Head Pose Estimation of Partially Occluded Faces

Markus T. Wenzel and Wolfram H. Schiffmann
University of Hagen (Germany), Department of Computer Science
Markus.Wenzel | Wolfram.Schiffmann@FernUni-Hagen.de

Abstract

This paper describes an algorithm which calculates the approximate head pose of partially occluded faces without training or manual initialization. The presented approach works on low-resolution webcam images.

The algorithm is based on the observation that for small depth rotations of a head the rotation angles can be approximated linearly. It uses the CamShift (Continuous adaptive Mean Shift) algorithm to track the users head. With a pyramidal implementation of an iterative Lucas-Kanade optical flow algorithm, a certain feature point in the face is tracked. Pan and tilt of the head are estimated from the shift of the feature point relative to the center of the head. 3D Position and roll are estimated from the CamShift results.

1. Introduction

Immersive systems may display their graphical user interface on a *Head Mounted Display* (HMD). In this setup head pose estimation is a valuable *Human-Computer Interface* (HCI), but all head pose estimation algorithms known to the authors assume a completely visible face. Therefore navigation using head movement detection has not yet been presented as an HCI in this setting.

For head pose estimation face detection in images is an issue, because it helps limiting the computational effort to the area exhibiting the face. Manifold research with respect to face detection was already conducted with different objectives and scaffolding. For use in human computer interfaces real-time performance both of face detection and of head pose estimation is compulsory.

This paper focuses on real-time head pose estimation in video images. At startup the algorithm detects the head region using the temporal change between subsequent images. From that head region a cutout containing skin color is used to initialize the CamShift algorithm, which tracks the head henceforth. In the same cutout region the mouth is searched for by a combination of heuristics giving its most probable location and a set of image filters responding to the typical light-dark intensity pattern resembling the mouth.

2. Related Work

Head pose estimation has become a broad area of research over the recent years. Most researchers group the various approaches with respect to the underlying method into *model-based*, *appearance-based*, and *feature-based* approaches [4, 16].

Model-based algorithms achieve good results but require training. Among these are Active Appearance Models (AAMs); see for example [9]. These are nonlinear parametric models derived from linear transformations of a shape model and an appearance model. Also, Neural Network based algorithms can be trained to distinguish between different persons or to distinguish poses of one persons face. The basic research was done by Rowley *et al.* [12]. Principal Component Analysis (PCA) based algorithms in general need the broadest training set. Their training is most often done in a way first proposed by Turk and Pentland [15]. With their research PCA was able to deal with images. As they are statistically inspired, PCA approaches are sometimes counted within the appearance-based approaches.

The main disadvantage of all AAMs, Neural Network approaches and PCA methods is their lack of universality. Their detection rate and accuracy decreases as soon as the head does not match the model. This will happen if the user puts on an HMD (Head Mounted Display). In the same manner the performance of any Neural Network based vision system trained for a specific user decreases if confronted with unknown users.

Appearance-based approaches use filtering and image segmentation techniques to extract information from the image. Furthermore, optical flow algorithms can be considered as appearance-based approaches. Among the widely used filters are edge detectors and in recent times Gabor wavelets. To the authors no head

pose estimation algorithms are known that are only appearance-based. On the other hand, filtering and segmentation play significant roles in head pose estimation.

Gabor wavelets are widely known as good feature detectors. These are complex sinusoids with a gaussian envelope. Some approaches use them in an appearance-based manner in a preprocessing step [10]. On the other hand Krüger invented an approach where the weights of a Gabor wavelet network directly represent the orientation of the face. Gabor head pose estimation is described in [16]. Its disadvantage, however, is the computational effort involved in fitting the network to the probe image. Moreover, they have to be trained and are very user specific.

The algorithm proposed in this paper uses the Gabor wavelet transform for parts of the face in order to search for the mouth.

The majority of *feature-based* algorithms use the eyes as features as these are easy to detect by symmetry and because of their prominent appearance. The nostrils often serve as trackable features as well, but they become invisible as soon as the user tilts his head downwards. The mouth can also be easy to find, provided it is not covered by a mustache or a beard. Several authors use a set of these features to estimate a 3D head orientation.

