FaceL: Facile Face Labeling

David S. Bolme, J. Ross Beveridge, and Bruce A. Draper

Colorado State University
Fort Collins, CO, USA
bolme@cs.colostate.edu

Abstract. FaceL is a simple and fun face recognition system that labels faces in live video from an iSight camera or webcam. FaceL presents a window with a few controls and annotations displayed over the live video feed. The annotations indicate detected faces, positions of eyes, and after training, the names of enrolled people. Enrollment is video based, capturing many images per person. FaceL does a good job of distinguishing between a small set of people in fairly uncontrolled settings and incorporates a novel incremental training capability. The system is very responsive, running at over 10 frames per second on modern hardware. FaceL is open source and can be downloaded from `http://pyvision.sourceforge.net/facel`.

1 Introduction

This paper presents Facile Face Labeler (FaceL), a simple open source tool meant to illustrate the essence of face recognition in less than five minutes. When FaceL is opened, it presents to the user a live video feed, face detection, and the ability to teach the system the identities of a few people. When trained, FaceL displays those identities back to the user over the video signal in real time. We call this annotation of video *Face Labeling*. The FaceL user interface is shown in Figure 1.

The primary motivation for FaceL is to provide real time demonstrations of face recognition performance. Colorado State University has been distributing an open source face identification evaluation system for over seven years[1]. This system runs from the command line to batch process face imagery to perform careful empirical evaluations of face recognition algorithm accuracy. This system is still available and widely downloaded, but it does not provide the experience of running a face recognition application with a camera and an observable result. FaceL fills that void.

FaceL also represents a timely coming together of technology. In recent years, webcams have become standard built-in features of most laptops and many desktop computers. OpenCV provides an implementation of the Viola and Jones cascade face detector [2], which is accurate and fast enough to detect faces in a live video feed. Robust real-time eye localization is provided by correlation filters learned through a new technique developed by us[3]. A support vector classifier is used to assign labels to faces [4]. Finally, `python` and the PyVision Computer Vision Toolkit[5] integrates the components into a cohesive user interface.

Most modern face recognition systems focus on the problem of identifying a novel (probe) face by finding the most similar image in a large database of face images. As such, most face recognition systems can be viewed as nearest-neighbor classifiers. In contrast, FaceL uses the Support Vector Classifier (SVC) to select the best of a small number of classes, typically one person per class. The support vector classifier appears to be very reliable for small numbers of people in uncontrolled environments [6]. In particular, it performs well under uncontrolled lighting with people moving their heads and adopting different expressions. However, one of the known issues with the SVC as

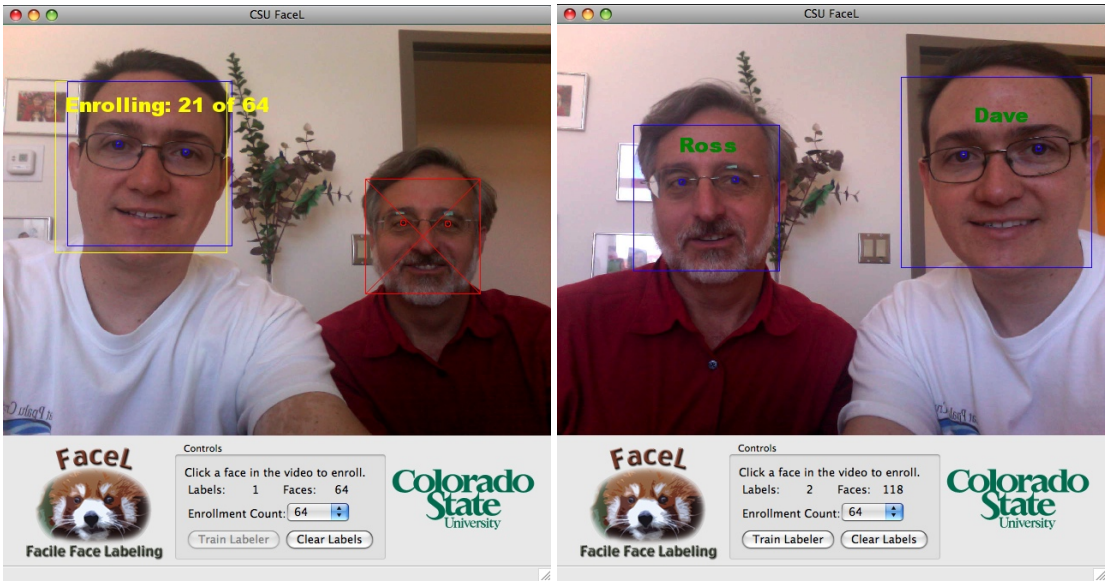

Fig. 1. FaceL User Interface. (Left) Clicking in a face detection rectangle enrolls the face. (Right) After training Foreheads are labeled.

it is used in FaceL is that performance degrades as the number of classes increases. In practice, we have not experimented with more than 8 people at a time.

