Visage: A Face Interpretation Engine for Smartphone Applications

Xiaochao Yang[1], Chuang-Wen You[1], Hong Lu[2], Mu Lin[1], Nicholas D. Lane[3], and Andrew T. Campbell[1]

[1] Dartmouth College, 6211 Sudikoff Lab, Hanover, NH 03755, USA,
[2] Intel Lab, 2200 Mission College Blvd, Santa Clara, CA 95054, USA,
[3] Microsoft Research Asia, No. 5 Dan Ling St., Haidian District, Beijing, China
{Xiaochao.Yang,chuang-wen.you}@dartmouth.edu,hong.lu@intel.com,
mu.lin@dartmouth.edu,niclane@microsoft.com,campbell@cs.dartmouth.edu

Abstract. Smartphones represent powerful mobile computing devices enabling a wide variety of new applications and opportunities for human interaction, sensing and communications. Because smartphones come with front-facing cameras, it is now possible for users to interact and drive applications based on their facial responses to enable participatory and opportunistic face-aware applications. This paper presents the design, implementation and evaluation of a robust, real-time face interpretation engine for smartphones, called *Visage*, that enables a new class of face-aware applications for smartphones. Visage fuses data streams from the phone's front-facing camera and built-in motion sensors to infer, in an energy-efficient manner, the user's 3D head poses (i.e., the pitch, roll and yaw of user's heads with respect to the phone) and facial expressions (e.g., happy, sad, angry, etc.). Visage supports a set of novel sensing, tracking, and machine learning algorithms on the phone, which are specifically designed to deal with challenges presented by user mobility, varying phone contexts, and resource limitations. Results demonstrate that Visage is effective in different real-world scenarios. Furthermore, we developed two distinct proof-of-concept applications, Streetview+ and Mood Profiler driven by Visage.

Key words: face-aware mobile application, face interpretation engine

1 Introduction

Smartphones come with an increasing number of embedded sensors (e.g., microphone, accelerometer, gyroscope, etc.) that are enabling a new generation of sensing applications on the phones, such as, understanding user behaviors [21], life patterns, wellness, and environmental impact [22]. Similar to the microphone [17], the phone camera is also a ubiquitous sensor, which is typically used for recreational purposes such as taking pictures and videos. To date, the camera has not been exploited as a powerful sensing modality in its own right. This is about to change for the following three reasons. First, mobile phones have increasing sensing, computing, storage and communication capabilities. Next,

mobile phones have evolved into a multimedia platform capable of downloading a wide variety of applications from Google Play and the App Store. As a result, the way in which users interact with their phones has drastically changed over the past several years; that is, users are increasingly interacting with mobile applications (e.g., tweeting, web surfing, texting) directly through their phones' touch screens. Third, front-facing cameras are becoming a standard feature of new smartphones, allowing for new forms of interactions between the user and the phone, for example, the phone is capable of *observing* users as they interact with different mobile applications in everyday uses. A key contribution of Visage is that it supports a set of sensing, tracking and machine learning algorithms on smartphones, which are specifically designed to deal with the difficult challenges presented by user mobility, varying phone contexts and resource-limited phones (e.g., limited battery lifetime, computational resources).

Traditional human-computer interactions rely on buttons and clicks, which emphasize the explicit communication but neglect the implicit reactions from users. In the 1970s, Ekman [11] found that human emotions express distinctive content together with a unique facial expression. He found that there are 6 prototypical emotion categories that are universal across human ethnicities and cultures, namely happiness, sadness, fear, disgust, surprise and anger. To enable natural ways to interact with applications, previous researchers [30] focus on vision techniques that use fixed cameras (e.g., desktop cameras). While there is a considerable amount of work in computer vision on topics such as face tracking and expression classification, there has been very little work done on solving these problems on mobile phones. On the contrary, shifting from fixed indoor settings to mobile environments, our proposed system specifically deals with varying phone context and unpredictable phone movement. Since mobile phones (equipped with sophisticated sensing, computing, and communication capabilities) have become an indispensable part of our everyday lives, this creates an opportunity to develop a face interpretation engine that senses users' visual feedback and utilizes their face responses to develop new intuitive ways to interact with applications on mobile phones. For example, a user's head rotation angle can be used to infer what the user is paying attention to, such as, points of interest in the real world. The rotation angle information, for example, can be used to drive exploration of 360-degree Google StreetView panoramas.

Most smartphones are equipped with a front-facing camera, which is capable of capturing users' faces and expressions while they are interacting with their phones. In this paper, we present *Visage*, a robust, real-time face interpretation engine for smartphones that enables a new class of face-aware applications on the phone. Visage fuses data streams from the phone's front-facing camera and built-in motion sensors to infer, in an energy-efficient manner, the users' 3D head poses (i.e., the angle of pitch, roll and yaw of the user's head with respect to the phone's orientation) and facial expressions (e.g., happy, sad, angry, etc.). We explore how to make use of this new channel of information in a ubiquitous and nonintrusive way through the implementation of two proof-of-concept face-aware applications based on the Visage engine: 1) StreetView+, which exploits

user's head angle to drive exploration of 360-degree StreetView panoramas from the existing Google Map service; and 2) MoodProfiler, which opportunistically senses users' expressions and provides visualized summary of their moods as they use specific applications (e.g., email client and YouTube viewer) in their daily lives.

The contributions of this paper are three-fold:

- *Context-aware system framework*: We design new sensing, preprocessing, tracking and learning algorithms for efficient inference of users' facial expressions under varying mobility conditions. The Visage face detection algorithm uses gravity estimation provided by the built-in motion sensors to compensate misalignment of the phone's camera and user's face. The system detects phone shakiness to reduce noise and accumulated error to improve head tracking robustness. The Visage preprocessing algorithm uses the light information of the face region to automatically adjust the camera's exposure level to ensure sufficient image quality for tracking and inference algorithms.

