Handheld Face Identification Technology in a Pervasive Computing Environment[⋆]

Eugene Weinstein[1], Purdy Ho[2], Bernd Heisele[3], Tomaso Poggio[4], Ken Steele[1], and Anant Agarwal[1]

[1] Massachusetts Institute of Technology, Laboratory for Computer Science,
200 Technology Square, Cambridge, MA 02139 USA
{ecoder, steele, agarwal}@lcs.mit.edu
[2] Hewlett Packard Corporation, Cambridge, MA USA
purdy_ho@hp.com
[3] HONDA R&D Americas, Inc., Cambridge, MA USA
heisele@ai.mit.edu
[4] MIT, Center for Biological and Computational Learning, Cambridge, MA USA
tp@ai.mit.edu

Abstract. We describe initial efforts at implementing a flexible, modular face identification framework in the context of a pervasive computing environment including handheld computers. Currently, the handheld is used for image capture, while all computation is performed on a back-end server, though other possibilities are discussed. Handheld face identification is difficult due to limitations in processing power and variations in lighting and background conditions. Initial experimental data are presented, showing recognition accuracy of 95.6% for a three user domain, with 1.7 second identification latency. Work is underway to improve both accuracy and scalability.

1 Introduction

Project Oxygen [1], a joint undertaking between the MIT Laboratory for Computer Science, the Artificial Intelligence Laboratory, and six industry alliance partners, is investigating methods of integrating various advanced technologies to make computational resources truly pervasive. Since September 2001, the Project Oxygen core team has worked on implementing an infrastructure to allow face identification technology to be incorporated into pervasive computing applications. There are two main goals for this work: (1) to create an infrastructure to allow developers with little or no expertise in the technology to use face identification in their applications, and (2) to experiment with face identification in a handheld computing environment.

2 Background and Motivation

An essential component of a natural human-computer interface is the ability of a computer to be aware of its environment, its user, and the user's community.

[⋆] Appeared in short paper session, Pervasive 2002, Zürich, Switzerland

There are many compelling reasons for exploring face identification technology for handheld computers. Since a person's face is exposed to the public, capturing face images is less intrusive than fingerprint, iris, or retinal scanning. Also, whereas other biometric devices require specialized hardware, face identification can be done with a standard low-resolution camera, which is already needed for video-conferencing and other applications. Face identification systems have a relatively simple enrollment process – the user takes snapshots of his or her face, which can be performed in the background as the handheld is being used.

Over the past 20 years numerous face identification papers have been published; a survey can be found in [2]. Most previous face identification systems have been implemented using stationary cameras with fixed background and light conditions, with recognition algorithms running on high-speed servers. Face identification using handheld devices is a relatively novel and challenging problem due to the significant variations of background and illumination conditions, as well as the constraints of lightweight mobile devices.

3 System

3.1 Computing Platform

The computing platform used in this project falls within the framework of the Project Oxygen pervasive computing infrastructure [1]. The handheld computer is a Compaq iPAQ with a 206 MHz StrongARM processor and 64 MB of DRAM, running Familiar [3], a port of the Linux OS. A custom expansion sleeve designed by Compaq's Cambridge Research Laboratory [4] and built by Delta Electronics [5] contains a color camera and PCMCIA slots for 802.11b. While the camera is capable of 640×480 resolution, the image is decimated in hardware to 160×120, which is sufficient to obtain a clipped 40×40 image of the face.

At the time of writing, the handheld was used only for image capture. After capture, images are sent over an 802.11 wireless network to computation servers, which run the face detection, recognition, and training algorithms. This allows for faster performance at runtime, since the desktop servers are at least an order of magnitude faster than the handheld (currently, we are using a 2 GHz Pentium IV with 512 MB of RAM). However, in the future it will be desirable to run face detection and recognition algorithms on the handheld itself (see Section 7.1).