Fitzpatrick shows another feature based approach to head pose estimation without manual initialization [5]. For feature detection and tracking he searches for the cheapest paths across the face region, whereby the cost of a path depends on the darkness of crossed pixels. The paths will therefore avoid dark regions. A pair of avoided regions is supposed to be the pair of eyes. The algorithm is thus dependant on the visibility of the eyes. He then estimates head pose based mainly on head outline and eye position.

Gorodnichy shows a way to track the tip of the nose [7]. His approach can almost be called model free. He uses the resemblance of the tip of the nose with a sphere with diffuse reflection. This template is searched in the image. He does not use it for head pose estimation, but simply tracks the nose tip across the video images.

The idea presented in this paper resembles his approach in that an individual feature is tracked in the images.

3. Pose Estimation with Partial Occlusion

The intent of this paper is to present an algorithm that is capable to adapt to every user and especially to faces partially occluded by a HMD (see Figure 1). No initial setup or training is needed. The algorithm delivers approximations of head roll, tilt and pan (see Figure 2). The head position in the image plane is given directly and the z axis translation can be estimated.

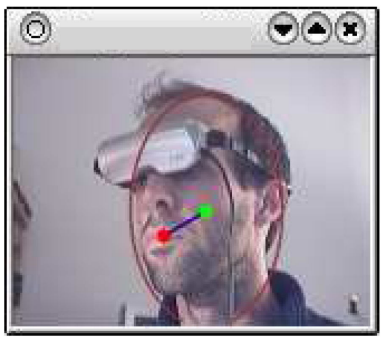

Figure 1. User with HMD

The pose of a face may be depicted as a 3D vector $\vec{d}$ with its origin between the eyes. For head pose estimation two pieces of information are needed: 3D head position given at the middle of the head and the position of a feature of the face.

The pose is then a linear transformation of the vector $\vec{d}$ in Figure 2. The user may define a certain position to be his "straight ahead" direction. The vector obtained in this position is stored and used as a reference. Then for small angles in depth rotation the rotation angle can be linearly estimated. Head roll can be estimated from the rotation of an ellipse in the image plane approximating the head outline. This will be described below.

Estimating mixed rotations around all three axes is still challenging, because every position is ambiguous. This problem is not discussed in this paper, but remains open for further research.

The algorithm has two phases we call the *initialization phase* and the *working phase* (Figure 3).

Initialization extracts the face region from the video sequence and finds a salient feature point to track. The proposed method aims at finding the mouth as a prominent feature. If the mouth is invisible or can not be detected, the algorithm will track another feature which, in most cases, is the tip of the nose or one of the nostrils. In case of failure the working phase returns to the initialization phase.

3.1. Initialization phase

The initialization phase needs head movement to detect the face of a user. Therefore, a state called *motion detection* starts the initialization phase when motion in the picture occurs. The initialization phase searches

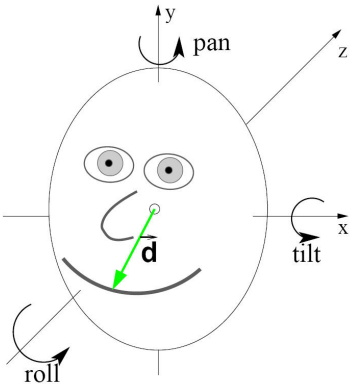

Figure 2. Basic idea

the users head. In the estimated head region the algorithm searches a salient trackable feature. It passes on the coordinates and the size of a region to CamShift. During the working phase the CamShift algorithm is used to track the biggest object in the picture that is of this regions dominant color.