Another distinct property of FaceL is that it can quickly enroll many images of a person from live video. In this way FaceL captures many different *views* of a person which include differences in expression and pose. Therefore, the system can produce an accurate model of faces under a variety of conditions, helping it to reliably label faces. A user can also add new images based upon observed performance, essentially helping FaceL to overcome mistakes.

Care has been taken to make FaceL easy to use and it is available both as a ready to run compiled application and as part of a larger open source library called PyVision (`http://pyvision.sourceforge.net`). The PyVision library is under a BSD license, which allows researchers and commercial vendors to integrate there own algorithms into the system.

2 Related Work

Face recognition has been an important and active field in computer vision research for many years. Multiple advancements in computer vision have come from the study of face recognition since Turk and Pentland first demonstrated the utility of eigenfaces [7] in the early nineties.

Modern commercial face recognition systems perform very well under controlled scenarios [6] and have demonstrated their utility for many applications including security and entertainment. For security, face recognition is now in use at banks, airports, government facilities, and border control. For entertainment, face recognition technology has found a home in personal photograph

applications including Facebook, YouTube.com, Google Picasa, Google Images, and Apple iPhoto. All these systems now perform some sort of face detection and recognition, annotating images with identifying information such as names and supporting identity-based image retrieval. This development is transforming the public perception of face recognition from a science fiction technology to a tool that is used in every day life.

While the tools just cited are useful, they do not provide a live experience. To fill that niche, FaceL is designed to provide an interactive face recognition experience that is accessible to the general public. Users of photo cataloging software may wonder why some faces are not detected in photos or why some faces are mislabeled. By providing a video interface to a face recognition algorithms, FaceL provides a dynamic and entertaining way to explore face recognition technology.

Prior to FaceL, we spent a significant amount of time developing a similar system called Scarecrow (a strawman for face recognition) [8]. Scarecrow was intended to be an educational tool that could quickly demonstrate the fundamentals of face recognition. Scarecrow had a simple Bayesian-based face detector[9] and an Eigenfaces-based face recognition algorithm. The system was written in C++ using the QT library for the user interface. Unfortunately, because Scarecrow used a simple 15 year old face recognition algorithm it could not accurately identify people unless the images were taken under very controlled conditions.

Scarecrow often produced inaccurate or some times even embarrassing results, it was therefore never released publicly. Many lessons were learned from Scarecrow that have made FaceL a success. Scarecrow illustrated that better registration was needed through the use of better face detection and the addition of a robust eye locator. The most notable change is that FaceL has abandoned the notion of a standard nearest neighbor face recognition algorithm, like Eigenfaces, which are designed to identify thousands of people. Instead FaceL trains an SVC to classify a face as belonging to a small number of labels.

We conducted a brief survey of free webcam-based face recognition demonstration systems and discuss four of these systems here, and a more complete list can be found on the FaceL website. Of these, most of the open source demonstrations are limited to face detection and are based on sample code provided with OpenCV [10] (the same detector used in FaceL). OpenCV is the only system that we were able to run on MacOS X, Linux, and Windows [1]. However, installing and running OpenCV requires significant software development skills and is therefore beyond the reach of many users.

We know of one other open source system similar to FaceL called Malic [11]. Malic captures video from a webcam and uses OpenCV for face detection and a Gabor filter based algorithm from the CSU Face Recognition Evaluation System [1] for identification. This project appears to have been abandoned in 2006. We spent a few hours trying to install Malic and its dependencies but were never able to get the system to run. This is the only other open source system that we know of that attempts face recognition in addition to face detection.

Pittsburgh Pattern Recognition [12] recently released a free demonstration application for their face detection and tracking software. Stand alone executables are available for Windows and Mac and are easily installed and run. The user interface for the demonstration is extremely simple, providing a video window with overlaid detection information and one control which selects the minimum face size. Their detector performs better under poor lighting conditions than OpenCV and detects face profiles and faces rotated up to 45°. Frame rates are approximately the same as FaceL.

