Face Tracking and Pose Approximation

Kat Bradley
Kaylin Spitz

August 20, 2009

Abstract

Face detection and tracking systems are a common and useful task in computer vision applications. In this paper, we provide a survey of existing methods of detecting and tracking faces, and then describe the system we have created. Some experimental results and images are included, as well as suggestions for future work and improvements.

1 Introduction

Robust face tracking is an important topic in Computer Vision and presents a variety of uses. In a large teleimmersive system such as that at UC Berkeley, face tracking allows the use of a higher resolution camera on the facial area. High resolution on the face is essential because of the importance of the face to humans for non-verbal communication. A high-resolution face also adds to the realism of the teleimmersive system, which is important because human identity is strongly tied to the face. Knowledge of the pose of the face can aid in analyzing human interactions and communications within the teleimmersive system, for example to detect the direction of a person's gaze. Face tracking and pose tracking can also be used as pre-processing stages for facial recognition tasks.

Our paper begins with a short survey of existing literature. In section 3 we discuss the assumptions used to create the pose tracking system and provide a brief overview of the software package used to implement the system. We then explain in detail the implementation of our system. In section 4, we evaluate some results from our system and offer suggestions for further work. Section 5 concludes the paper.

2 Summary of Literature

Face detection is a topic long-studied in Computer Vision. The haar classifier technique used in our system was first proposed by Viola Jones [12]. Haar classifiers use a specific image representation called an integral image, which allows for the features to be efficiently represented and classified. The AdaBoost machine learning algorithm is then applied to a database of faces, creating a cascade structure that can be used to quickly narrow down where in the image the face is located. Other techniques for face detection not used in this paper include finding roughly elliptical areas based on color segmentation [10], texture, shape [9] , and edges. Most systems use some combination of these techniques to improve performance.

Face Tracking focuses on using knowledge about a face in previous frames to produce a faster, or more robust, face tracking system than using repeated face detection alone. The method used in this paper was proposed by Comaniciu, Ramesh, and Meer [4].

Once a face has been detected, it is necessary to detect features in the face. This can be done either by searching for specific facial features such as the eyes, nose, and mouth, or by simply finding generic feature points on the face. Because our system is simply looking for the location and pose of the face, we employ the second approach, but the first is very useful in systems which are interested in facial recognition or facial expressions, and could be integrated into our system if needed. One method of special feature location is to use feature-specific haar classifiers in a heuristically-reduced area of the face [13], but this suffers the same issues discussed later with facial detection haar classifiers. Color segmentation has also been used to find the eyes, nose, and mouth [10]. Deformable templates are another well-known technique for finding such areas [14].

One of the most well-known interest point detection algorithm is Harris Corners [5]. We initially used these features in our system, but discovered that the algorithm often returned points on the edge of the face, which became occluded in subsequent frames when the subject rotated his head. SIFT features [7] are another type of commonly used feature detector. The basic algorithm uses a Difference of Gaussians (DoG) at several different scales to find image points. It seeks to combat the edge issue found in Harris by using a Hessian matrix to eliminate points found along an edge. The algorithm also removes points that have low contrast. Our system uses Speeded Up Robust Features (SURF) which are similar to SIFT features. However, SURF features are faster than SIFT, and are more robust to various image transformations [1]. Our system uses the OpenCV implementation of SURF.

Finally, head pose detection can be approached in three basic ways: generalization from 3D points on the face (as in our system), using heuristics and knowledge of properties of the face, and 3D modeling of the face. An example of the heuristic method is given in [6], which uses distortions in the shape of the face to approximate pose from a monocular view. The 3D modeling approach is utilized in [3], which produces a pose estimation suitable for use in 3D animation.

3 Approach

3.1 Assumptions

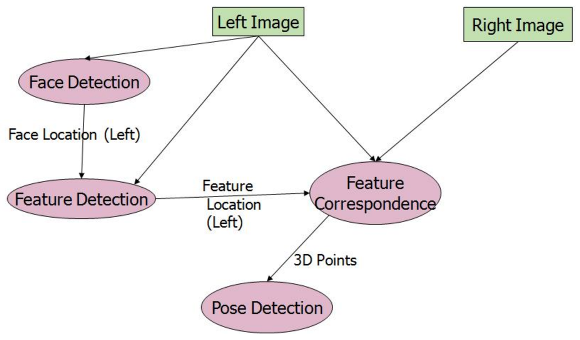
Our goal is to detect and track the pose of a human face through stereo video. We assume that there is only a single face in the initial frame of the image. The system could be easily adapted to track multiple faces in parallel, but we have not implemented this functionality. When initally presented with several faces, the system will choose the largest one. We also assume that the face begins the image sequence mostly frontal, and that it remains in similar lighting throughout the image sequence. The face needs to be large enough in the image that distinct points can be found. The system was tested on images at approximately 150x150 px resolution, but should be able to handle larger faces. However, exceptionally large faces may hinder performance.

