenges. For a single person, he can have many facial expressions which are difficult to match. Similarly, hair style, cosmetics, with or without glasses, the illumination and the varied viewpoints can cause strikingly different features of the face. For a system that answers a query of identifying a random person in the image, searching over all images and doing pairwise comparisons between face features will amplify all these difficulties and make the task extremely hard.

The new generation of mobile devices sheds light of solutions to alleviate this problem. There are many sensors embedded in the smart, small, thin and light mobile devices. They can tell where users are standing and what direction they are moving. Some even have sensors for temperature and humidity, combining with microphones to better determine the location and surroundings. In addition, smart phones are used as go-to cameras by more and more people, because they are easy to carry and convenient to use. A study said smartphones took 27 percent of all photos in 2011, while regular cameras account for 44 percent. In Flickr, an online social photo sharing website, iPhone 4 is responsible for more photos posted to this site than any other devices. These facts show that we can easily obtain a huge data set of mobile-taken photos with extra sensor information associated to each photo.

Among all sensors, we are interested in the location information. We introduce the location aware face recognition problem and investigate how we can take advantage of location information of the image. The intuition is that given information of where the user is, we can narrow down the search space of who the user is. People took photos at a certain place. A user will have different probabilities to appear in photos which are taken at some places than other places that the user has never been to. For instance, supposing Alice lives at Palo Alto, California, photos taken at Alice's home have very high chance that belong to Alice, her family and friends, instead of a random resident from the south of China. Therefore, when we want to identify the person in a photo, we can save some effort and gain accuracy by only comparing photos which are taken at places that person usually appears.

Based on these assumptions, we build the system with a location based data structure to organize photos and recognize faces. Formally the face recognition problem is : Given a set of face images labeled with the person's identify (the training set) and an unlabeled set of testing photos from the same group of people (the testing set), we aim at identifying each person in the testing photos. In our system, each face image associates with a location. The server creates many clusters of locations from the training set. Each location cluster contains a set of users who have photos in that location, their photos, besides photos of their friends. The client can take a photo and attach its location information,

SenSys'12, November 6–9, 2012, Toronto, ON, Canada.

then send it to the server and query for the person in the photo. The server will answer the query and return the identification of the person in the photo. Challenges lie in how to form the location clusters and what granularity we choose for locations; how to process the photo and extract useful features; how to search smartly to recognize the face and identify the person; what we can do to accelerate the whole process and avoid long response time on the client side. We will explain our approach to tackle each challenge in the paper.

We conducted experiments on a dataset containing about two thousand images with the ground truth. We compared the face recognition accuracy between our approach with an algorithm which does not consider locations. The result shows a significant improvement on the accuracy. We also present that the system runs quite fast and the whole process takes less than two seconds.

Our main contributions are as follows.

1. We make use of the location information from mobile-taken photos and propose a face recognition algorithm which reduces the search space to a large extent. We build a hybrid face recognition algorithm. We will firstly searching and matching photos within the given location, if this fails, we then search over all photos.
2. We take into account of the social network information. When we identify a user appearing frequently in a location, his friends also have high chance to show up in that location. Thus friends photos are also used to train the face classifier for the location.
3. We transmit the compressed face descriptor to the server for the query rather than sending the original image, which saves the network traffic and reduces the response time.

The paper is organized as follows: Section 2 outlines the recent work in the mobile computing and the face recognition. Our system and algorithm are presented in Section 3. Experimental evaluations are shown in Section 4. Finally, we conclude the paper and discuss the future work in Section 5.