Motion Detection. For the face detection step to succeed, the head must have moved far enough. This is ensured by calculating the difference between a base image $I(t_0)$ and a subsequent image $I(t_1)$, $I(t_2)$, $\cdots$ from the video stream until the difference $I(t_0) \ominus I(t_k)$for a certain $k \geq 1$ exceeds a threshold τ. In the prototype implementation, the $\ominus$-operator is implemented as a sequence of operations: Subsequently a XOR image (pixel-wise and per channel) is built. Morphological opening (erosion then dilation) and closing (dilation then erosion) operations account for closing holes and deleting small outliers. From the remaining image only intensity information is kept. This intensity image will represent the most significant motion with the brightest color. In this way thresholding further removes noise.

Face Detection. After the above operation, a contour search gives the largest remaining BLOBs (binary large objects) in the image. Given the assumption that only head movement occurs during initialization, the biggest BLOBs can be considered to represent the head. The image is saved for later steps.

The next step requires a skin color probe. Care must be taken not to include the HMD in this extracted skin color probe, because it may reflect arbitrary colors and thus invalidate the results. We assume that a small region in the lower half of the moving part will mainly contain skin. The coordinates and the size of this *color probe region* are given by simple heuristic assumptions

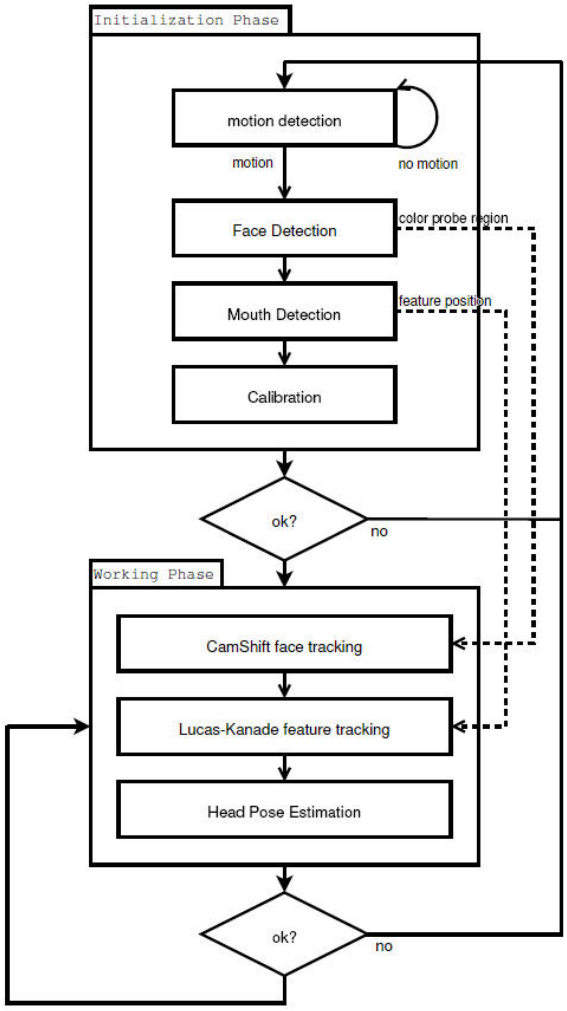

Figure 3. Details of algorithm

about face biometrics. They are relative to the size and location of the biggest BLOBs in the above image.

Skin color segmentation is done in the "normed green" color space. The algorithm calculates a normed green image from the color camera image by applying $g_{norm} = \frac{G}{R+G+B+1}$ to every RGB (Red Green Blue) pixel of the image. A color histogram of the above color probe region is then calculated in the g_{norm} image. Thresholding the g_{norm} image within a tight band around the maximum color value of the histogram gives the locations of skin regions. The coordinates and the size of the color probe region are passed on to CamShift as well.

Skin color tones are tightly clustered in the r_{norm}/g_{norm} color space [13]. Experiments show that in the given context the g_{norm} part is sufficient. The transformation into this space is inexpensive in terms of computational effort. It shows

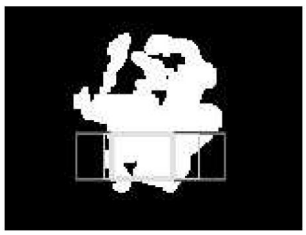

Figure 4. Estimated head; search region for mouth

more reliable results than thresholding the hue channel in the HSI (Hue Saturation Intensity) color space.