The images provided to our system must be rectified beforehand, meaning that the right and left image have been projected into a common image plane. Our system must know certain information from the rectification process to function properly, including the intrinsic matrix, the baseline, and the focal length [8].

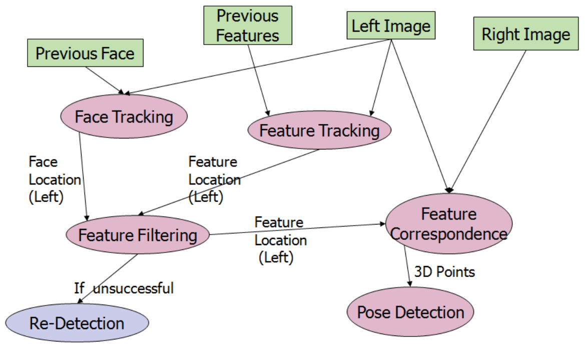
3.2 Software Package Used

Our system relies on Intel's OpenCV, a free open source C library of functions for vision research. Version 1.1 or later is needed to compile the source. OpenCV provides low-level functions, such as matrix and image structs, as well as high-level functions implementing many vision algorithms, such as facial detection, tracking, and point detection [2].

3.3 Overall system design



The overall layout of our detection system. A face is detected in the left-hand image, using a Haar classifier. Then, feature points in the face are detected using SURF. The feature points are matched to points in the right image using optical flow, and their 3D position is computed. A plane is fit to those 3D points, and the normal of the plane is the pose.



The overall layout of our tracking system. A face is tracked in the left-hand image, using knowledge about its previous position and the target color, with a mean-shift procedure. Then, feature points in the face are tracked using optical flow. The features are filtered, removing those not on the face and those that tracked poorly. The filtering process also recognizes if the tracked features and tracked face location disagree, allowing the system to re-detect instead of using poor tracking. Finally, as before, the feature points are matched to points in the right image using optical flow, their 3D position is computed, and a planar regression is performed.

3.4 Face Detection

A Haar classifier was used to detect the face, as implemented in OpenCV. The classifier is based off of Viola-Jones' original algorithm [12], and uses the cascades provided with OpenCV. This classifier has a tendency to identify multiple faces. Therefore, a heuristic is needed to decide which detected faces are correct. The system as presented simply finds the largest face in the image. A more robust approach is to find the face in both images, and match the two results which are most similar in location and area. However, the face detection is the slowest element of the system, and detecting the face in both images significantly reduced performance. In our tests, simply choosing the largest face seems to work well; in other applications, especially when one is attempting to track multiple faces, a more sophisticated heuristic will be needed.

3.5 Face Tracking

The face tracking component is based off Comaniciu, Ramesh, and Meer's Kernel-Based Object Tracking [4]. Colors are grouped into 128 'bins'. The target distribution, assumed to be the correct color of the face, is the histogram of the colors found in the original detected face. In each frame, the previous location of the face is used as an initial position, and the position is iteratively improved by moving towards areas more similar to the target distribution.

The original system has a variety of issues. If the lighting changes, the color of the face changes and the system cannot track it. If an item of similar color is placed in front of the face, the system may misidentify the item as the face and track it instead of the face. When the face turns, its color often changes, especially as shadows change. This paper will refer to these errors as "consistent errors", because it is easy to identify cases where they may occur, and because they tend to occur in both images when tracking is performed in stereo. In addition, sometimes the tracker misidentifies other areas as the face when there are no large lighting or angle changes. These areas often do not appear, to a human, to be a similar color as the face. Furthermore, these errors tend to occur in only a single image when tracking is performed in stereo. Thus, this paper refers to these errors as "inconsistent errors".