2 Related Work

With the improvement on the computation power on mobile devices, many researchers seek to shift the computation to the mobile client. In [4], the authors propose a tool called Kobe that aids mobile classifier development, which helps to optimize mobile classiers for accuracy and cost. PhoneGuide [2] is one of the first object recognition systems performing the computation on a mobile phone, instead of sending the images to a remote server. The system employs a neural network trained to recognize normalized color features and is used as a museum guide. Seifer et al. [13] use a mobile system based on a hand-held device, GPS sensor, and a camera for roadside sign detection and inventory. Their algorithm was efficient enough to ensure good quality results in mobile settings. In the context of augmented reality, Fritz et al. [6] use a modified version of the SIFT algorithm [10] for object detection and recognition in a relatively small database of mobile phone imagery of urban environments. The system uses a client-server architecture, where a mobile phone client captures an image of an urban environment and sends it to the server for analysis. Takacs et al. [14] build an outdoor augmented reality system for mobile phones that matches camera-phone images against a large database of location-tagged images using a robust image retrieval algorithm. Their idea to store a set of informative features for each location is similar to ours but their goal is to recognize rigid objects such as buildings or road signs.

Besides the general object recognition, there has been considerable work on automatic face recognition. The descriptor based methods [3] [21] and subspace based methods [16] [1] [20] are two representative appearance-based approaches. The descriptor based methods extract discriminative information from the facial landmarks, and the subspace-based algorithms learn an optimal subspace for recognition. Context information has also been exploited to help face recognition [7]. Context information that has been investigated includes body and clothes, GPS tags and time stamps, people co-occurrence, etc. In the most recent work, people, location and event have been jointly recognized by a generic probabilistic model [9].

Several face recognition algorithms are implemented on mobile devices. In [8], the authors prototype and test a face recognition tool on the smartphone for blind users. The tool utilizes smartphone technology in conjunction with a wireless network to provide audio feedback of the people in front of the blind user. Qin et al. [12] build an automatic tagging system on a mobile phone that senses the people, activity, and context in a picture. They develop three different methods based on posing, compass, and movement, to identify the people in a picture.

3 The Framework

Figure 1 shows the overview of our idea. On the client side, the user uses the mobile phone to take a photo of the people and sends the recognition query to the server via wireless networks. The face features are extracted and compressed on the phone. Both face features and location information are transmitted to the server for recognition. On the server side, we organize the face database by locations. The face database at one location contains images of people appeared there in the past. We also augment images from friends in the social network. The intuition is that people may invite their friends to visit their home or office with some probability so we need to include them into our face database associated with this location. Some public locations such as landmarks, streets, and parks might contain many random people; while other locations like home and office mainly contain a specific group of people. The latter type of locations is more useful for our algorithm. Images within a location are then used to train a classifier. We also maintain a backup classifier when is obtained from all images in our database. When the query at one location fails, we use the backup query to identify the person.

3.1 Location Clustering

When we have a collection of labeled photos with geo-location information, we use the agglomerative clustering to discover location clusters. We consider each geo-location data including the longitude and the latitude as a point in the two dimensional space. Initially, we have n points and assign