Pixel-wise AND-concatenation of the biggest BLOBs image with the binary thresholded g_{norm} image gives a re- fined picture of the moving head region. Figure 4 shows the result (white areas).

Position and size of a bounding box of these segments serve as the starting point for the search for a salient trackable feature point. The *mouth search region* is found based on an initial heuristic estimation of its most probable height. We assume $\frac{2}{5}$ the height of the bounding box, located in its lower region. The width is that of the bounding box. The resulting initial mouth search region is displayed in Figure 4; outmost box. This box is shrunk to a size containing mostly white pixels (innermost box in Figure 4).

Mouth Search. The mouth is a salient trackable feature, because it exhibits contrasting regions like edges and corners. Simple heuristics give a rough estimate of its position (the *mouth search region*). A near-frontal position of the users face is assumed during initialization. Figure 4 shows the segments displaying the head (white areas) and subsequent mouth search regions (grey rectangles, starting with the largest and ending with the innermost).

A series of filters is then applied to the innermost region of the image. Figure 5 gives an example of a mouth search region (top image in figure) and the series of intermediate results. Note that the size of this region is variable.

The algorithm starts with a Sobel operator to enhance intensity changes (second from top). The Sobel filter works with the first y derivative and an aperture size of seven pixels.

Then the algorithm applies a horizontally oriented Gabor filter to the result. Its frequency resembles the smooth vertical intensity gradient of the lips to reduce the higher and lower frequencies in the image. For the result see the middle picture.

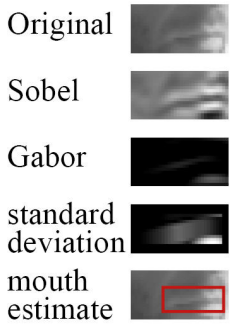

Figure 5. Filtering the mouth search region

Then a box three times higher than phase width is shifted across the result image. The standard deviation of every possible box codes the intensity of a corresponding pixel in a result image (second from bottom). Figure 6 illustrates this. Here the resulting image is normalized for better visibility of the results. The box is currently of fixed size, but can be made dependant on the size of the mouth search region.

A second fixed-sized box is shifted across the image, this time with the expected size of the mouth. The box with the greatest sum of standard deviations (i.e. highest pixel intensities) is considered to be the most likely mouth region. In Figure 5, bottom image, this region is marked with a rectangle.

The approach performs best for near-horizontal pictures of the mouth, but is still accurate for head roll up to 10°. Head pan up to 30° will not influence performance and the same is true for head tilt.

3.2. Working phase

During working phase the algorithm performs three tasks. *Head Tracking* is done with CamShift. *Feature tracking* is accomplished with an iterative Lucas Kanade optical flow algorithm in image pyramids. The results of both are combined for *head pose estimation.* Each task is described below.

Head Tracking. The head is tracked using the CamShift algorithm. CamShift is able to track every object with a distinct hue value. We initialize it as described above to track the moving object in the image.

CamShift is an adaption of the *Mean Shift* algorithm for face tracking. Mean Shift is a non-parametric technique that climbs the gradient of a color distribution to find its mode (peak) [1].

CamShift augments the Mean Shift algorithm by adding a variably sized search region that gives rise to its name: *Continously Adaptive Mean Shift.*

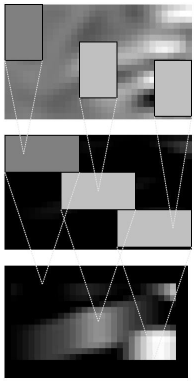

Figure 6. Shifting boxes across the image

The position of the tracked face is given by the centroid of the color distribution. It is calculated from the moments of the distribution, as given by the below formulas. Utilizing the size of the search region, an estimate of z-axis position can be obtained.

