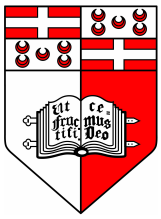

University of Malta
L-Università ta' Malta

Technical Reports in Systems and Control Engineering

Faculty of Engineering **SCE-TR-2014-01**
Department of Systems and Control Engineering **November 24, 2014**

Department of Systems and Control Engineering
Faculty of Engineering
University of Malta
Msida, MSD 2080
Malta
www.um.edu.mt/eng/sce

Fasthpe: A recipe for quick head pose estimation

Michael Sapienza and Kenneth P. Camilleri

June 2011

mikesapi@gmail.com
kenneth.camilleri@um.edu.mt

Abstract

Estimating the head orientation of a person from a single camera is an important step for human-computer interaction, especially for widely available laptops and hand-held devices. This work aims to track a human face and estimate its orientation in the 6 degrees of freedom from an uncalibrated monocular camera, keeping the user free of any devices or wires. We propose a novel algorithm based on existing computer vision techniques for a real-time (2ms) head pose estimation system, which can start and recover from failure automatically without any previous knowledge of the user's appearance or location. We demonstrate that this computationally efficient pose estimation system is able to track the continuous roll, yaw, and pitch angles within absolute errors of 3.03, 5.27 and 3.91 degrees respectively. We show that with the tracking of only four face features, it is possible to obtain continuous head orientation measurements in real-time (2ms).

1 Introduction

Estimating a person's head orientation may be of little difficulty to humans, however solving this complex problem with computers still remains a challenging research topic [1]. Automated head pose estimation has far reaching applications in the ways people control computers and in the examination of human behaviour. Potential applications include facial expression analysis, gesture recognition, virtual reality and new ways of interacting with games. Since human head orientation is also an indication of a person's gaze direction [2], estimating head pose can also provide critical feedback on commercial products and advertisement impact. Ultimately, if a robot is to keep eye contact with its human friend, it must first find the user's head pose.
In this work, the head pose estimation problem is considered in a gaming scenario, in which the user must control the position of a marker using his head orientation alone. For this game to be of widespread use on standard laptops and mobile devices, the only sensor that will be considered is a standard monocular web camera. The head pose will be estimated directly from the natural 2D image appearance, leaving the user completely free of any artificial markers or special glasses. Moreover, this non-intrusive system should start automatically and recover from failure automatically, without any previous knowledge of the user's appearance or location. Finally, the system should be able to track a person's head pose at a reasonable distance without any camera calibration, and output a continuous value of head orientation estimates needed for controlling the computer game.

2 Previous Work

Human head pose estimation is a hard problem because it suffers from classical problems in computer vision such as viewpoint variation, illumination variation and occlusion. Moreover, face features such as hairline, moustache, eyebrows and facial hair all vary considerably from person to person. Various approaches to head pose estimation have been proposed in the literature [1].

Algorithm 1 Fasthpe main processing steps

1. Automatic face and face feature detection via the Viola-Jones algorithm [14] (§ 3.1).
2. Face feature tracking using the Normalized sum of squared difference (NSSD) (§ 3.2).
3. Head pose estimation from the eye nose and mouth face features using a geometric approach [2] (§ 3.3).
4. Detection of tracking failure:
   if failed then go back to step 1, otherwise keep tracking features (§ 3.4).