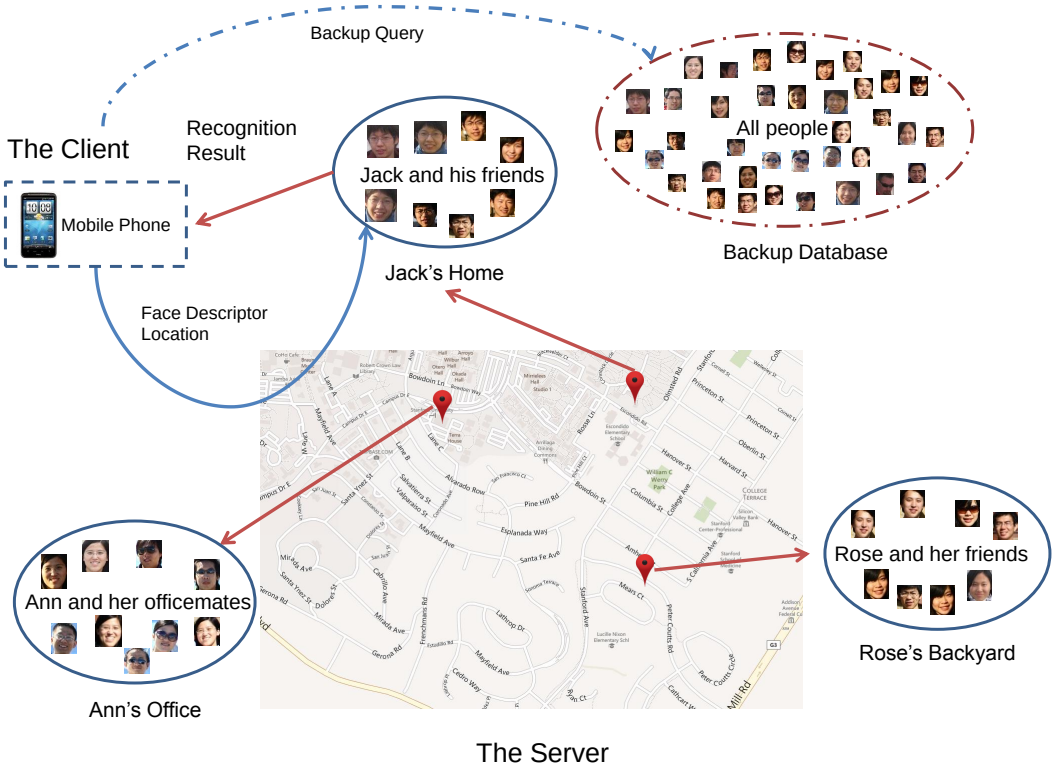

Figure 1. The overview of our location aware face recognition algorithm.

them to n different clusters. In each iteration of the clustering algorithm, we merge two clusters if the distance between two clusters is the minimum among all pairs of clusters. The distance between two clusters A and B is defined as:

$$d(A,B) = \frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} d(a,b) \qquad (1)$$

where $d(a,b)$ is the Euclidean distance between point a and b. We keep merging clusters until the minimum distance in each iteration is above a threshold or the number of clusters we want to obtain is reached.

3.2 Face Feature

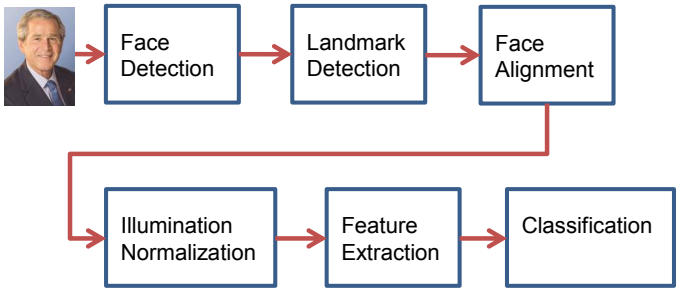

Figure 2. The pipeline of the face feature extraction.

To train a classifier for each location, we need to convert photos associated with the location to feature descriptors. To describe each face, we adopt local descriptor based face feature pipeline, which is shown in Figure 2. The Viola-Jones face detector [18] is used to detect face patches from the input image. To obtain high accuracy, a nested nose detector is applied to reduce the false positive rate. All face patches are normalized to the same size. We use the algorithm in [17] to detect seven facial landmarks from each face patch including four eye corners, two mouth corners and the nose. To align each face patch, four landmarks (outer eye corners and mouth corners) are registered to the pre-defined canonical positions using the affine transformation. The canonical position of each landmark is obtained from averaging landmarks of all faces. Then all seven facial landmarks are aligned by the computed affine matrix. Let $\mathbf{x}^f = (x_0^f, x_1^f, 1)^{\mathrm{T}}$ be the homogeneous coordinates for the landmark f of a non aligned image, and $\mathbf{y}^f = (y_0^f, y_1^f)^{\mathrm{T}}$ the desired coordinates for the same landmark. We want to obtain the affine transformation $\mathbf{A}_{2\times 3}$ such that $\mathbf{y}^f = \mathbf{A}\mathbf{x}^f$. To obtain the six parameters of $\mathbf{A}$ only three landmarks are needed, however, we can use more landmarks to obtain the set of parameters which minimize the least squares error.