Let W be a search window with image points $I(x, y)$. The centroid of the distribution $\vec{\mu} = (\mu_x, \mu_y)^T$ is then given by

$$\mu_x = \frac{M_{10}}{M_{00}}; \quad \mu_y = \frac{M_{01}}{M_{00}} \tag{1}$$

with M_{ij} being the i^{th} moment in x direction and j^{th} moment in y direction. M_{ij} is defined as

$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y)$$

The orientation of the color distribution can be calculated using zero$^{\text{th}}$ to second moments. CamShift delivers the center of a rotated ellipse and its axes lengths. CamShift works by first back projecting the hue histogram of the color probe on the current image. Then it calculates the first and second moments of the color probability distribution. These give the lengths of first and second axis and the ellipses rotation angle.

For implementation details for CamShift see [3]. The implementation used in the prototype application is included in the Intel OpenCV computer vision library [11]. Some of its shortcomings are explored in [1].

Feature Tracking. For feature tracking the algorithm uses an enhanced optical flow algorithm as proposed by Bouguet [2]. It can be used to track features in a video sequence.

The basis of the feature tracking approach is an iterative Lucas Kanade optical flow algorithm. It works in image pyramids to suit the task of robust feature tracking. The fundamental Lucas Kanade optical flow algorithm was proposed by Lucas and Kanade in [8].

Figure 7. Image pyramid, levels 0-3

Let $W \subset J$ be a window of size $(2w_x + 1) \times (2w_y + 1)$. Let its center be the coordinates $\vec{u} = (u_x, u_y)^T$ of a feature point in the preceeding picture J. Let I be the current picture.

Then optical flow algorithms try to find a window W' in I of equal size to W that is most "similar" to W. Similarity may be measured by the least square error between two images. In this case estimating optical flow means minimizing ϵ in

$$\begin{aligned} \epsilon(\vec{d}) &= \epsilon(d_x, d_y) \\ &= \sum_{\substack{x=u_x-w_x \\ y=u_y-w_y}}^{\substack{x=u_x+w_x \\ y=u_y+w_y}} (I(x, y) - J(x + d_x, y + d_y))^2 \end{aligned} \tag{2}$$

by finding the optimal *image velocity* $\vec{d}$ concerning the image point $\vec{u} = (u_x, u_y)^T$.

The Lucas Kanade algorithm estimates $\vec{d}$ based on discrete estimations of spatial intensity derivatives in the current image. Assuming small displacement vectors, the first order Taylor series expansion about the feature point can substitute $J(x + d_x, y + d_y)$ in the above Equation 2, giving

$$\frac{\partial \epsilon(\vec{d})}{\partial \vec{d}} \approx -2 \sum_{\substack{x=u_x-w_x \\ y=u_y-w_y}}^{\substack{x=u_x+w_x \\ y=u_y+w_y}} \begin{bmatrix} \frac{\partial J}{\partial x} \\ \frac{\partial J}{\partial y} \end{bmatrix} \left(I_{xy} - J_{xy} - \vec{d} \begin{bmatrix} \frac{\partial J}{\partial x} \\ \frac{\partial J}{\partial y} \end{bmatrix} \right) \tag{3}$$

Note that this is only valid for small displacement vectors $\vec{d}$.

The iterative Lucas Kanade iterates over this step taking $\vec{d}$ from the previous step as an initial guess for

Figure 8. Displacement of feature with respect to face center

the next step. This introduces greater accuracy but alone cannot overcome the limitation to small displacement vectors.

Bouguet [2] suggests the usage of image pyramids to improve on this limitation. Image pyramids are pyramids of subsampled images with successively lowered resolution. For a 160×120 base image (zeroth pyramid level), the first level image has a size of 80×60 etc. Figure 7 shows zeroth (top) to third pyramid levels of an image, created with a 5×5 gaussian filter.

The iterative Lucas Kanade in image pyramids conducts iterative Lucas Kanade steps in each level of an image pyramid starting at the highest level, i.e. with the lowest resolution. The result of the lowerresolution level is propagated to the next higher-resolution pyramid level and used as an additive offset.

This enables iterative Lucas Kanade to track optical flow with large displacements and great accuracy.