[1] Most tests were run on a MacBook Pro running MacOS 10.5. Windows XP and Kubuntu Linux were tested on the same machine using Parallels and a Logitech QuickCam 5000 Pro USB webcam.

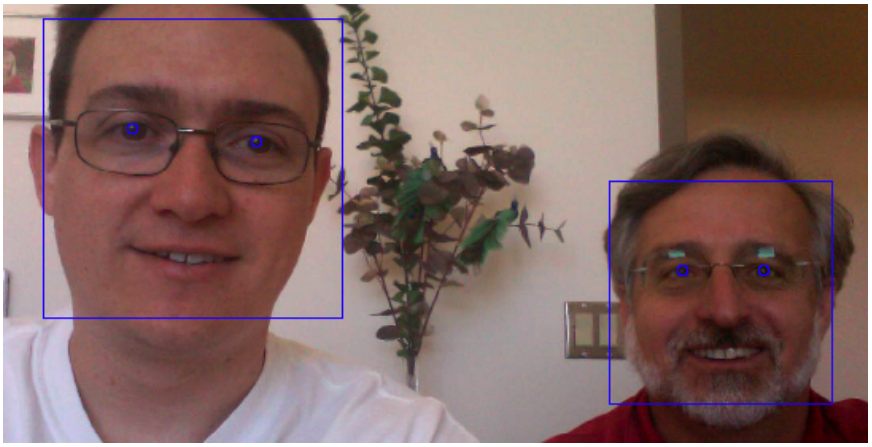

Fig. 2. Face detection (blue rectangles) and eye localization (blue dots) are displayed in real time in the video window.

NeuroTechnology has a free demonstration application for their VeriLook SDK[13] that performs face verification on Windows [2] The user interface is more complex than FaceL and appears targeted towards software developers. The primary differences between Verilook and FaceL is that VeriLook has a more traditional face recognition algorithm that outputs a similarity score. Verification is performed on the video by pressing a button labeled "M" for "Match". The system then captures a user defined number of frames and outputs a similarity score for any enrolled face that pass a threshold. The VeriLook face identification algorithm is impressive and only made three or four mistakes for the approximately 50 times we pressed the "match" button with 8 people enrolled.

3 Face Processing

There are three main components to the FaceL face labeling system: face detection, eye localization, and face labeling. FaceL processes a live video stream which means that these components must be fast. Because the system needs to operate using a variety of cameras in a variety of settings, they need to be robust to changes in lighting and background. They also need to be accurate enough to distinguish between a small number of face classes.

3.1 Face Detection: OpenCV Cascade Face Detector

Face detection is performed by the OpenCV cascade face detector, which is based on the Viola and Jones face detection algorithm[2], as extended by [14] (See Figure 2). OpenCV includes both the code and training files for the cascade classifier. The training files specify a decision tree which discriminates between face and non-face image subwindows. The classifier is easy to use and is

[2] Verilook did not run on Windows XP on Parallels, but did install and run without issues on a Windows Vista machine.

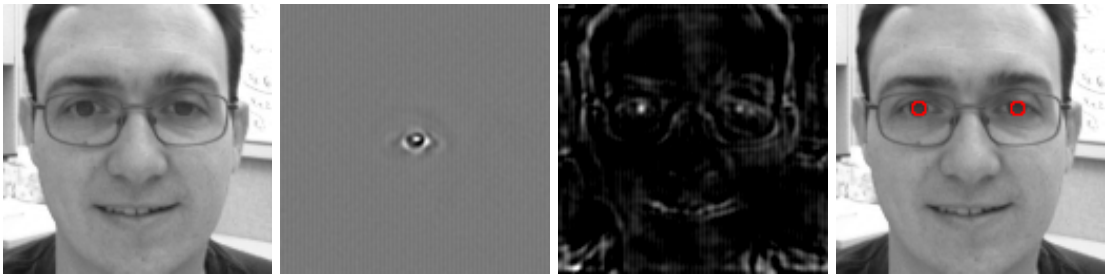

Fig. 3. Eye localization using ASEF Filters. From left to right: output from the face detector, the left eye ASEF filter, the correlation output from the left filter, and the image with the left and right correlation peaks marked in red. (The right eye filtering results are not shown.)

nicely suited for video processing. The output of the classifier is a set of rectangles which indicate face detections. We have found that the face detector performs very well detecting frontal faces under a variety of conditions and is also fast enough to run in real time [15].