3.2 Component Location

The design of the handheld face identification environment stresses modularity of its core components, which are: (1) a handheld computer image capture client, (2) a face detection server, for cropping faces from captured images, (3) a face recognition server, for identifying a cropped face, and (4) a face recognition training server for training the classification mechanism based on captured data.

The modularity of the core components facilitates the creation of distributed systems utilizing face identification. Figure 1 shows our current arrangement of

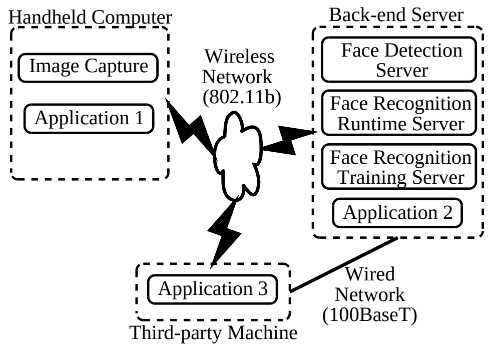

Fig. 1. One possible arrangement of the face identification components. In this configuration, images are captured on the handheld device, and computation is performed on a faster back-end server. Applications on any machine are then able to use the results.

software components; however, an API is provided to access the components using TCP sockets, which allows for control and computation to be arbitrarily distributed. Any application can connect to the image capture client to trigger face recognition. The face detection and recognition computation can then be performed either on the handheld or externally on a faster back-end server. Applications running on any machine on the network, including the server and handheld, can register to be notified on face detection events, and then use the results of the face recognition.

4 Face Identification Technology

The face detector used in this work [6, 7], developed at Compaq's Cambridge Research Laboratory, is trained via an algorithm that selects a small number of critical visual features to train extremely efficient classifiers. The classifiers are connected in a "cascade" configuration, which allows irrelevant features to be discarded very efficiently.

The face recognition component [8, 9], developed at the MIT Artificial Intelligence Laboratory and the Center for Biological and Computational Learning, is based on Support Vector Machine (SVM) classifiers [10].

4.1 Training the Face Recognizer

Training images are obtained by naive users using the handheld computer's camera. Users are instructed to attempt to capture images from various illumination and background settings, such that a larger range of lighting and background conditions is taken into account when training the SVMs. The application developer can hand-select the training images in order to eliminate bad data caused by incorrect face detection. The gray values of the training image pixels are converted into feature vectors, one 1600-dimensional vector per 40×40 image, for

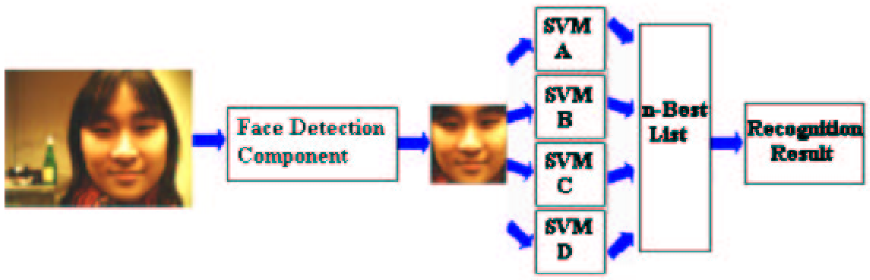

Fig. 2. The runtime face recognition process for four possible people in the training set. A face detector localizes and extracts the face. The pixel values of the extracted face are converted into a feature vector which is fed into an array of linear SVMs (A, B, C, and D). Each SVM returns a score representing the likelihood of the image matching the person that SVM has been trained for. An n-best list of the scores is returned.

training the SVM classifiers. These classifiers are based on the one-vs-all strategy (see [10]), in which each classifier is trained to distinguish one person from all the other people in the training set.