- *Resource-aware management*: We study and identify the optimal resolution of the video stream from the front-facing camera to reduce the computation load on the phone while maintaining acceptable real-time inference accuracy. We adaptively tune workflow parameters and workflow depth for typical phone usage patterns to avoid unnecessary computational load.

- *Enabling face-aware applications*: We propose two proof-of-concept applications built on top of the Visage system: 1) StreetView+, which makes use of the head pose inference from the Visage pipeline and provides user navigation on-the-go; and 2) MoodProfiler, which opportunistically senses users' expressions and provides visualized summaries while users are interacting with specific applications (e.g., email client and YouTube viewer) in their daily lives.

The rest of this paper is structured as follows. Section 2 describes the design considerations for the Visage system. Section 3 introduces an overview of the system architecture, followed by a detailed description of the Visage sensing, preprocessing, tracking and inference algorithms. Section 4 presents the implementation of the Visage system, while Section 5 summarizes the evaluation and benchmark results of the system. Section 6 applies the Visage engine to build two proposed face-aware applications. Section 7 reviews related work. Section 8 offers concluding remarks.

2 Design Considerations

2.1 User Mobility

Mobile users hold and interact with their phones in an uncontrolled manner and often in totally unexpected ways, such as, holding their phones at different angles or moving them around while doing various activities. These are new challenges

not addressed by the computer vision community when designing face detection and tracking algorithms [26] or computing the facial expressions of users [20]. Existing algorithms [2, 24] are designed and optimized for laboratory settings, where the camera is mounted and stable, with an angle that does not change over time. Because the front camera on the phone is subject to all degrees of freedom, we design Visage to make no assumptions about the camera motion, tilt and rotation relative to the user's face. The movement of the phone causes the image quality to drop in comparison to a mounted camera. Poor image quality will hurt all stages of the vision processing pipeline. Rather than adopting more sophisticated image processing techniques to address this problem (which would be inappropriate for resource-limited mobile phones), we propose taking advantage of accelerometer and gyroscope sensors to detect the tilt and motion of the phone. This design approach is computationally light in terms of resource consumption particularly in comparison to adopting more heavy duty signal processing techniques. As discussed in Section 5.3, Visage uses an opportunistic reinitialization scheme based on multimodal sensing for tracking that reduces the error to acceptable levels.

Furthermore, people take their phones wherever they go, ranging from bright outdoor areas through dark clubs and restaurants. Typically, computer vision based tracking algorithms assume consistent lighting conditions [26]. However, the lighting conditions in mobile environments change. To address this issue, Visage analyzes the exposure level of the local face region in the image rather than considering the entire image as default. Visage adaptively controls the front camera on the phone by maintaining a good exposure level in the face region of the image.

2.2 Limited Phone Resources

Video processing pipelines are computationally costly in general. The front-facing camera will produce 800 times and 50 times more data than those produced by the accelerometer and the microphone, respectively, in terms of the unit-time size of the raw data stream on an iPhone 4 mobile phone. One possible solution might be to off-load the interpretation pipeline to the cloud [29] and send inference results back to the phone. However, this leads to transmission energy consumption costs and variable delays in processing that complicates the development of continuous real time face-aware applications. In addition, users would have significant privacy concerns off-loading live videos containing their faces to the network or the cloud. To address the privacy issues and communication costs, Visage exploits a phone-based approach which processes video streams in a real-time manner (i.e., videos are neither off-loaded to the cloud nor stored on the phone). Although the processor speed and memory size found on the latest smartphones are increasing rapidly, they are still considerably limited in comparison to desktop or laptop computers. Importantly, phones are energy constrained (compared to a desktop), and video processing is the most computationally intensive task on the phone. Computer vision researchers focus

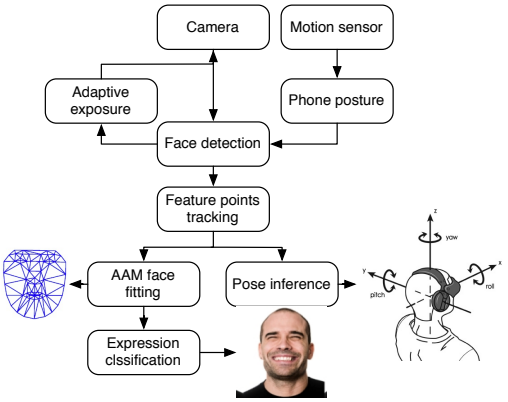


Fig. 1. Visage System Architecture

on the development of algorithms [23] that deal with time varying high dimensional image data with little energy, processor and memory constraints. Visage is designed to operate in real-time while also addressing these constraints.

3 System Architecture Overview and Design

The Visage system architecture shown in Fig. 1 comprises sensing, preprocessing, tracking and inference stages. The *sensing stage* captures the video stream from the phone's front-facing camera and raw motion data from accelerometer and gyro sensors on the phone. The *preprocessing stage* comprises of three components: (1) phone posture, (2) face detection, (3) adaptive exposure components. By adaptively correcting the shaky/tilted frames (based on orientation and shakiness given by the phone posture component) and adjusting the image exposure setting (based on the illumination of face region given by the adaptive exposure component), the face detection component locates the user's face in the view. Given a face detected in the preprocessing stage, the *tracking stage* initializes a cylinder head model and continuously tracks individual feature points within the detected face region in all frames. If excessive motion is detected, Visage reinitializes the tracking module. By tracking relative locations of the feature points marked on the face, the user's 3D head pose can be inferred. Finally, in the *inference stage*, a shape and appearance fitting method is used to extract the face texture and structure features for facial expression classification.