Appearance based modelling methods generally consider the head pose as a collection of face images from multiple viewpoints. In a training phase, a model is presented with a set of face images of known pose. In an online phase, a query face image is compared to a leaned model to find the most faithful match. This can be achieved by matching in the image-space [3], or on some linear [4, 5] or non-linear [6, 7] manifold. A neural network may also be used to learn a mapping from cropped face images to a pose estimate [8]. However without interpolation, these appearance based approaches are only capable of estimating discrete pose locations. Moreover, they require large amounts of training data, and are highly sensitive to small changes in face position and image scaling [9].
On the other hand, flexible models such as active appearance models (AAM) [10] show good invariance to head localization error since they adapt to the image and find the location of the facial features. Various 3D modelling approaches assume the head to be a cylinder [11], or some generic deformable 3D head model [12]. Although precise, using 3D head models and image registration techniques are computationally intensive and require a high image resolution that could slow down real-time performance. Moreover, without some key-frame generation method [9], these models may suffer form error drift over time.
Geometric based approaches [2, 13] can potentially achieve accurate pose measurements by tracking the location of only a few facial features such as the eyes, nose, and mouth. The computational efficiency of tracking a few features and determining the head orientation from the their configuration allows real-time performance and is well suited for our application. The downside is that feature detection and tracking are critical to the accuracy of the pose estimation results, and can lead to failure when features are lost. However, with the success of recent feature detectors [14], it is possible to create a head pose estimation system which is fully automatic, user invariant, quick to set-up and easy to use. In this work, we propose to combine feature detection and tracking with the geometric face model of Gee *et. al* [2]. We demonstrate that our system, dubbed *fasthpe*, outputs continuous and reliable pose estimates in real-time (2ms) and can be used in practical applications such as gaming on a mobile device.

3 Methods

The head pose estimation system used in this work is designed around a facial model proposed by Gee *et al* [2], which needs only the eyes, nose and mouth feature locations to calculate the head pose. These feature positions are not greatly affected by facial expressions and have a low variation across users when compared to other features such as the hairline or face contours. The automatic detection and tracking of these features is described in sections 3.1 and 3.2. A simple facial model [2] subsequently utilizes the feature locations to estimate the head pose, as detailed in section 3.3. An overview of the head pose estimation algorithm is shown in Algorithm 1.

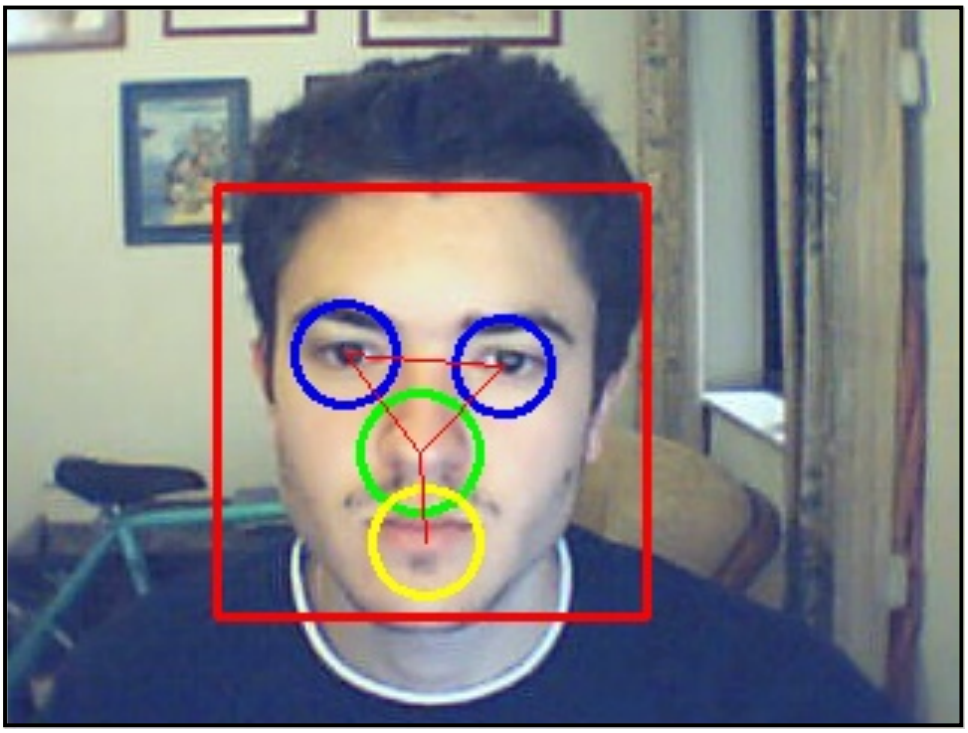

Figure 1: The first step towards automatic initialization of the head pose estimation system is the detection of the face (red) and the face features required to model the subjects head, namely the eyes (blue), nose (green), and mouth (yellow).