4.2 Runtime Face Recognition

In the process of runtime face recognition, the face detector is first used on the input image to extract the face. The face image size is then normalized to 40×40 pixels. As during training, the gray values of the image pixels are converted into a feature vector. The number of SVMs present is equal to the number of people in the training set, and each one distinguishes a single person from the rest. Each SVM is evaluated against the input image and produces a score representing the likelihood that the image matches the person for which the SVM was trained (distance from decision hyperplane). The SVM with the highest score is picked as the first choice for the identification, the next highest is the second choice, etc. (see Figure 2). Currently, there is only a very simple score-based rejection mechanism (based on raw score or on closeness to other scores). We are exploring other rejection methods, which may prove more robust.

5 Experiments

5.1 Face Recognition Accuracy

Comprehensive experiments gauging the accuracy of the system are yet to be completed. However, preliminary results have been encouraging. Figure 3 presents the accuracy of the face recognizer for various training set sizes. Accuracy figures are shown both for two data sets, with or without hand selection of images. The data were obtained using a training set of three people, and a test set of 15 images for each person, with no overlap between the training and testing data sets. The peak recognition accuracy was shown to be at 95.6% at 90 training images for the hand-selected data set, and at 84.4% at 45 training images for the raw data set. These data show that hand-screening training images yields a 6.2% relative average accuracy improvement, and a 13.2% relative maximum accuracy improvement.

5.2 Network Latency/Component Runtime

We have conducted several initial experiments to gauge the runtime of particular technology components and network latencies. The component configuration used for this experiment was that of Figure 1. The following table illustrates average runtime/latency figures over 100 test runs.

Task	Runtime/Latency(ms)
Send raw image over 802.11 networking	444
Convert image to grayscale	260
Run face detection	193
Total time excluding face recognition	897
Run face recognition (3 people/70 training images)	878
Run face recognition (4 people/70 training images)	1203
Run face recognition (5 people/70 training images)	1622

Sending a 160×120 color images captured by the handheld camera (57.8KB) over 802.11 networking took 444ms, and would take 380ms for 40×40 post-detection grayscale images (1.7KB). Thus, most of the latency associated with sending images over TCP sockets is due to the connection overhead, and moving face detection to the handheld would not yield a significant improvement in network latency. However, it may be possible to reduce overhead by maintaining a persistent socket connection. Additionally, capturing grayscale images in the camera would eliminate the need for an expensive (260ms) secondary conversion.

Our face detector code runs in 193ms on a 160×120 image; however [6] reports runtime to be 67ms on a larger 384×288 pixel image. This is most likely due to process setup overhead (since we fork off a separate face detection process for each image). This suggests that gains can be made by either using process pooling, or continually running face detection on the handheld and sending off face images when a detection is made (see Section 7.1).

The data show that runtime for one-vs-all SVM-based classification (see [10]) scales linearly with the number of people in the training set (for N people, $runtime \approx ((N + 1) \times 400)ms$). Thus, scalability is a significant issue for this implementation of the technology. For realtime performance, face recognition domains would probably be limited to 20 people or less (around 8 seconds) when using non-parallel desktop servers, and only a handful of people when running the recognition code on handheld computers. The memory footprint of the face recognition algorithm is $(3 + N \times 0.006)MB$ for N people. This scales to around 9MB for 1000 people, which is reasonable for handhelds.

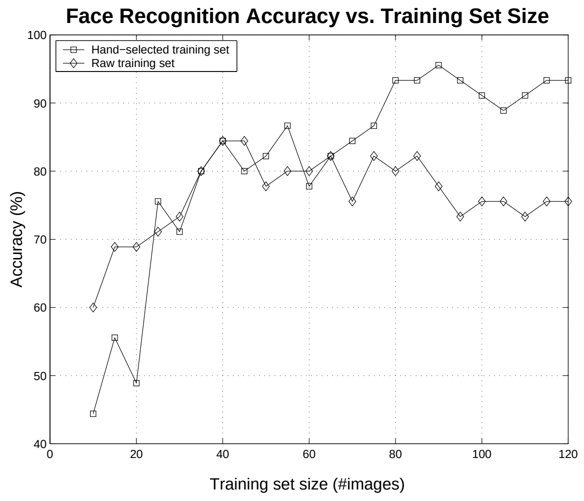

Fig. 3. Preliminary testing results showing the accuracy of the face recognition system when trained to recognize three people. Two data series are given, representing the accuracy with and without hand selection of training data.