3.1 Preprocessing Stage

Phone Posture Component The preprocessing component identifies frames which contain the user's face, and monitors the phone posture (i.e., the tilt and motion status of the phone). Visage face detection and tracking are augmented

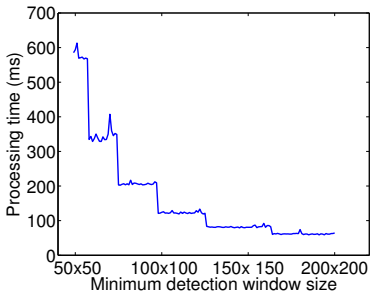


Fig. 2. Early termination scan scheme

with phone posture information to compensate the impact of the phone mobility. This module takes raw readings of accelerometer and gyroscope sensors and estimates the direction of gravity and the intensity of motion from the variance of those two sensor readings, respectively. The system calculates the mean and variance on each direction at the same frequency as the video frame rate. The mean of accelerometer data provides a smoothed estimate of gravity direction. The variances of accelerometer and gyroscope data indicate the phone's motion intensity. They are used by the face detection and tracking modules to ensure system robustness.

Face Detection with Tilt Compensation The process starts with locating the user's face in the first frame. This information is critical to the tracking task in consecutive frames. We use AdaBoost object detector [27] with tilt correction to make it robust to phone orientation changes, and to optimize the face-searching task by incorporating prior knowledge of the range of typical facial regions in images captured on mobile phones. The AdaBoost object detector operates by scanning the image using a large (e.g. 200×200 pixels) moving window. If no face is detected from a round of scans, the window size scales down by a constant factor of 1.2. The process terminates when a face is located, or scanning reaches the minimum window size. The algorithm normally starts from the largest possible detection window and scales down at each round by a constant factor to the smallest detection window, e.g. 20 pixels. In this paper, however, window sizes that are too small were omitted because the distance between the phone and the user's face is normally constrained by the user's arm length during mobile phone usage. This provides an estimate of the lower bound of the size of the user's face in the image, enabling us to prune scans with windows sizes that are too small. In our experiments, this resulted in an early termination scan scheme and higher system responsiveness. As illustrated in Fig. 2, on a 320×240 image, a 128×128 window size lower bound results in a detection time of about 80 ms, which gives more than 10 frames per second.

The standard AdaBoost object detector algorithm assumes that faces in the images are in an upright position, which is not necessarily true for phone cameras when a user holds the phone in a non-upright way. To work with possibly

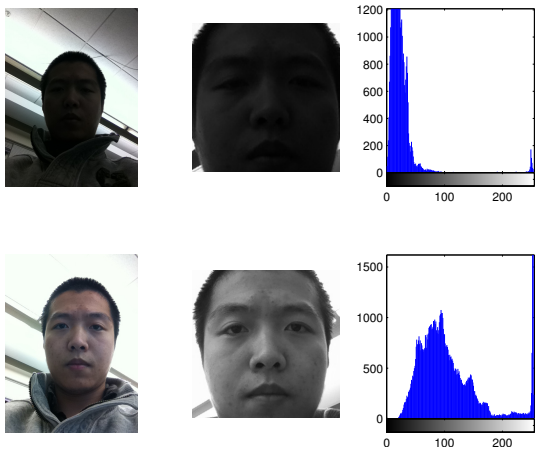


Fig. 3. Top: underexposed image, face region, and regional histogram; bottom: the image after adaptive exposure adjustment, face region, and regional histogram

tilted faces, two techniques are normally used: i) designing a set of classifiers handling each specific angle of tilted faces, or ii) using upright face detector on every possible angles. However, detectors trained for tilted faces are generally not as discriminative as the ones trained specifically for upright faces, and multiple detections incur more computational overhead. Thus neither technique is viable on mobile phones. Our solution leverages multi-modality sensing. With the input of accelerometer, Visage infers how the phone is tilted by estimating the gravity direction. The phone posture component calculates the gravity direction projected onto the coordinate system of the camera and feeds it to the face detection component. It is derived from smoothed accelerometer samples (i.e., a_x and a_y) of x and y axis on the phone by:

$$\theta_g = \frac{180}{\pi} \arctan \frac{a_x}{a_y} \tag{1}$$

Then the image is rotated by:

$$I_r = \begin{bmatrix} \cos\theta_g & -\sin\theta_g \\ \sin\theta_g & \cos\theta_g \end{bmatrix} I_i \tag{2}$$

where I_i and I_r are the matrices of original and corrected images, respectively. First, the tilt of the camera is compensated by the correction rotation. Then, the AdaBoost object detector is applied on the rotated image. The estimated gravity vector may include noise introduced by the user's movement. The AdaBoost object detector algorithm, however, has a tolerance window from about -15 to 15 degrees. It tolerates the rotation noise caused by the error of the estimated gravity direction.

Adaptive Exposure Component In mobile environments, illumination is not always guaranteed and varies frequently. A clear face region is critical for tracking and inference. Proper exposures under dynamic illumination conditions play a key role. The default automatic exposure adjustment balances the exposure of the entire image, which results in underexposed faces in the captured images, as demonstrated in Fig. 3. When underexposed, certain details in the face region are lost, making the head pose tracking and facial expression classification more difficult. To address this issue, Visage uses the local lighting information within the detected face region to correct the camera hardware exposure level. After a face is detected, we calculate the local histogram H_{face} of the face window, and then determine its exposure level by computing the centroid of H_{face}:

$$C_{H_{face}} = \frac{\sum_{i=0}^{255} iH_{face}(i)}{\sum_{i=0}^{255} i} \tag{3}$$

where i is the intensity bins of the histogram. $C_{H_{face}}$ lying in the lower or higher ends of H_{face} would indicate the face being under- or over-exposed. This information is sent to the sensing module to adjust the camera exposure setting iteratively, until $C_{H_{face}}$ is in the center area of the histogram. After this step, the tracking and inference start.