Improvements were added to the system to reduce consistent errors and remove most inconsistent errors. The face tracking is performed on both sides of stereo images (left and right). Each frame, the face tracking is performed in two color spaces, RGB and HSV and the two tracked faces with the most similar movement are chosen. Because the inconsistent errors usually occur in only one frame, by correlating the two frames, the inconsistent errors are almost removed. Furthermore, using two color spaces reduces consistent errors, which often occur ony in one color space, or occur more dramatically in one color space than in another.

However, tracking two sides in two color spaces with each of three sizes presents a performance problem. While the original mean-shift procedure is fast, it is not fast enough to be performed six times (as opposed to the original three times) per pair of frames in real-time. The larger the face is, the longer the original mean-shit procedure takes. Therefore, to reduce runtime, sampling was used. A 20x20 grid of evenly distributed points on the face is used for the mean-shift procedure instead of all points. Experimentally, using a sample of points instead of all points appears to produce similar results, although it increases the inconsistent noise. However, as the correlation between frames removes most inconsistent noise, this is a small issue.

3.6 Feature Detection, Tracking, and Filtering

The initial points to track are detected in the facial area of the left image using Speeded Up Robust Features (SURF). SURF finds key points in the image as described previously.

Once the SURF features have been detected in the initial frame, they are tracked in subsequent frames using Lucas-Kanade optical flow. The optical flow method is provided in OpenCV. The algorithm looks for sparse points locally, assuming spatial and temporal coherence between frames. It uses a pyramid scheme, which searches in a window that slowly shrinks in size, moving from less to more detail. This ability to track locals points in a global context gives the Lucas-Kanade method its robustness [2].

The movement of each feature point within the face (as tracked using the face tracking component) is computed, and the movements form two distributions (in x and y). Points are rejected if their movement is far from the mean (more than three standard deviations in either direction) or if they are far from the face. Points near the face but not in it are not automatically rejected, as the face tracking is imperfect.

If the majority of points are not in the detected face, re-detection of both face and feature points is performed (as clearly one of them is incorrect). If the number of points after filtering drops too low (less than 15), the feature points are re-detected.

3.7 Feature Correspondance

Using the points found in the left image, correspondences are found between the left and right images in a frame using the same Lucas-Kanade optical flow method that was previously used for feature tracking.

This is an unconventional use of optical flow, but it works well for our purposes because of the pyramid scheme previously described. The window must be slightly larger than when used to simply track the points. This is in anticipation of the larger disparity between the left and right images. With no other modifications, optical flow provides a robust estimate of the correspondences between the two images.

The OpenCV optical flow method provides an error estimate for each point match; we have screened the point matches according to a threshold to reduce the number of false points provided to the pose detection algorithm.

3.8 Pose Approximation

Using the correspondence information, a disparity (the distance between x coordinates in each image) is calculated for each valid point. Next, we use the disparity d to calculate the depth coordinate Z of the point in 3D space. This is done using the simple equation $Z = fb/d$, where f is the focal length and b is the baseline.

From [8], we have the following relation between image and world coordinate systems, where $K = \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$ is the intrinsic matrix.

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = K \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

From this, we derive the following explicit relations between X and Y (the 3D world coordinates) and x' and y' (the image coordinates):

$$\begin{aligned} x' &= \frac{Xfs_x + Yfs_\theta + Zo_x}{Z} \\ y' &= \frac{Yfs_y + Zo_y}{Z} \\ X &= \frac{Zx' - Yfs_\theta - Zo_x}{fs_x} \\ Y &= \frac{Zy' - Zo_y}{fs_y}. \end{aligned}$$

Once the points' location in 3D space has been established, the next step is to regress the points to a plane in which the face is likely to be located. This is done using a series of matrix operations from [11]. The normal vector to the regressed plane indicates the pose of the face in 3D space. The pose can then be drawn on the image by simply choosing two points in 3D space which are on the normal vector, converting those points to image points according to the above equations, and drawing a line between them. This is visually more useful when a point in the center of the face is chosen as a start point. The vector used should also be a unit vector in the 3D world, so that the length of the line drawn in the image plane helps visually convey the pose.

3.9 Kalman Filter

It is worth noting significant unsuccessful efforts not integrated in the final system. As the mean-shift procedure is highly dependent on color, an alternate method, utilizing feature points and a Kalman filter, was attempted.