6 Applications to Pervasive Computing

Since virtually any computer in an intelligent computing environment can benefit from the ability to identify its users, there is a vast number of applications for face identification technology within pervasive computing. Examples include:

– *Login:* The user is able to log into a computer using his or her face. This application is especially useful for handheld computers, since it is cumbersome and time-consuming to type logins on the small screens.
– *Face identification-based location services:* Face identification is combined with a location information system such as Cricket [11] to provide information about the position of people present in a user's environment.

7 Conclusion

In this work, we have designed and implemented a modular infrastructure for handheld face identification in the context of the Project Oxygen pervasive computing environment, which allows non-expert developers to create applications using this technology, and non-expert users to enroll in and use the applications.

7.1 Future Work

We are pursuing several strategies for improving face identification performance and usability. One such method is improving rejection performance, by using a generic SVM trained on pictures of people not in our positive database. Another method is to synthesize images based on a 3-D head model estimated from a much smaller training set (around 5 images) as described in [12]. We are also porting the face detection code to the handheld StrongARM processor, to make it possible to keep the face detector constantly running on the handheld, and trigger a face recognition process in the background when a face is detected. We intend to collect a much larger training data set in the near future, and much more complete test results should be available by the time of the conference. Lastly, we plan to develop a toolkit that allows the developer to control the entire application: image capture, training set screening, training, and runtime recognition, from a web interface similar in philosophy to [13].

Acknowledgements

We would like to thank the MIT AI Lab/CBCL for providing us with SVM-based face recognition code, and Compaq for supplying the face detection software. We would like to thank Jamey Hicks and Gita Sukthankar from Compaq for help with software and OS infrastructure. We would also like to thank David Dopson, Jim Glass, TJ Hazen, and Jean Hsu for proofreading this paper. This work was funded by an industry consortium supporting the MIT Oxygen Alliance.

References

1. "MIT Project Oxygen," http://oxygen.lcs.mit.edu/.
2. R. Chellappa, C. Wilson, and S. Sirohey, "Human and Machine Recognition of Faces: A Survey," in *Proc. IEEE*, 2001, vol. 83.
3. "The Familiar Project," http://familiar.handhelds.org/.
4. "Compaq Research Labs Project Mercury," http://crl.research.compaq.com/projects/mercury/.
5. "Delta Electronics, Inc.," http://www.delta.com.tw/.
6. P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," in *Proc. CVPR*, Kauai, Hawaii, 2001.
7. P. Viola and M. J. Jones, "Robust Real-time Object Detection," Tech. Rep. CRL 2001/01, Compaq Cambridge Research Laboratory, Cambridge, MA, 2001.
8. B. Heisele, P. Ho, and T. Poggio, "Face recognition with support vector machines: Global versus component-based approach," in *Proc. ICCV*, Vancouver, Canada, 2001, vol. 2, pp. 688–694.
9. P. Ho, "Rotation Invariant Real-time Face Detection and Recognition System," Tech. Rep. 2001-010, MIT Artificial Intelligence Laboratory, Cambridge, MA, 2001.
10. V. N. Vapnik, *The Nature of Statistical Learning Theory*, Spinger-Verlag, Berlin, Germany, 1995.
11. N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The Cricket Location-Support system," in *Proc. MobiCom*, Boston, MA, 2000.
12. V. Blanz and T. Vetter, "A Morphable Model for the Synthesis of 3D Faces," in *Proc. SIGGRAPH*, Los Angeles, CA, 1999.
13. J. Glass and E. Weinstein, "SpeechBuilder: Facilitating Spoken Dialogue System Development," in *Proc. Eurospeech*, Aalborg, Denmark, 2001.