3.2 Tracking Stage

Feature Points Tracking Component After locating the bounding box of the user's face and setting the proper exposure level, this component initiates face-tracking algorithms using inter-frame information and estimates the 3D pose of the head. Feature Points Tracking Component first selects candidate feature points in the facial region from the first frame, and then tracks their locations in subsequent frames. Feature points represent landmarks on the facial object. Their spatial relationship to each other on the face is used to estimate the rotation of the user's head. Selecting proper feature points to track is the first step. Compared to points in textureless patches, points on object corners and edges (e.g., eye corners and edges of mouths) are normally selected as feature points because their salient visual features are stable across frames, and therefore, useful for tracking tasks. The level of textureness of a point is associated with the 2×2 autocorrelation matrix H of the second derivative image over a neighborhood W. If we let I_x and I_y be the gradients of the vertical and horizontal directions, respectively, then we have

$$H = \sum_{(x,y)\in W} \omega_{x,y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{4}$$

where $\omega_{x,y}$ is a Gaussian weighting term.

The smallest eigenvalues of H are sorted; the largest of these smallest eigenvalues correspond to feature points on corners and edges that can be tracked easily. We used the Lucas-Kanade method [3] (LK) to track the movement of

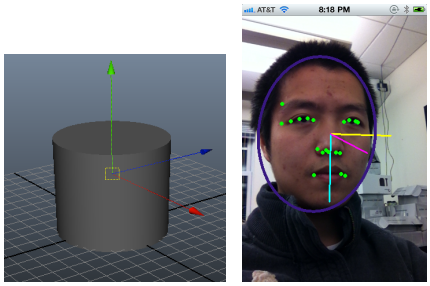


Fig. 4. (a) Cylinder model and (b) Tracking with pose estimation.

the selected points in successive frames. LK assumes brightness constancy, that is, pixels in a tracking patch look the same over time.

$$I(x(t), t) = I(x(t + dt), t + dt) \tag{5}$$

where I is an image, t is time variable, and x is the pixel location. Assuming that a pixel's neighborhood has the same motion vector as the pixel itself, this vector (i.e., $[u\ v]^T$) can be expressed as:

$$\sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \sum_{(x,y)\in W} \begin{bmatrix} I_x I_t \\ I_y I_t \end{bmatrix} \tag{6}$$

where I_t is the derivative between images over time. Fitting the neighborhood to a least square estimator solves the motion vector ($[u\ v]^T$). In practice, a multilevel adaptive scheme is used to incorporate large motions between frames [3]. Feature points tracking provides precise details on the motion of the face, but the error of individual tracking points accumulates over time. This is a common problem of the LK method. Therefore, by using a more robust blob-tracking algorithm called CAMSHIFT [6], the tracking points are corrected if the phone motion is larger than the existing threshold. In CAMSHIFT, after the facial region in the bounding box is transformed from RGB space to HSV space, a histogram of the hue channel is calculated as a template. In the incoming video frames, a probability of face blob is calculated at every pixel by examining the counts in the histogram template at the location of the pixel's hue. The algorithm estimates the new location of the face by searching the region with maxima on the face probability image, and shifts the tracking blob to that location.

Pose Estimation Component The Pose from Orthography and Scaling with ITerations (POSIT) algorithm [10] is used to estimate the 3D pose of the user's head. If an object's exact dimensions are known, POSIT is able to recover the six degrees of freedom movement of the object, with coordinates of at least four non-coplanar points on its 2D surface image and corresponding points describing its 3D structure.

To reconstruct the 3D pose, the algorithm requires the location of the 2D feature points on the image and their corresponding positions on the 3D head. It

is infeasible to pinpoint the 3D locations of the feature points on the user's face; thus, the human head is simplified to a rigid cylinder, as shown in Fig. 4. The diameter of the cylinder was set to the unit length. It is then straightforward to determine the 3D coordinates (X, Y, Z) of a 2D image point (x, y) on the initial frame. Because the initial frame contains a front face, coordinates in horizontal and vertical directions X and Y can be estimated directly by scaling image coordinates x and y, respectively; depth Z is obtained from the shape of the cylinder.

$$Z = \sqrt{r^2 - (X - r)^2} \tag{7}$$

After initialization, the 3D geometry of the points on the cylinder model is obtained. As the head moves in successive frames, the 2D feature points are tracked by LK. Using a rotation matrix and a translation vector to match the 3D cylinder points to the 2D image points, POSIT determines the directions for rotating the cylinder. Thus, the 3D pose of the cylinder is recovered, and the head pose is determined. POSIT requires at least four non-coplanar points; our study generated at least eight points.

The simplified cylinder face model introduces modeling errors because the surface of a face is different from that of a cylinder. Visage compensates for the modeling errors by a linear regression-based model calibration on all three rotation angles. On a given rotation angle, the calibration parameter is determined by

$$P_{calib} = (\Theta_{raw}^T \Theta_{raw})^{-1} \Theta_{gt} \tag{8}$$

where Θ_{raw} is a vector of raw output rotation angles, and Θ_{gt} is the corresponding ground truth angle. With calibration, the output angle in the inference step becomes:

$$\theta_{calib} = P_{calib} \begin{bmatrix} \theta_{raw} & 1 \end{bmatrix}^T \tag{9}$$

The inferred head pose can be used independently for subsequent components (or applications such as Streetview+) to detect when to trigger the eye region detection component.

3.3 Inference Stage

Active Appearance Model To save computation, the following steps happen only when face orientation is determined as near frontal. The face features for classification in the system are representing both structure and texture of the user's face. They are obtained by Active Appearance Models [19] (AAM), a statistical method which incorporates texture and shape variations into the model to improve the accuracy. It describes the shape of a 2-dimensional image by a triangular mesh consisting of a set of landmark points. Active Appearance Models require a set of training images with a set of predefined shape-critical feature points. In particular, human facial expression recognition requires accurate feature points on human faces, such as edges and contours of eyes and the mouth. The shape of these feature points is the major component of facial expression.