In an initial frame, face detection, feature detection, and pose approximation are performed as before. Thus, we have an estimate for the position and pose of the face:

$$x, y, z, \theta, \phi, \psi,$$

where x, y, z are its position in absolute space, and θ, ϕ, ψ are its rotation along the x, y, z axes, respectively.
Likewise, we have an estimate for the position of each feature point (from the 3D reconstruction) and so we can compute its offset from the face on each of the x, y, z axes. These offsets are

$$o_{i,x}, o_{i,y}, o_{i,z}$$

for the ith point.
Assuming the face to be rigid (most feature points are on areas that do not move much with the face, such as eye corners and nostrils), the apparent positions of the feature points are related to pose (including position) of the head. If the apparent position of a feature point i is $a_{i,x}, a_{i,y}$ (in a single frame), then we expect:

$$\begin{aligned} a_{i,x} &= 1/z(x + o_{i,x}\cos(\phi) + o_{i,x}\cos(\psi) \\ &\qquad + o_{i,y}\sin(\psi) + o_{i,z}\sin(\phi)) \\ a_{i,y} &= 1/z(y + o_{i,y}\cos(\theta) + o_{i,y}\cos(\psi) \\ &\qquad + o_{i,x}\sin(\psi) + o_{i,z}\sin(\theta)). \end{aligned}$$

This assumes x, y, z are measured in arbitrary units (depending on how $a_{i,y}$ is measured). However, they can easily be converted to other units by a normalization constant. Knowing $x, y, z, \theta, \phi, \psi$ in the previous frame, we can approximate the positions of each point $a'i, x$ in the current frame as a more linear function of

$x', y', z', \theta', \phi', \psi'$ (the position in the current frame):

$$\begin{aligned} a'_{i,x} = 1/z'(x' &+ o_{i,x}(\cos(\phi) + \cos'(\phi)(\phi' - \phi) \\ &+ o_{i,x}(\cos(\psi) + \cos'(\psi)(\psi' - \psi)) \\ &+ o_{i,y}(\sin(\psi) + \sin'(\psi)(\psi' - \psi)) \\ &+ o_{i,z}(\sin(\phi) + \sin'(\phi)(\phi' - \phi))) \\ a'_{i,y} = 1/z'(y' &+ o_{i,y}(\cos(\theta) + \cos'(\theta)(\theta' - \theta)) \\ &+ o_{i,y}(\cos(\psi) + \cos'(\psi)(\psi' - \psi)) \\ &+ o_{i,x}(\sin(\psi) + \sin'(\psi)(\psi' - \psi)) \\ &+ o_{i,z}(\sin(\theta) + \sin'(\theta)(\theta' - \theta))) \end{aligned}$$

By inspection, we see that, except for the $1/z'$ term, this is a linear system in $x', y', \theta', \phi', \psi'$, taking the offsets(o terms), as well as the position in the previous frame $(x, y, z, \theta, \phi, \psi)$ as constants. We can thus write these equations as:

$$\begin{aligned} a'_{i,x} &= 1/z'(x' + c_{i,x,\phi}\phi' + c_{i,x,\psi}\psi' + c_{i,x,1}) \\ a'_{i,y} &= 1/z'(y' + c_{i,y,\theta}\theta' + c_{i_x,\psi}\psi' + c_{i,y,1}), \end{aligned}$$

for some constants c (acquired through algebraic manipulation on the earlier linearized equations).
Furthermore, $1/z'$ can be approximated without the use of such equations by considering the average distance between feature points (which will scale with $1/z$. Such an estimate is imperfect, but allows us to determine $1/z'$ and treat it as a constant, reducing our equations to:

$$\begin{aligned} a'_{i,x} &= d_{i,x,x}x + d_{i,x,\phi}\phi' + d_{i,x,\psi}\psi' + b_{i,x} \\ a'_{i,y} &= d_{i,y,y}y + d_{i,y,\theta}\theta' + d_{i_x,\psi}\psi' + b_{i,y}, \end{aligned}$$

where the d's are simply constants. Thus, our observations can be related as a linear function of the pose. Finally, we manipulate our equation to find:

$$\begin{aligned} a'_{i,x} - b_{i,x} &= d_{i,x,x}x' + d_{i,x,\phi}\phi' + d_{i,x,\psi}\psi' \\ a'_{i,y} - b_{i,y} &= d_{i,x,y}y' + d_{i,y,\theta}\theta' + d_{i_x,\psi}\psi', \end{aligned}$$

so the left side is a matrix multiplication on the pose:

$$A = Dv,$$

where

$$A = \begin{bmatrix} a'_{0,x} - b_{0,x} \\ a'_{0,y} - b_{1,y} \\ a'_{1,x} - b_{0,x} \\ a'_{1,y} - b_{1,y} \\ . \\ . \\ . \end{bmatrix}$$