Head Pose Estimation. Figure 8 sketches the idea underlying depth rotation approximation. The center of the head (i.e. the centroid of the color probability distribution as given by CamShift) is marked with a green spot. The tracked feature is marked by a red spot. The arrow gives the relative shift of the red spot with respect to the "straight ahead" position of the mouth. From the current shift relative to the initial shift which marks the straight ahead direction, the current depth rotation is estimated. Currently, only a linear approximation is calculated.

In the figure, the "straight ahead" mouth position is marked for clarity. In the prototype immersive system, all estimated rotation parameters and z translation are pictured by compass windows (see Figure 9).

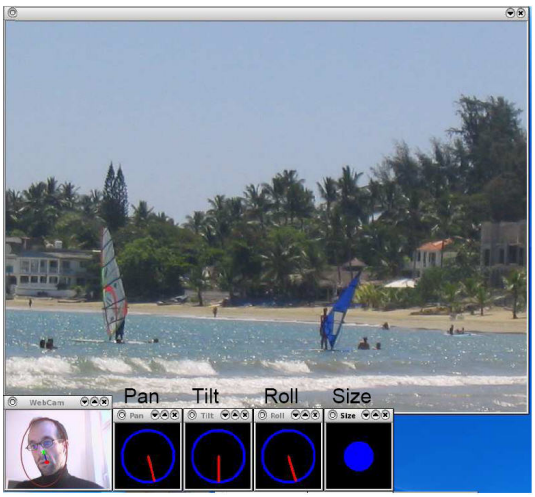

Figure 9. Demo Application with Compass Windows

4. Results

The algorithm performs with far more than camera frame rate on a 1,066 MHz notebook. The tests were conducted both on still images and live video sequences captured by a Logitech QuickCam 3000 and aimed at evaluating quality and speed of the algorithms protoype implementation.

Assessment of Speed. All tests on still images were conducted by using two pictures I and J that were presented vice versa to the different parts of the algorithm.

Concerning initialization phase, no efforts were taken to improve performance of the implementation. Nevertheless, the initialization phase calculated about 25 pose estimations per second.

In the working phase, CamShift aims primarily at high speed. With the described setup and given that the area tracked covers about a quarter of the 160×120 picture, it performs at about 500 pictures per second.

The iterative Lucas Kanade feature tracker tracking a singular point in the image performs at roughly 3000 pictures per second regardless of displacement of the feature point.

When both algorithms were combined, a rate of about 400 pictures per second was measured. Thus, real time pose estimation can be achieved by means of commodity hardware.

Assessment of Quality. The initialization phase was tested for quality with various image pairs showing the face of the user in slightly altered positions (approximately 5° change in pan angle).

The initialization phase proved to be insensitive to lighting conditions. As well, long hair is tolerable as long as certain thresholding values in motion detection are altered slightly. Most importantly, the usage of an HMD will not influence the results of initialization. The algorithm thus achieves the goals it is designed for.

Severely disturbing for the goals of initialization are, however, all moving objects in the scene that are not a users head, even more so, if they are of skin color. The initialization phase will focus on whatever moves in the scene and try to detect a mouth in the moving parts. Due to the model free nature of the implementation, no checks are currently possible to account for this.

We then tested the overall performance quality in a small demonstration application (see Figure 9). A panoramic picture is presented to the user via HMD. By panning and nicking his head, he may alter the viewport to the panorama.

During working phase it is apparent that the linear approximation of the angle does not provide enough accuracy to feel fully immersed. Also, it is necessary to dampen the pose and orientation estimates. Both CamShift and the Lucas Kanade feature tracker sometimes pass on suddenly changing results that lead to a shattering impression of image movement on the output device.

Damping diminishes this impression, but with the downside of retarded system reaction. Although the reaction lags behind head rotation for only a fifth of a second, this can be disturbing.

5. Further Work

Because reliable detection of the mouth region is crucial to pose estimation, one task is to improve the initialization phase of the algorithm. Incorporation of recent research results making use of methods resembling human eye movement (saccadic search, see [14]) promise more exact and more robust estimations. For those approaches to work, a model of the feature and consequently a training step is necessary. Current research suggests that user specific training may not be mandatory. Therefore it seems to be a promising approach.