The face detector uses several optimizations to improve run-time. First, the Haar-based features used for classification are computed from an integral image. This integral image allows each feature to be computed in constant time regardless of the size of the feature. The second optimization is the use of a cascaded decision tree. This is a decision tree that is built in layers where the purpose of each layer is to reject non-face subwindows. The idea is that most of the non-face subwindows are rejected quickly in the first few layers of the decision tree, which saves processing time while still allowing for high accuracy.

To further improve run-time, images are down sampled to 320×240 pixels before passing them to the cascade classifier. The minimum detection size was also set to 60 pixels. The result is that the minimum face size detected by the system is approximately 120 pixels across in the original 640×480 image. This setting is ideal for FaceL which expects the faces to be approximately 3 to 4 feet from the camera. However it also means that the system has difficultly detecting faces that are more than 5 feet from the camera. The tuned face detection algorithm takes between 15ms and 22ms to process a frame.

3.2 Eye Localization: Average of Synthetic Exact Filters

The second step in the system is to accurately register or align the faces. Faces are registered by locating the eyes and then rotating and scaling the faces such that the eyes are in fixed locations. We have found that accurate localization and subsequent registration of faces to a canonical size and position is critical for producing accurate labels.

Each eye is located by convolving the face with an Average of Synthetic Exact Filters (ASEF) filter (Figure 3) trained to detect eyes[3]. ASEF filters are simple correlation filters that are trained to produce sharp peaks corresponding to the eyes in the face image. The location of these peaks correspond to the location of the eyes.

ASEF filters are constructed from a set of training images and corresponding synthetic output. In this case the synthetic output is a bright peak at the location of the eye and zero every where else.

An *exact filter* is constructed which exactly maps a training image to the desired output. This exact filter is usually very specific to the particular training image. By averaging a few thousand exact filters, the features of the filter that are idiosyncratic are averaged out while features appearing in many of the exact filters remain. The resulting filters have the appearance of an eye and perform very well at locating eyes in face images.

Originally FaceL used the same filters that were produced using the FERET dataset as seen in[3]. We found that these filters performed well for frontal images but had some difficulty when presented with poses that were to the left or right of frontal. We believe this is because the FERET images are taken under controlled lighting with the person looking directly at the camera.

To correct this deficiency, we retrained the filters on the PIE Illum dataset which includes frontal poses and four neighboring poses (27,5,29,9,7) to expose the filters to broader variations of pose and lighting. The resulting filters perform much better in the settings where we have tested FaceL. Eyes are correctly located for most faces detected by the cascade classifier.

One advantage of using correlation filters to detect eyes is that correlation is very fast. The subwindows produced by the face detector are rescaled to 128×128. Those images are then correlated with two filters (one for each eye), using the FFT to speed up the computation. A real valued discrete FFT and inverse FFT from OpenCV are used because they require approximately half the time and memory as a full complex FFT. Eye detection has been benchmarked at under 3ms per face.

Correlation filters are also much faster than other common techniques for locating eyes. Gabor jet based methods often correlate the image with up to 80 different gabor filters which takes far more time than convolving with the two specially tuned filters of ASEF. Further more, ASEF filters have been benchmarked at almost three times faster than cascade classifiers trained to detect eyes.[3]

3.3 Labeling: Support Vector Classifier

Before classification some common preprocessing is done to the image to improve the accuracy of the classifier and reduce the time needed to classify the image. The first step uses the eye coordinates to geometrically align the face. A 128×160 pixel image of the face is constructed by translating, rotating, and scaling the original frame to place the eye coordinates at (26,40) and (102,40).

A high pass filter is applied to the image to remove some of the effects of lighting. This is done by converting the image to gray scale and convolving the face image with a Gaussian of radius $\sigma = 8.0$. This produces a smoothed image emphasizing the low frequencies. To obtain the higher frequencies the smoothed image is subtracted from the original face image. These images are then normalized to have a mean pixel value of zero and a standard deviation of one.