The texture model captures pixel color intensities to enhance the accuracy of model fitting.

By simultaneously considering the shape ($\mathbf{x}$) and texture ($\mathbf{g}$) of users' faces, an active appearance model can be synthetically determined with a combination of shape and texture models:

$$\begin{aligned} \mathbf{x} &= x_0 + Q_g c \\ \mathbf{g} &= g_0 + Q_s c \end{aligned} \tag{10}$$

where x_0 is the mean shape, g_0 is the mean texture, and Q_s and Q_g are variation matrices of shape and texture models, respectively. Instead of separating shape parameters from texture parameters, the combined model has a synthesized parameter (i.e., c) to describe the appearance of a human face. The training and fitting process is described as follows: (1) face images with predefined 20 landmarks are loaded from the face database to construct shape and texture vectors accordingly. (2) Principal Component Analysis (PCA) is then performed on normalized shape vectors and texture vectors.

Expression Classification Visage uses both geometric and appearance features for facial expression classification as shown in Fig. 5. The combination of these two features makes Visage more robust to complex mobile environments. Visage recognizes seven expression classes: angry, disgust, fear, happy, neutral, sad, and surprise.

The face feature vector is constructed from the relative location of the 20 landmarks and the texture intensity within the tracking rectangle. We use a facial analysis technique called Fisher Linear Discriminant Analysis (Fisherface) [5] for the classification task. Fisherface seeks a projection direction (P) that maximizes the determinant of the between-class scatter matrix and minimizes that of the within-class scatter matrix:

$$P_{opt} = \arg\max_P \frac{|P^T S_B P|}{|P^T S_W P|} \tag{11}$$

where S_B is the between-class scatter matrix, and S_W is the within-class scatter matrix, respectively. S_B captures the variations between different facial expression classes, while S_W represents the variance within a given expression class, including different person identities, various lighting conditions, etc. The problems could be solved by a generalized eigenvalue solver.

$$S_B P = \lambda S_W P \tag{12}$$

where P is the projection matrix. All face feature vectors (i.e., I_{face}) are projected to this expression-specific subspace (i.e., v_{exp}) by

$$v_{exp} = P I_{face} \tag{13}$$

Given the predefined 7 classes of facial expressions, the size of the feature vector is reduced to 6 by the projection. The projected feature vectors are fed to a SVM classifier trained by LibSVM [8] in nu-SVC mode, with a RBF kernel and the nu parameter set to be 0.4.

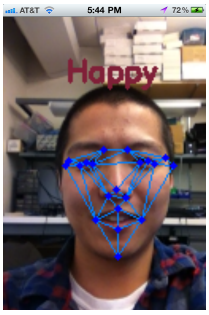


Fig. 5. Visage expression classification on the iPhone 4

4 Implementation

We prototyped the Visage architecture and algorithms on an Apple iPhone 4. Objective C is used to handle the GUI and hardware API. The core processing and inference routines are implemented in C. The OpenCV library [28] is ported to iOS as a static library, on top of which we built the Visage pipelines. The AAM fitting algorithm is ported and modified from VOSM [14] projects.

Standard picture resolutions (e.g., VGA (640×480) or higher resolutions) can be easily handled by desktop computers in real-time, but not mobile phones. Image size is an important factor impacting the overall computational cost, as demonstrated in Table 1. For phone-based systems, downsampling images to a lower resolution and skipping frames are inevitable. In theory, information loss caused by downsampling would degrade the performance. However, in practice, a phone's front-facing camera is usually positioned close to the user's face. Therefore, the size of the face regions captured by the phone's camera is much larger than those captured by distant cameras used in the surveillance or desktop cases. Therefore, downsampling introduces a much smaller performance penalty in our case. Studies show that face images smaller than 64×64 [9] lead to noticeable performance drops for expression classification tasks. To balance the computational load and the performance, Visage uses 192×144 as the operating resolution, which normally contains faces of size around 64×64. At the same time, Visage

Resolution	Time (ms)
640×480	4090
480×360	2123
320×240	868
192×144	298
160×120	203
96×72	68
80×60	53

Table 1. Computational cost of face detection operations under different input image resolutions

Tasks	Avg. CPU usage	Avg. memory usage
GUI only	< 1%	3.18MB
Pose estimation	58%	6.07MB
Expression inference	29%	4.57MB
Pose estimation & expression inference	68%	6.28MB

Table 2. CPU and memory usage under various task benchmarks

also uses a frame skipping scheme where if the processing cannot keep up with the incoming frame rate, oldest unprocessed frames will be dropped. By doing this, the inference output is alway synchronized with the latest scene in the view range of the camera.

5 Evaluation

5.1 CPU and Memory Benchmarks

We evaluate the Visage system on the Apple iPhone 4 and present the detailed performance of each component of the multi-stage inference process.

Table 2 presents results of the CPU and memory usage under different face analysis tasks (i.e., benchmarks), while Table 3 shows the processing time of all Visage pipeline components. The face detection module is computationally expensive. But, after the first frame containing a face is identified by the detection module, Visage will track the detected face in consecutive frames without performing detection operations. When both detection and tracking modules are enabled, it takes approximately 53 ms to detect and track the face in a frame on average. The CPU usage for the iPhone 4 is about 68%, when the full pipeline is engaged.

Component	Average processing time(ms)
Face detection	53
Feature points tracking	32
AAM fitting	92
Facial expression classification	3

Table 3. Processing time benchmarks

5.2 Tilted Face Detection

To test the effectiveness of the accelerometer-augmented phone posture correction, we conduct the following experiment. Let the user hold the phone in his/her hand and rotate it during the face detection algorithms running on the phone. Images are captured when the phone is tilted by different degrees, as illustrated in

Fig. 6. The experimental results from two schemes (i.e., the default AdaBoost algorithm and proposed Visage's Face Detection with Tilt Compensation scheme) are reported with the rotation angles ranging from $-90° \sim 90°$, separated by an angle of 10° degrees. Detected faces are marked by red bounding boxes. The default algorithm fails to locate the user's face when the tilted angle is beyond the $[-15°, 15°]$ range. The proposed scheme is robust to the various degrees of tilted angles, far beyond the $[-15°, 15°]$ tolerance range of the default scheme.