$$D = \begin{bmatrix} d_{0,x,x} & 0 & 0 & 0 & d_{0,x,\phi} & d_{0,x,\psi} \\ 0 & d_{0,y,y} & 0 & d_{0,y,\theta} & 0 & d_{0,y,\psi} \\ d_{1,x,x} & 0 & 0 & 0 & d_{1,x,\phi} & d_{1,x,\psi} \\ 0 & d_{1,y,y} & 0 & d_{1,y,\theta} & 0 & d_{1,y,\psi} \\ ... & ... & ... & ... & ... & ... \end{bmatrix}$$

$$v = \begin{bmatrix} x \\ y \\ z \\ \theta \\ \phi \\ \psi \end{bmatrix}$$

As the constants vary over time, the matrices need to be updated at each time step.
The transition matrix from one hidden state to the next is simply the identity.
Unfortunately, no working implementation of this technique was completed.

4 Results

The facial detection is fairly accurate for frontal faces, but not very accurate for non-frontal (such as profile) faces. When no face is detected, it simply tries the next frame, and often within a couple frames the face can be detected.

The facial tracking is accurate when the lighting on the face does not change. However, lighting on the face may change for a variety of reasons, such as movement through a non-uniform space, shadows, and even tilting the head towards or away from the light. Nevertheless, in most of our experiments, the lighting stayed constant enough for effective tracking.

The feature detection consistently finds enough features for pose approximation. However, many (about 40%) of the detected features do not track well. These features are consistently removed through filtering, leaving features that track well. Likewise, some of the features do not correspond well (between the two frames), often because of occlusions. Again, the filtering seems to remove most incorrect correspondences, allowing for accurate 3D reconstructions. An example of detected features and the results of filtering is shown below.

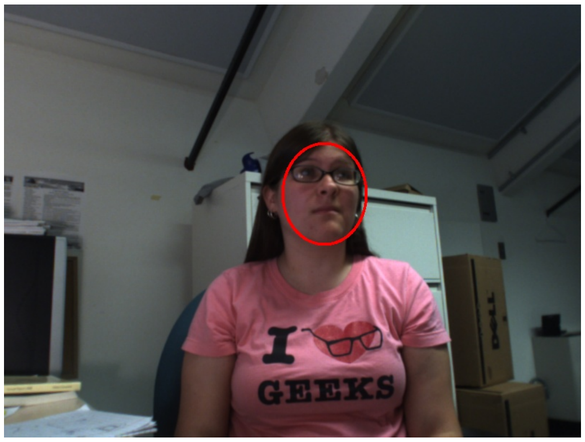


Figure 1: A computed facial location after detecting and some tracking.

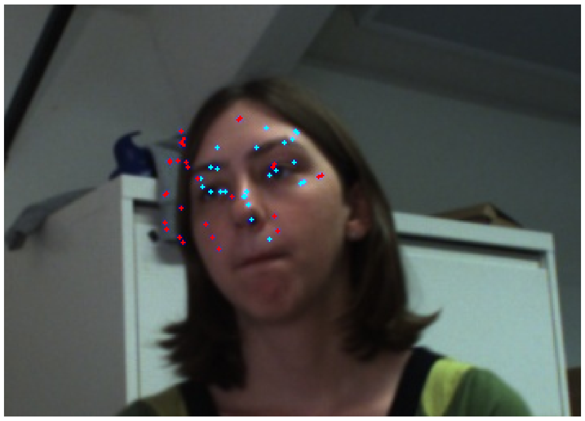


Figure 2: Initially detected points. Those in blue remained after tracking; those in pink were filtered out.

Note that the features filtered out tend to be in homogenous regions (such as the cheeks) or off the face. The remaining features are well-located and on the face.

Pose detection as presented is highly dependent upon the points that are found and matched previously in the algorithm. If the points are not well-distributed across the face, or points which are not actually on the face are used, the results are poor, so it is important that the face and feature tracking are highly accurate. Also, regressing all the points to a plane treats the face as though it is a rigid object, which may also hinder robustness, especially when there are occlusions or the subject is wearing objects on the face such as glasses. Despite these limitations, there was an acceptable degree of accuracy for pose detection.