We use the method similar to [15] to remove the effects of illumination. First, Gamma correction is used, i.e. a transformation of the pixel gray level values I using the non-linear transform $\hat{I} = I^{\gamma}$, with $0 < \gamma < 1$. This enhance the dynamic range by increasing the intensity in dark regions and decreasing it for bright regions. Next, the image is convolved with a Difference of Gaussians (DoG) kernel, a bandpass filter

which is intended to remove gradients caused by shadows (low frequency), to suppress noise (high frequency), and maintaining the useful signal for recognition (middle frequency). Finally, the histogram equalization is used to have a standard contrast spectrum for the image.

From each landmark, two SIFT descriptors [10] of different scales are extracted and concatenated to form the face feature descriptor.

3.3 System

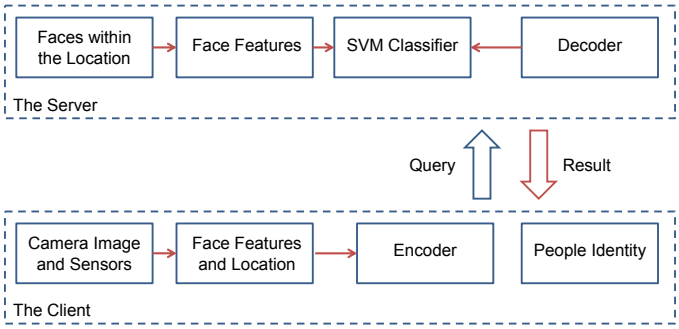

Figure 3. The client server architecture of the system.

Figure 3 shows the architecture of our system which is divided into two major components, the mobile client and the server. These components communicate over a wireless network.

On the server side, face descriptors are extracted from photos and trained by a Support Vector Machine (SVM) classifier [5] with the linear kernel. Each location has its own SVM classifier and is represented by the coordinate of the cluster center. We store all locations in an Approximate Nearest Neighbor (ANN) [11] structure for the fast nearest neighbor search. When a query is received on the server, it checks the location information of the query and finds the nearest location in the database. The classifier associated with the nearest location is used for the face recognition. A confidence score is defined for each location based query, which reflects how well the query is classified. If the confidence score is below a threshold, we can infer that the query people may not appear at the current location. We then redirect the query to our backup database to search over all photos.

On the client side, the face descriptor is computed using the same pipeline in Section 3.2. To save the bandwidth, we transmit the compressed descriptor over the wireless network. The Lempel-Ziv-Welch (LZW) compression algorithm [19] is used to compress the descriptor before it is sent to the server. We are able to produce an efficient encoding for the descriptor that is $1/3$ smaller than the original at the same recognition performance.

4 Evaluations

We use the dataset to validate our approach which contains $2,001$ images taken from 60 people. The dataset has 6 locations. We know the names and the social network relations among these 60 people as the ground truth. After this step, each location is augmented with images from friends in the social network. We also store a backup database containing images of all people, which is used when the location based query fails. The distribution of people at each location is shown in Figure 4.

Table 1. Evaluation Summary. Each row represents the accuracy of 5 fold cross validation at one location. In each test, 80% images are used as the training set and 20% images are used as the testing set.