5.3 Motion Based Reinitialization

The performance of feature points tracking is sensitive to camera movement. To test the benefit of our proposed motion based reinitialization, we conduct a field study and monitor the cumulative head tracking error of the feature points tracking with and without motion based reinitialization. In this experiment, we focus on the error caused by phone movements, therefore we keep the head still in an initial pose, in which case the ground truth is zero for all three rotation angles (i.e., yaw, roll and pitch). The estimated rotation angles of the head pose are recorded as well as the phone's motion intensity represented by normalized variance of accelerometer readings. Experimental results are shown in Fig. 7. Without motion-based reinitialization, after the phone stops moving, the head pose estimation drifts on all three rotation angles, and the estimation error accumulates over successive movements. With Visage's opportunistic reinitialization, the system automatically corrects the feature points tracking and cylinder model reconstruction errors whenever motion of the phone is detected. The tracking error is thus suppressed and does not accumulate over time.

5.4 Accuracy of Head Pose Estimation

In this first experiment, several evenly-spaced markers are placed along a circle of one-meter radius. Three volunteers participate in this experiment. Each of them is asked to stand at the origin of the circle and look at each of the markers. Ground-truth head rotation angles are measured by a protractor at the origin of the circle. Each participant contributes 5 samples for every marker. The head pose estimations without model calibration, calibrated readings with

Fig. 6. Images captured by the front-facing camera assuming varying phone tilted angles from -90 $\sim$ 90 degrees, separated by an angle of 15 degrees. The red boxes indicate the detection results. The first row is detected by the standard Adaboost face detector. The second row is detected by Visage's detector.

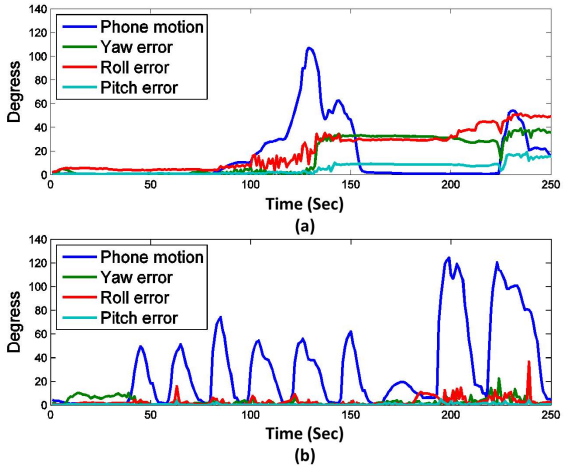


Fig. 7. Phone motion and head pose estimation errors (a) without motion-based reinitialization, and (b) with motion-based reinitialization

model calibration, and the ground truth are shown in Fig. 8. With model calibration, the value of the mean absolute error drops by 60%, from $14.48° \pm 2.67°$ to $5.51° \pm 1.99°$.

5.5 Accuracy of Facial Expression Classification

To evaluate the accuracy of facial expression classification, we test Visage on two facial expression datasets (i.e., a public facial expression dataset and the other dataset collected by ourselves). In this experiment, five volunteers participate. Each one is asked to perform a list of predefined expressions. We capture 100

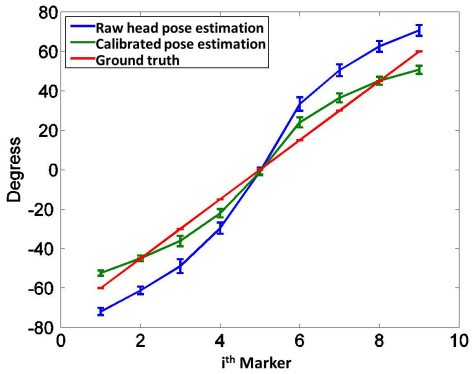


Fig. 8. Head pose estimation error

Expressions	Anger	Disgust	Fear	Happy	Neutral	Sadness	Surprise
Accuracy(%)	82.16	79.68	83.57	90.30	89.93	73.24	87.52

Table 4. Facial expression classification accuracy using the JAFFE dataset

consecutive frames of each expression and use 5-fold cross validation. This experiment assumes that the facial expression on a mobile phone is personalized to its owner. So, for each user, the model is trained and tested by using the user's own training data. Table 4 shows the average expression recognition accuracy, with an overall accuracy of 83.78%.

To verify our method in more general cases, we test Visage's algorithm on the JAFFE [18] public expression dataset. JAFFE contains 10 subjects with 7 different expressions as in Table 5. Each subject contributes $3 \sim 4$ examples per expression. We carry out 10-fold cross validation on the entire dataset. The evaluation is repeated 10 times. Table 5 shows the confusion matrix of the classification results. The overall classification accuracy is 84.6%, which is comparable to 81% reported in [25] and 85.6% in [15]. Note that our framework automatically detects faces, whereas [25, 15] used manually labeled face images.

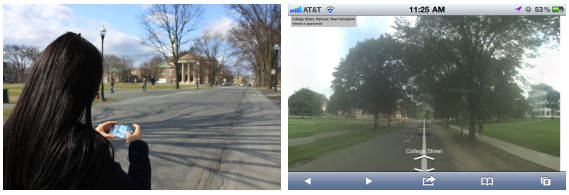
6 Visage applications

6.1 Streetview+

Streetview+ is a demo application built using the underlying Visage software engine. In this application, the user's head rotation with respect to the screen is tracked and used as an input source for users. Given the user's current location (i.e., longitude and latitude) obtained from the GPS sensor, the application shows the 360-degree panorama view with respect to that location from Google Streetview. The user's 3D head rotation is then used to change the viewing angle of the panorama as showed in Fig. 9. In practice, Visage achieves a tracking rate of around $12 \sim 15$ frames per second. Streetview+ provides an intuitive and realistic viewing experience in the virtual world, and the user can navigate the Google Streetview smoothly with continuous head movements.