Principal Components Analysis (PCA) is used to perform dimensionality reduction. PCA training is performed on the enrolled images. The eigenvectors corresponding to 95% of the energy in the distribution are retained and define a subspace into which both enrolled images and new images are projected. The PCA basis is also used to whiten the data before classification. This takes the original distribution of the face images, which is assumed to be an elongated hyper ellipsoid, and non uniformly rescales it using the Eigenvalues such that the point cloud has unit variance in every direction. This transformation has been shown to have a positive effect on the accuracy of the Eigenfaces algorithm [1].

Finally, the system uses a SVC from `libsvm`[4] to classify the faces as belonging to a small number of user definable classes. A typical SVC finds a decision boundary that optimally separates two classes by maximizing a linear margin between those two point distributions. The `libsvm` library has extended this to a multiclass classifier by constructing a SVC for each possible pair of classes. A voting strategy is then used to assign the final classification.

One consequence of extending a two-class SVC to a multiclass problem is that labeling will only work well for a small number of classes. This is different from many face recognition algorithms which distinguish between hundreds of people by basing decisions on similarity computations between pairs of face images. FaceL therefore performs well only when distinguishing a small number of names, poses, or expressions.

In order to produce an accurate label, parameters of SVC needs to be tuned to produce good decisions. The PyVision SVC automatically tunes the classifier based on the method suggested by the authors of `libsvm`[16]. The enrolled images are first randomly partitioned. Two thirds are used as a training set and one third is used as a validation set. Tuning uses a Radial Basis Function (RBF) kernel to produce a nonlinear decision boundary and performs a grid search to find optimal values of C which is a penalty parameter for the error and γ which controls the radius of the RBF kernel.

4 Using the Application

FaceL can be downloaded from the PyVision website at `http://pyvision.sourceforge.net/facel`. A binary distribution is currently only available for MacOS 10.5. All libraries and components used by FaceL are cross platform so running FaceL on other systems should be possible. FaceL has been tested and runs on Linux from the source code distribution. On windows OpenCV has difficulty retrieving images from a webcam so additional work is needed.

FaceL requires a webcam for a live video feed. Video capture is delegated to the OpenCV library and therefore FaceL should support any webcam available to OpenCV. On Mac OS, FaceL has been tested with firewire and build-in iSight cameras as well as a few USB webcams from Microsoft and Logitech with no problems.

After starting FaceL, a window is displayed that contains live video feed from the webcam. Face detection and eye localization is run on each video frame. A blue rectangle is rendered around each face detection and small blue circles are rendered for each eye.

To enroll a face, the user simply clicks within the blue detection rectangle. FaceL will add the next 64 images of that face to its training dataset. After those images are collected the system prompts the user for a label such as the person's name. This step is repeated for each person enrolled in the database.

Once at least two distinct labels have been enrolled, the user trains the SVC by clicking the "Train Labeler" button. For a small dataset (128 faces), training takes less than 10 seconds. Training time increases with more enrolled faces. Once the SVC is trained the video resumes and faces in the video will have green labels on their foreheads. Additional faces can be enrolled at any time by using the same enrollment process and retraining the labeler. As will be discussed below, this incremental learning proves to be very useful.

4.1 Accuracy and Improving Recognition

Poor lighting conditions have the largest effect on the accuracy of FaceL. If the person's face is dark and the background is bright, face detection often fails. Because the face detector is pre-trained the best way to correct this problem is to set up the camera facing away from bright backgrounds or light sources such as windows. In extreme cases it also helps to use a lamp or other light source to illuminate the face.

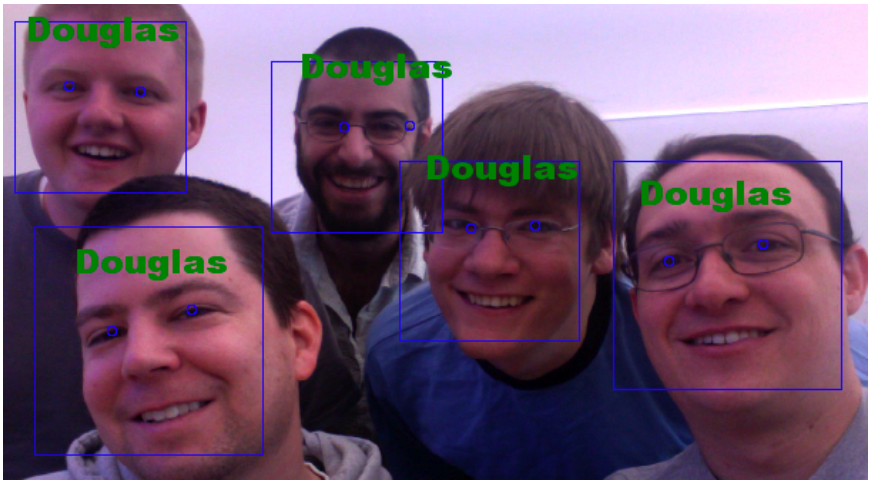

Fig. 4. Using live feedback at over 10fps users can quickly search out expressions or poses that confuse the labeler. Here the real "Douglas" is in the center of the image. In this case the users found that a smile in combination with some poses where associated with the label "Douglas".