	Test 1	Test 2	Test 3	Test 4	Test 5	Average
L1	0.876	0.907	0.814	0.897	0.887	0.876
L2	0.861	0.861	0.806	0.750	0.722	0.800
L3	1.000	1.000	1.000	0.941	0.941	0.976
L4	0.581	0.645	0.516	0.742	0.548	0.606
L5	0.563	0.688	0.625	0.750	0.688	0.663
L6	0.714	0.762	0.619	0.810	0.857	0.752
L7	0.574	0.525	0.495	0.535	0.525	0.531

4.1 Face Recognition Accuracy

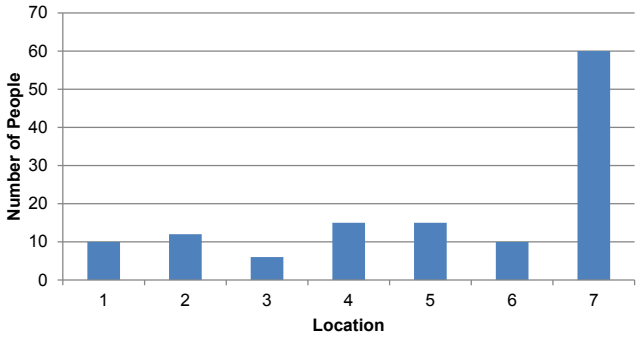

Figure 4. The distribution of the number of people at each location. Location L1 to L6 are 6 locations in our dataset. L7 is used as the backup database which contains images of all people.

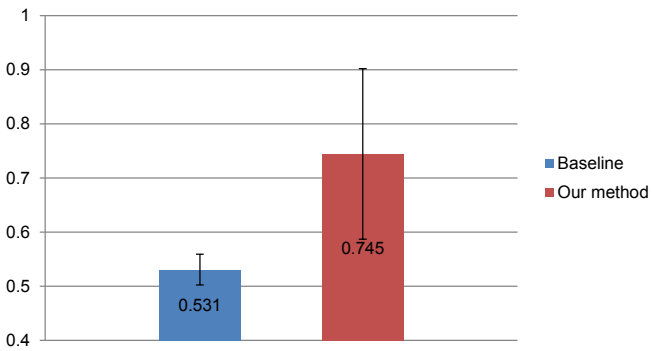

Figure 5. The performance of our method compared with the baseline method. The error bar represents the standard deviation in 5 tests.

To extract the face descriptor, each face patch is normalized to 128×128 pixels and is registered to the canonical pose using the affine transformation. To remove the illumination, we use the DoG filter with $\sigma_0 = 1$ and $\sigma_1 = 2$. The

Table 2. The time cost of each step on the phone.

	Face Detection	Feature Extraction	Compression	Transmission	Total
Time (second)	0.73	0.39	0.14	0.38	1.64

final face descriptor is formed by the concatenation of SIFT descriptors, obtained from two different scales (4 and 8 pixels width) at each landmark. From our experiments, we find that the selection of SIFT scales is very important.

The five-fold cross validation is conducted to test our approach. We compare our approach with the baseline method which uses no location information. On this dataset, we improve the accuracy from 0.531 to 0.745, which is shown in Figure 5. Each validation result is shown in Table 1.

4.2 Computation on the Phone

We test our algorithm on the Samsung galaxy S II i9100 1.2GHz with 2 cores and 1GB RAM to obtain the benchmark. The phone is running on Android 2.3.6 and our algorithm is written in the native code. The captured image is down-sampled to the resolution of 640×480 for the following processing. Table 2 shows the cost of each step on the phone. Each SIFT descriptor has 128 dimensional floating point numbers and 7×2 descriptors are extracted per face. Therefore, each face descriptor takes 7k bytes before compression. After the compression the size of the descriptor is around 4.8k bytes. It takes 0.38 seconds in average to transmitted the compressed descriptor via 3G networks.

4.3 Computation on the Server

A single workstation with Intel i7 1.6GHz with 4 cores and 8GB RAM is deployed as the server. The nearest location of the query is first found from the ANN structure. The complexity of querying the ANN structure is $O(\log k)$, where k is the number of locations in our system. The average time to find the nearest location is 0.001ms. After we know the current location of the user, we query the SVM at this location to get the recognition result. The average running time is 0.27ms. The complexity of querying the SVM is $O(n)$, where n is the number of people at one location. The space requirement of SVM only depends on the number of people at one location not the number of training data, which saves lots of space when the training set is huge.