3.1 Face feature detection

For the head pose estimation system to start automatically the eyes, nose and mouth features used to calculate the subject's head orientation need to be detected in the video stream. The face may be found from various image cues including: skin colour [15, 16], edge patterns, image motion, and ellipse fitting [17, 18]. Face features may then be located in the face by using a simple threshold (assuming the features are the darkest regions of the face) and by imposing geometrical constraints [15]. In our method, since the feature locations are critical for this geometric based approach, the face and feature locations are found using the state-of-the-art Viola-Jones algorithm [14].

We used freely available trained cascades [19, 20] to detect the face, eyes, nose and mouth regions. These detectors have been trained on a wide variety of training images, making the detection robust to a wide variety of people with varying skin colour and general appearance.

Initially, face detection is performed to constrain the face feature locations to areas inside the face region. This operation increases the computational efficiency of feature detection by reducing the search area and by limiting false-positive detections. An example illustrating the face and face feature detection procedure is shown in Fig. 1.

The Viola-Jones method is based on the values of simple Haar-like operators, and a cascade of weak classifiers. initially, sub-windows in the query image are passed to a series of classifiers of increasing complexity as the stages progress. In this way, any non-features are quickly rejected in the early classification stages using minimal computations. Subsequent stages that eliminate further negatives use more complex classifiers and require additional computation, however these stages only focus on promising regions of the image. The early rejection of sub-windows and the efficient method for operator computation significantly increases the speed of the detector [14]. The required processing time needed to detect the face features is of approximately 50ms on a modest laptop, fast enough for the automatic initialization not to pose any unwanted burden

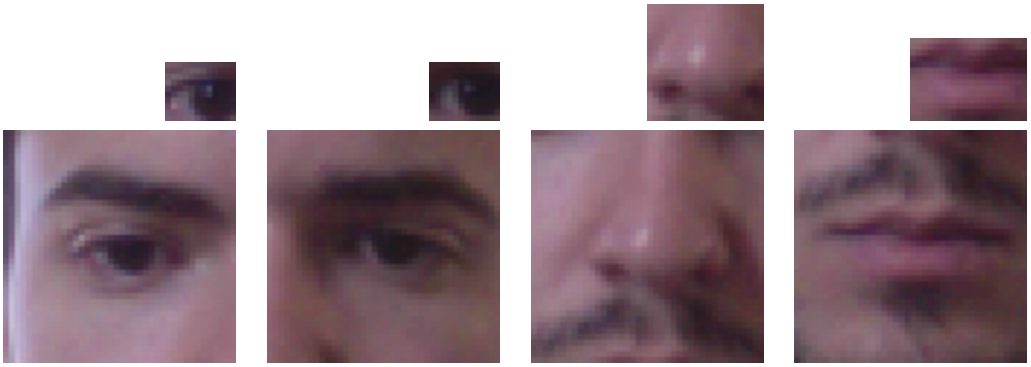

Figure 2: Template images captured immediately after the detection stage. Starting from left: left eye, right eye, nose and mouth template images and corresponding regions of interest. These templates are correlated with similar sized patches within a region of interest around the estimated feature location.

on the user.

3.2 Face feature tracking

The estimation of feature movement is based on the matching of a template image within a region of interest (ROI). The new feature location is updated to the position where the best match is found and this process is repeated to continually update the feature location in the video stream. Once the face features have been found, template images are around the detected feature locations are captured and stored for subsequent matching. The trained Harr-cascades used in this work [19, 20] have been trained on frontal feature images and cannot be used to detect the features under rotation. Therefore, for correct initialization of the system, the user simply needs to look straight at the camera.
To ensure that the template images are not initialized when the user's face is not facing the camera, the centroid of the triangle formed between the eyes and the mouth is compared to the position of the detected nose tip. If the nose tip is within a predefined distance of the centroid, then the person is considered to be in a frontal pose, and the tracking system may be initialised. The maximum distance from the centroid is set manually and allows for variations in human face appearance. Example images of captured templates and regions of interest are shown in Fig. 2.
Once the template images have been extracted, the feature motion is estimated by matching the template over an area of fixed size centred on the previous feature location. This method is uses the simple normalized sum of squared differences (NSSD) to compute image patch similarity as follows:

$$NSSD(x,y) = \frac{\sum_{y',x'} \left[T(x',x') - I(x+x',y+y')\right]^2}{\sqrt{\sum_{y',x'} T(x',y')^2 \sum_{y',x'} I(x+x',y+y')^2}}. \quad (1)$$