Currently, the system takes approximately 350 milliseconds for initial face detection and 40 milliseconds for face tracking on a 3.06 GHz machine. Approximately 130 milliseconds are spent detecting the SURF features; 20 milliseconds are spent tracking them, and 25 milliseconds are used finding the correspondences. Thus, in a typical frame where the face and features are tracked, the total time spent is approximately 85 milliseconds. It is thus potentially suitable for real-time tracking, especially if certain optimizations such as parallelizing the face tracking code were made.

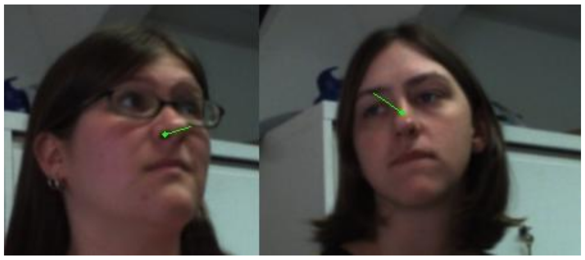


Figure 3: Sample pose detections, shown by a unit vector.

Redetection of the face is by far the slowest part of the process, and therefore should be done as sparingly as possible. Ideally, redetection should happen every 10-20 frames to ensure accuracy, but for real time, it is best to only redetect when the tracking has gone significantly awry. Redetecting the SURF features can happen more often and still maintain real-time speeds.

4.1 Future Work

As previously mentioned, the most time-intensive component of our system is the detection of the initial face. Faster, more robust methods of detecting the initial face location should be considered in any future work. When redetection of the face is necessary, it may be possible to use the incorrect location and size to heuristically reduce the search space for the new location. The image size can also be reduced before attempting face detection, which can significantly reduce runtime.

More interaction between steps in the system may improve robustness. For example, facial points could be fed back into the face tracking and detection to enhance the overall estimate of face location. The pose detection may also be used to improve the tracking in subsequent frames; a Markov model predicting the movement of the face may further enhance performance. Finally, the use of a feature detection scheme that searches out definite points such as the eyes, nose, and mouth is recommended if this system is to be expanded for work in facial recognition or human computer interaction.

5 Conclusions

In this paper, we have presented a stereo vision system which detects, tracks, and estimates the 3D pose of human faces. This system is adaptable for many uses, including focusing a higher resolution camera on the face in the context of a large tele-immersive system. Other applications of our work include analysis of human facial expression and movement.

The authors have learned much from this work about computer vision and the research process in general. We would like to thank the members of the tele-immersion lab at UC Berkeley, especially prof Ruzena Bajcsy, Dr Gregorij Kurillo, and Zeyu Li for their guidance and assistance. In addition, we would like to thank the CRA-W and the DREU program for funding and support.

References

[1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded-up robust features. In *9th European Conference on Computer Vision.*

[2] Gary Bradski and Adrian Kaehler. *Learning OpenCV*. O'Reilly, 2008.

[3] Junchul Chun, Ohryun Kwon, and Peom Park. A robust 3d face pose estimation and facial expression control for vision-based animation. *Advances in Multimedia Modeling*, 2006.

[4] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(5):564–577, May 2003.

[5] C. Harris and M. Stephens. A combined corner and edge detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[6] Qiang Ji. 3d face pose estimation and tracking from a monocular camera. *Image and Vision Computing*, 20(7):499 – 511, 2002.

[7] David Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, pages 1150–1157, 1999.

[8] Yi Ma, Stefano Soatto, Jana Kosecka, and S. Shankar Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models.* Springer, 2004.

[9] Eli Saber and A. Murat Tekalp. Frontal-view face detection and facial feature extraction using color, shape, and symmetry based cost functions. *Pattern Recogn. Lett.*, 19(8):669–680, 1998.

[10] Karin Sobottka and Ioannis Pitas. A novel method for automatic face segmentation, facial feature extraction and tracking. *Signal Processing: Image Communication.*

[11] Dan Teague, Bob Stephenson, Chris Olsen, and Gloria Barrett. Regression analysis. *Notes from the NCSSM Statistics Leadership Institute*, 1999.

[12] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Computer Society Conference On Computer Vision and Pattern Recognition*, pages 511–518, 2001.

[13] Phillip Ian Wilson and John Fernandez. Facial feature detection using haar classifiers. *J. Comput. Small Coll.*, 21(4):127–133, 2006.

[14] Alan L. Yuille, Peter W. Hallinan, and David S. Cohen. Feature extraction from faces using deformable templates. *Int. J. Comput. Vision*, 8(2):99–111, 1992.