5 Conclusion

We propose a novel face recognition algorithm on mobile phones, which leverages the location information to improve the recognition accuracy. It is an interesting problem and has many potential applications. One example is that the police can use our application to query any suspect and search for his identity without stopping him. There are many future works that can be done. Scalability will be one issue. When there are too many locations, is it better to build an index or ontology of locations to improve the search? Our system now supports querying persons who exist in the training set. It's useful to have some mechanisms to increase the training set incrementally through the aid of social network or the crowd wisdom. Another direction is handling the situation that people move from one place to another. In current system, one person can be in many locations, which causes some duplications. It will be interesting if we can build a local graph of locations for a single person, such that we can easily search all possible locations a person will appear. The last but not the least, other sensors including audio or motion sensors might contribute to face recognition and person identification as well.

6 References

[1] P. N. Belhumeur, J. a. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *TPAMI*, 1997.

[2] E. Bruns and O. Bimber. Adaptive training of video sets for image recognition on mobile phones. *Personal Ubiquitous Computing*, 2009.

[3] Z. Cao, Q. Yin, X. Tang, and J. Sun. Face recognition with learning-based descriptor. *CVPR*, 2010.

[4] D. Chu, N. D. Lane, T. T.-T. Lai, C. Pang, X. Meng, Q. Guo, F. Li, and F. Zhao. Balancing energy, latency and accuracy for mobile sensor data classification. In *SenSys*, 2011.

[5] C. Cortes and V. Vapnik. Support-vector networks. In *Machine Learning*, 1995.

[6] G. Fritz, C. Seifert, and L. Paletta. A mobile vision system for urban detection with informative local descriptors. In *ICVS*, 2006.

[7] A. C. Gallagher and T. Chen. Using context to recognize people in consumer images. *IPSJ Transactions on Computer Vision and Applications*, 2009.

[8] K. Kramer, D. Hedin, and D. Rolkosky. Smartphone based face recognition tool for the blind. In *EMBC*, 2010.

[9] D. Lin, A. Kapoor, G. Hua, and S. Baker. Joint people, event, and location recognition in personal photo collections using cross-domain context. In *ECCV*, 2010.

[10] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[11] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, 2009.

[12] C. Qin, X. Bao, R. Roy Choudhury, and S. Nelakuditi. Tagsense: a smartphone-based approach to automatic image tagging. In *MobiSys*, 2011.

[13] C. Seifert, L. Paletta, A. Jeitler, E. Hödl, J.-P. Andreu, P. Luley, and A. Almer. Visual object detection for mobile road sign inventory. 2004.

[14] G. Takacs, V. Chandrasekhar, N. Gelfand, Y. Xiong, W.-C. Chen, T. Bismpigiannis, R. Grzeszczuk, K. Pulli, and B. Girod. Outdoors augmented reality on mobile phone using loxel-based visual feature organization. In *Proceedings of the 1st ACM international conference on Multimedia information retrieval*, 2008.

[15] X. Tan and B. Triggs. Enhanced local texture feature sets for face recognition under difficult lighting conditions. *Image Processing, IEEE Transactions on*, 2010.

[16] M. Turk and A. Pentland. Face recognition using eigenfaces. In *CVPR*, 1991.

[17] M. Uřičář, V. Franc, and V. Hlaváč. Detector of facial landmarks learned by the structured output SVM. In *VISAPP*, 2012.

[18] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001.

[19] T. Welch. A technique for high-performance data compression. *Computer*, 1984.

[20] L. Wiskott, J.-M. Fellous, N. Kuiger, and C. von der Malsburg. Face recognition by elastic bunch graph matching. *TPAMI*, 1997.

[21] Q. Yin, X. Tang, and J. Sun. An associate-predict model for face recognition. In *CVPR*, 2011.