Another cause of difficulty is uncooperative subjects. Users can attempt to confuse the labeler by finding a pose or expression that causes a face to be mislabeled. Because FaceL runs at over 10fps with immediate feedback to the user it is easy for users to quickly search for poses, expressions, lighting, etc. that cause a failure in labeling. Figure 4 shows one such example where the system learned to associate a smile with the label "Douglas" and hence all four people found ways to cause the system to label them "Douglas".

This amusing aspect of FaceL has two serious implications for fielded biometric systems. The first is that any system that allows a user to interact with an authentication system may be able to discover ways of gaming the system. In other words, they could experiment with changes until the system falsely associates them with another person's identity and unlocks whatever the system is endeavoring to protect. Hence, any system meant to provide security should probably avoid real-time feedback of the type illustrated by FaceL.

The second implication concerns the nature of enrollment. Traditional approaches to face biometrics have tended to take a rather narrow view of enrollment, often assuming only a single image is necessary. We have found that FaceL performs best when a variety of views are captured. Thus, users are encouraged to present as many views of their face as possible by rotating their head up, down, left, and right by up to 10° and also to assume different expressions. This goes a long way toward making FaceL more robust.

However, the "five Douglases" problem, where FaceL provides realtime feedback on the circumstances under which it becomes confused, opens the door for a staged and more intelligent approach to enrollment. So, if a user finds a view that causes difficulty, an additional video sequence of that view can be enrolled by clicking the face and entering the corrected label. When the system is retrained it will use the additional frames for training and the resulting classifier will no longer make the same mistakes.

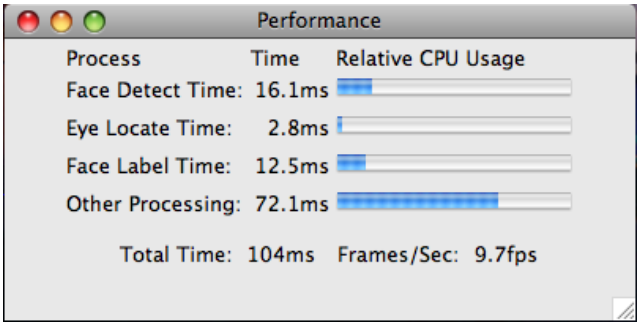

Fig. 5. Performance window measures the execution time of FaceL components for each frame processed.

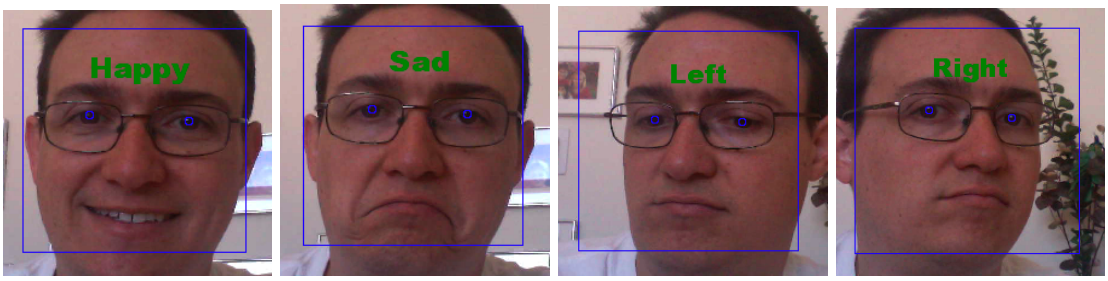

Fig. 6. FaceL can be trained to recognize expressions (left two images) or pose (right two images).

4.2 Real-time Labeling

The video in FaceL is very responsive. Processing times can be seen in Figure 5. We have found that FaceL runs at about 10 to 20 fps for a frame with one face. Tuning the face detector had the largest effect on the speed of the system. By sacrificing the ability to detect small faces the speed of face detection can be greatly improved. Eye localization time and face labeling time are of course proportional to the number of faces detected. Acquiring an image from the camera seems to be the most time consuming task in the image processing pipeline and could possibly be improved by interfacing directly with Quicktime instead of using the OpenCV abstraction.