Also, a modification to CamShift will enable it to adapt to changes of the tracked color. This can improve tracking results. The methods suggested in [1, 6] promise better color segmentation in difficult lighting conditions and the just-in-time adaption to changing hue values of the tracked object.

To overcome the restrictions in accuracy introduced through the linear approximation of rotation values, we consider tracking more than one feature to apply a weak projection model for head pose estimation.

Also, as mentioned before, the problem of estimating mixed rotations around two or three axes remains unsolved yet. With more feature points tracked, ambiguities might be overcome.

6. Conclusion

The purpose of this paper was to show how a user's head pose can be tracked in a video stream when parts of his face are occluded. We presented an algorithm composed of two dedicated phases, initialization and working phase, that achieved this goal.

During initialization, the face is detected from subsequent pictures by its motion. A color probe is drawn and given to CamShift for tracking during working phase. The mouth is searched for with a combination of heuristics and filtering. Heuristics give the most probable region where the mouth is expected. The mouth is then detected by applying a series of filters to this region.

In the working phase, head and feature position are tracked in time. The relative shift of the feature point relative to both the "straight ahead" feature position and the center of the head gives the user's head pose. The algorithm estimates six degrees of freedom (rotation and translation with respect to all three spatial dimensions). The algorithm is usable as a HCI component in immersive systems. This was shown in a demonstration application.

References

[1] J. G. Allen, R. Y. D. Xu, and J. S. Jin. Object tracking using CamShift algorithm and multiple quantized feature spaces. In Conferences in Research and Practice in Information Technology, volume 36, 2003.

[2] J.-Y. Bouguet. Pyramidal implementation of the Lucas Kanade feature tracker. OpenCV Documentation.

[3] G. R. Bradski. Computer vision face tracking as a component of a perceptual user interface. In Workshop on Applications of Computer Vision, pages 214219, Princeton, NJ, Oct. 1998. Microcomputer Research Lab, Santa Clara, CA, Intel Corporation.

[4] L. M. Brown and Y.-L. Tan. Comparative study of coarse head pose estimation. Technical report, IBM T.J. Watson Research Center, Hawthorne, NY, 2002.

[5] P. Fitzpatrick. Head pose estimation without manual initialization. Technical report, AI Lab, MIT, Cambridge, USA, 2000.

[6] A. R. Francois. Real-time multi-resolution blob tracking. Technical Report IRIS-04-423, Institute for

Robotics and Intelligent Systems, University of Southern California, July 2004.

[7] D. O. Gorodnichy. On importance of nose for face tracking. In Proceedings of the International Conference on Automatic Face and Gesture Recognition (FG2002), NRC 45854, pages 188196, Washington, DC, USA, May 2002.

[8] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In Proc. of 7th International Joint Conference on Artificial Intelligence (IJCAI), pages 674679, 1981.

[9] I. Matthews and S. Baker. Active appearance models revisited. International Journal of Computer Vision, 60(2):135 164, Nov. 2004. In print.

[10] S. J. McKenna and S. Gong. Real-time face pose estimation. Real-Time Imaging, 4(5):333347, 1998.

[11] OpenCV. Sourceforge.net OpenCV project homepage. http://sourceforge.net/projects/opencvlibrary.

[12] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(1):2338, Jan. 1998.

[13] S. K. Singh, D. S. Chauhan, M. Vatsa, and R. Singh. A robust skin color based face detection algorithm. Tamkang Journal of Science And Engineering, 6(4):227234, 2003.

[14] F. Smeraldi and J. Bigun. Retinal vision applied to facial features detection and face authentication. Pattern Recognition Letters, 23:463475, 2002.

[15] M. Turk and A. Pentland. Eigenfaces for recognition. Journal of Cognitive Neuroscience, 3(1):7186, 1991.

[16] Y. Wei, L. Fradet, and T. Tan. Head pose estimation using gabor eigenspace modeling. Technical report, National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Science, 2001. 7