Expressions	Anger	Disgust	Fear	Happy	Neutral	Sadness	Surprise
Anger	93.33	6.67	0	0	0	0	0
Disgust	6.90	75.86	17.24	0	0	0	0
Fear	0	7.41	92.54	0	0	0	3.23
Happy	0	0	0	87.10	6.45	3.23	0
Neutral	0	0	0	0	90.00	10.00	0
Sadness	0	6.45	9.68	3.23	9.68	70.97	0
Surprise	0	0	3.33	3.33	0	0	93.33

Table 5. The confusion matrix of the facial expression classification based on the JAFFE dataset



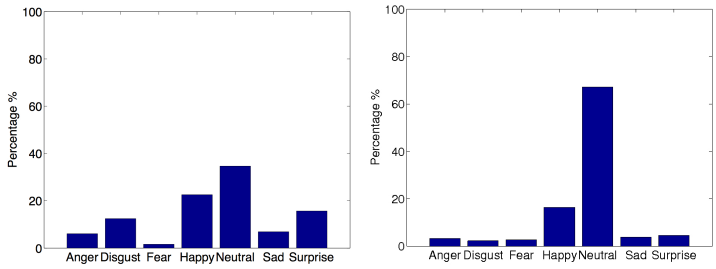
(a) Streetview+ on the go (b) Head facing front

(c) Head facing left (d) Head facing right

Fig. 9. Steetview+ enhanced with awareness of user head rotation

6.2 Mood Profiler

The mood profiler application uses Visage to profile a user's mood in the background as they interact with different applications on their phone. When a user interacts with the phone, their realtime facial expression is monitored as well as the name and type of the foreground application. To reduce the resource usage, the emotion classification is performed only once per second. Fig. 10 shows the expression histograms of a single user using two different applications, the YouTube mobile app and the Email client. The histogram shows the subject's mood when using these two applications. In the case of watching YouTube videos, the subject manifests a wider range of expressions including happy, surprise, sad and disgust, while in the case of reading emails, the neutral expression is the dominant class.



(a) YouTube (b) Email

Fig. 10. Expression histogram when using (a) the YouTube mobile application and (b) an email client.

7 Related Work

There is a growing interest in applying computer vision algorithms in support of mobile applications [7]. SenseCam [12] is a life logging application. It takes pictures of the user's everyday life. However, it involves very limited image processing. MoVi [4] is a collaborative application where users collectively send images to backend servers to mine common interests. On the mobile side, Recognizr [1] is an application using the back camera to recognize objects. It connects the object's real identity with its virtual identity on the web. Recently, several content-based image retrieval mobile frameworks (e.g., Google Goggles) emerged. They usually require the user to take a photo of an object and then send the image or feature vector to a remote server for further processing. The mobile phone is mainly used as a camera and the communication client with very limited local image processing. In contrast to these approaches Visage processes all the information locally on the phone - enabling, for example, head rotation as a novel UI for mobile phones.

In the HCI domain there is research that integrates the user's face into the mobile UI. PEYE [13] performs simple tracking of 2D face representations from the captured images; however, UI controls are limited to 2D directions (i.e., left, right, up, down) on the phone screen. Visage is capable of tracking the user's 3D head pose in a real-world coordinate system.

There is a considerable amount of research on head pose estimation [31] and facial expression analysis [31]. However, this work does not addresses challenges specific to mobile environments, e.g., camera motion, uncontrolled context, and computation efficiency. Much of the work achieved robustness by increasing the feature vector dimensions, finding more complex features. These purely vision-based approaches are typically computationally-demanding and not suitable for mobile phone applications. Littlewort *et al.* [16] proposed an expression classification algorithm for robots, however, the feature vector is based on pure texture instead of structure.

8 Conclusion

In this paper, we propose Visage, a face analysis engine specifically designed for resource limited mobile phones. Visage carries out all the sensing and classification tasks directly on the mobile phone without the need for backend server resources. In contrast to traditional mobile image analysis systems that rely on remote servers, Visage performs online processing at a lower computational cost but yields results that are comparable to existing off-line systems. Furthermore, multi-modality sensing is used to boost the system robustness in mobile environments. We present two proof-of-concept applications, and believe that the flexibility and robustness of Visage make it suitable for a wide range of face-aware applications.

References

[1] Recognizr, http://news.cnet.com/8301-137723-10458736-52.html

[2] Adiv, G.: Determining Three-dimensional Motion and Structure from Optical Flow Generated by Several Moving Objects. In: Trans. Pattern Anal. Mach. Intell., 7(4), pp. 384-401 (1985)

[3] Baker, S., Matthews, I.: Lucas-kanade 20 Years On: A Unifying Framework. In: Int'l J. Comput. Vision, 56(3),pp. 221-255 (2004)

[4] Bao, X., Choudhury, R.R.: MoVi: Mobile Phone based Video Highlights via Collaborative Sensing. In: Proc. the 8th int'l conf. Mobile systems, applications, and services, pp. 357-370. ACM, New York (2010)

[5] Belhumeur, P.N., Hespanha, J.P., Kriegman, D.J.: Eigenfaces vs. Fisherfaces: Recognition using Class Specific Linear Projection. In: Trans. Pattern Anal. Mach. Intell., 19(7), pp. 711-720 (1997)