4.3 Tasks Other Than Identification

Many face recognition systems are focused only on accurately identifying a person in an image. FaceL, however, has no such bias, and therefore can be trained to perform other tasks (See Figure 6). For example, the system does very well at detecting which direction a persons head is looking: “Left”, “Center”, or “Right”, or at distinguishing a small set of expressions: “Happy” or “Sad”. These are also some simple ways to test the system with only one person.

5 Conclusions and Future Work

In our experience over the past few months we have found FaceL to be an interesting and fun way to demonstrate face recognition technology. FaceL has a responsive and intuitive user interface that makes using the application easy. Finally, the OpenCV face detector, ASEF eye locator, and SVC labeling algorithms perform well with a small number of cooperative people.

Almost every person with a background in computer vision to whom we have demonstrated FaceL has said something to the effect: "Couldn't you do _____ with this!" As it stands, FaceL is serves to demonstrate real-time face recognition projected over a live video feed. In the future we intend to introduce features that make FaceL more useful as a face recognition tool such as the ability to detect faces, locate eyes, and label whole directories of images. which make FaceL more useful and more educational. We would also like FaceL to be more educational by exposing more of the inner workings of the algorithms.

Another key contribution of FaceL from the standpoint of biometrics is the compelling manner in which it opens up the notion of incremental and intelligent enrollment. Enrolling a large number of different views and additional enrollments for problem views has proven very useful in producing a robust classifier. Similar procedures may prove very useful in any biometric system where video enrollment and fast face processing algorithms are available.

Acknowledgments

We thank Yui Man Lui, Patrick Flynn, and other FaceL Beta testers for volunteering their time and for providing useful feedback. We also thank Ward Fisher and Jilmil Saraf for their work on Scarecrow and for the knowledge and experience that we gained from that system.

References

1. Bolme, D.S., Beveridge, J.R., Teixeira, M.L., Draper, B.A.: The csu face identification evaluation system: Its purpose, features and structure. In: Int. Conf. on Comp. Vision Systems. (2003)
2. Viola, P., Jones, M.J.: Robust real-time face detection. Int. J. Comput. Vision **57**(2) (2004) 137–154
3. Bolme, D.S., Draper, B.A., Beveridge, J.R.: Average of synthetic exact filters. In: CVPR. (2009)
4. Chang, C.C., Lin, C.J.: LIBSVM: a Library for Support Vector Machines. (2007)
5. Bolme, D.S.: Pyvision - computer vision toolbox. WWW Page: http://pyvision.sourceforge.net (2008)
6. Phillips, P., Scruggs, W., O'Toole, A., Flynn, P., Bowyer, K., Schott, C., Sharpe, M.: FRVT 2006 and ICE 2006 large-scale results. National Institute of Standards and Technology, NISTIR (2007)
7. Turk, M.A., Pentland, A.P.: Face recognition using eigenfaces. In: CVPR. (1991)
8. Fisher, W.: An introduction to and analysis of Scarecrow. Master's thesis, Colorado State Univ. (2008)
9. Saraf, J.: An assessment of alternative features for a semi-naive Bayesian face detector on single face images. Master's thesis, Colorado State University (2007)
10. Willow Garage: Opencv libarary (April 2009) `http://opencv.willowgarage.com`.
11. Suga, A.: Malic - malib with csufaceideval and opencv (January 2006) `http://malic.sourceforge.net`.
12. Pittsburgh Pattern Recognition: Webcam face tracker (April 2009) `http://demo.pittpatt.com/`.
13. NeuroTechnology: Verilook demo (April 2009) `http://www.neurotechnology.com/download.html`.
14. Lienhart, R., Maydt, J.: An extended set of haar-like features for rapid object detection. In: ICIP. (2002)
15. Beveridge, J., Alvarez, A., Saraf, J., Fisher, W., Flynn, P., Gentile, J.: Face Detection Algorithm and Feature Performance on FRGC 2.0 Imagery. In: Biometrics: Theory, Applications, and Systems. (2007)
16. Hsu, C.W., Chang, C.C., Lin, C.J.: A practical guide to support vector classification. LibSVM (2007)