[6] Bradski, G.R.: Real Time Face and Object Tracking as a Component of a Perceptual User Interface. In: Proc. the 4th IEEE Workshop on Applications of Computer Vision, pp. 214–219, IEEE Computer Society, Washington, DC (1998)

[7] Chai, S.: Mobile Challenges for Embedded Computer Vision. In: Embedded Computer Vision, Advances in Pattern Recognition, pp. 219–235, Springer-Verlag London (2009)

[8] Chang, C.-C., Lin, C.-J.: LIBSVM: A Library for Support Vector Machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27, (2011)

[9] Cunningham, D.W., Nusseck, M., Wallraven, C., Bulthoff, H.H.: The Role of Image Size in the Recognition of Conversational Facial Expressions. Research Articles. Comput. Animat. Virtual Worlds 15, 3-4, pp. 305–310 (2004)

[10] Dementhon, D.F., Davis, L.S.: Model-based Object Pose in 25 Lines of Code. In: Int'l J. Comput. Vision 15, 1-2, pp. 123–141 (1995)

[11] Ekman, P., Friesen, W.V.: Constants Across Cultures in the Face and Emotion. In: Journal of Personality and Social Psychology, 17(2), pp. 124-129 (1971)

[12] Hodges, S., Williams, L., Berry, E., Izadi, S., Srinivasan, J., Butler, A., Smyth, G., Kapur, N., Wood, K.: SenseCam: A Retrospective Memory Aid. In: Proc. the int'l conf. Ubiquitous Computing, pp. 177–193, Springer-Verlag, Berlin (2006)

[13] Hua, G., Yang, T., Vasireddy, S.: PEYE: Toward a Visual Motion Based Perceptual Interface for Mobile Devices. In: Proc. of the 2007 IEEE int'l conf. Human-computer interaction, pp. 39–48, Springer-Verlag, Berlin (2007)

[14] Jia, P., Vision Open Statistical Models, http://sourceforge.net/projects/vosm (2011)

[15] Liao, S., Fan, W., Chung, A., Yeung, D.-Y.: Facial Expression Recognition using Advanced Local Binary Patterns, Tsallis Entropies and Global Appearance Features. In: IEEE Int'l Conf. Image Processing, pp. 665–668 (2006)

[16] Littlewort, G., Bartlett, M., Fasel, I., Chenu, J., Kanda, T., Ishiguro, H., Movellan, J.: Towards Social Robots: Automatic Evaluation of Human-robot Interaction by Face Detection and Expression Classification. Advances in neural information processing systems, 16, pp. 1563–1570 (2004)

[17] Lu, H., Pan, W., Lane, N., Choudhury, T., Campbell, A.: SoundSense: Scalable Sound Sensing for People-centric Applications on Mobile Phones. In: Proc. the 7th int'l conf. Mobile systems, applications, and services, pp. 165-178, ACM (2009)

[18] Lyons, M., Akamatsu, S., Kamachi, M., Gyoba, J.: Coding Facial Expressions with Gabor Wavelets. In: Proc. 3rd IEEE Int'l Conf. Automatic Face and Gesture Recognition, pp. 200-205, IEEE Computer Society, Washington, DC (1998)

[19] Matthews, I., Baker, S.: Active Appearance Models Revisited. In: Int'l J. Comput. Vision, 60(2), pp. 135-164 (2004)
[20] Michel, P., Kaliouby, R.E.: Real Time Facial Expression Recognition in Video using Support Vector Machines. In: Proc. the 5th int'l conf. Multimodal interfaces, pp. 258–264, ACM, New York (2003)
[21] Miluzzo, E., Lane, N., Eisenman, S., Campbell, A.: CenceMe: Injecting Sensing Presence into Social Networking Applications. In: Proc. the 2nd European conf. Smart sensing and context, pp. 1–28, Springer-Verlag, Berlin (2007)
[22] Mun, M., Reddy, S., Shilton, K., Yau, N., Burke, J., Estrin, D., Hansen, M., Howard, E., West, R., Boda, P.: Peir: The Personal Environmental Impact Report, as a Platform for Participatory Sensing Systems Research. In: Proc. the 7th int'l conf. Mobile systems, applications, and services, pp. 55-68, ACM, New York (2009)
[23] Radovanovic, M., Nanopoulos, A., Ivanovic, M.: On The Existence of Obstinate Results in Vector Space Models. In: Proc. the 33rd int'l conf. Research and development in information retrieval. pp. 186–193, ACM, New York (2010)
[24] Ristic, B., Arulampalam, S., Gordon N.: Beyond The Kalman Filter: Particle Filters for Tracking Applications. Artech House Publishers (2004)
[25] Shan, C., Gong, S., McOwan, P.: Facial Expression Recognition based on Local Binary Patterns: A Comprehensive Study. Image and Vision Computing, 27(6), pp. 803-816 (2009)
[26] Szeliski, R.: Computer Vision: Algorithms and Applications, Microsoft Research (2010)
[27] Viola, P., Jones, M.J.: Robust Real-time Face Detection. In: Int'l J. Comput. Vision, 57, pp. 137-154 (2004)
[28] Willogarage, OpenCV, `http://opencv.willowgarage.com/wiki` (2010)
[29] Yan, T., Kumar, V., Ganesan, D.: CrowdSearch: Exploiting Crowds for Accurate Real-time Image Search on Mobile Phones. In: Proc. the 8th int'l conf. Mobile systems, applications, and services, pp. 77-90, ACM (2010)
[30] Yilmaz, A., Javed, O.,Shah, M.: Object Tracking: A Survey. ACM Comput. Surv., 38 (2006)
[31] Zeng, Z., Pantic, M., Roisman, G., Huang, T.: A Survey of Affect Recognition Methods: Audio, Visual, and Spontaneous Expressions. In: IEEE Trans. Pattern Anal. Mach. Intell., 31(1), pp. 39-58 (2008)