Where T is the template image and I is the query image patch. Using this equation, a perfect match will have a value of zero, and a mismatch will have a large sum of squared difference. The position in the query image patch which results in the smallest NSSD is regarded as the best match, and therefore the minimum of the NSSD values is the most likely position of the feature in the image.
The typical proximity between corresponding face features across frames allows the template matching to be restricted to an area around the previous feature locations. This drastically reduces the search space and increases the overall computational efficiency of the system. From

the feature tracking alone, estimates of the head translation, scale and roll may be extracted. The *translation* may be estimated form the average feature locations in the image, the *scale* may be estimated as a scale factor relative to the initial separation between the features, and the head *roll* may be estimated from the angle the eyes make with the vertical image axis.

3.3 Pose determination

Inspired by renaissance artist's depiction of gaze direction, Gee *et al.* used a simple and generic face model using measurements from only the eyes, nose and mouth. In this work we consider the 3D method proposed by [2] since this is well suited for near-frontal head orientations, where all the face features are visible. Moreover, this method is more suitable for applications in which the user is directly facing the camera of a mobile device.
The 3D face model is made up of three world lengths: L_f, L_n, and L_m which denote the *eye-mouth*, *nose-mouth*, and the *nose base-nose tip* distances, as illustrated in Fig. 3. The corresponding distances in the image are denoted by small case letters. The face is assumed to a plane passing through the eye and mouth feature locations, and the *facial normal* is found by joining the nose base to the nose tip. The symmetry axis is found by joining the midpoint between the eyes to the mouth, and the nose base is located using the model ratio $R_m = L_m/L_f$, as shown in Fig. 4. A further model ratio $R_n = L_n/L_f$ is required for the 3D model, however these two ratios need not be calibrated manually for each subject unless high accuracy is needed. The angle σ which the facial normal makes with the vector $\mathbf{d}$ normal to the image plane is called the slant, as shown in Fig. 5. This angle may be found by using the image measurements θ, l_n, ln, and the model ratio $R_n = L_n/L_f$. It can be shown that in camera centred coordinates, the facial normal $\hat{\boldsymbol{n}}$ is given by:

$$\hat{\boldsymbol{n}} = [\sin\sigma\cos\tau, \sin\sigma\sin\tau, -\cos\sigma] \tag{2}$$

We refer the reader to [2] for the supporting theory.

3.4 Recovery from failure

At some stage, due to severe occlusions, fast movements, or quick lighting variation the tracking system is bound to fail. Tracking failure is detected in two ways: i) the best matching score between the template and ROI is below a certain threshold, and ii) there is a significant change in the distance ratios between features. In the case that one feature has been lost (low correlation) and the confidence measures of the other features is high, the face's symmetry and geometrical proportions are exploited in order to constrain the search area of the lost feature. The current implementation can recover from single feature tracking failures and occlusions without having to go back to the feature detection routine, if the person has returned to a near frontal pose.
In the event that more than one feature is lost or the model ratios between features has changed significantly, the algorithm will return to the feature detection step described in section 3.1.

4 Experimental analysis

The algorithm was implemented in *c* on a 2GHz laptop that used an in-built webcam for real-time testing (§ 5). In order to test the quantitative performance of the algorithm, we test the head pose estimation system on 18 video sequences from the Boston University dataset [11]. The videos were captured from a Sony Handycam at 30 frames per second and 320×240 resolution. Ground truth pose estimates were collected via a *Flock of Birds* 3D magnetic tracker attached to the subject's head.
Two sets of videos were chosen (two subjects in nine different video clips each) that are representative of the whole range of movements in the 6 degrees of freedom (6DOF). The lighting

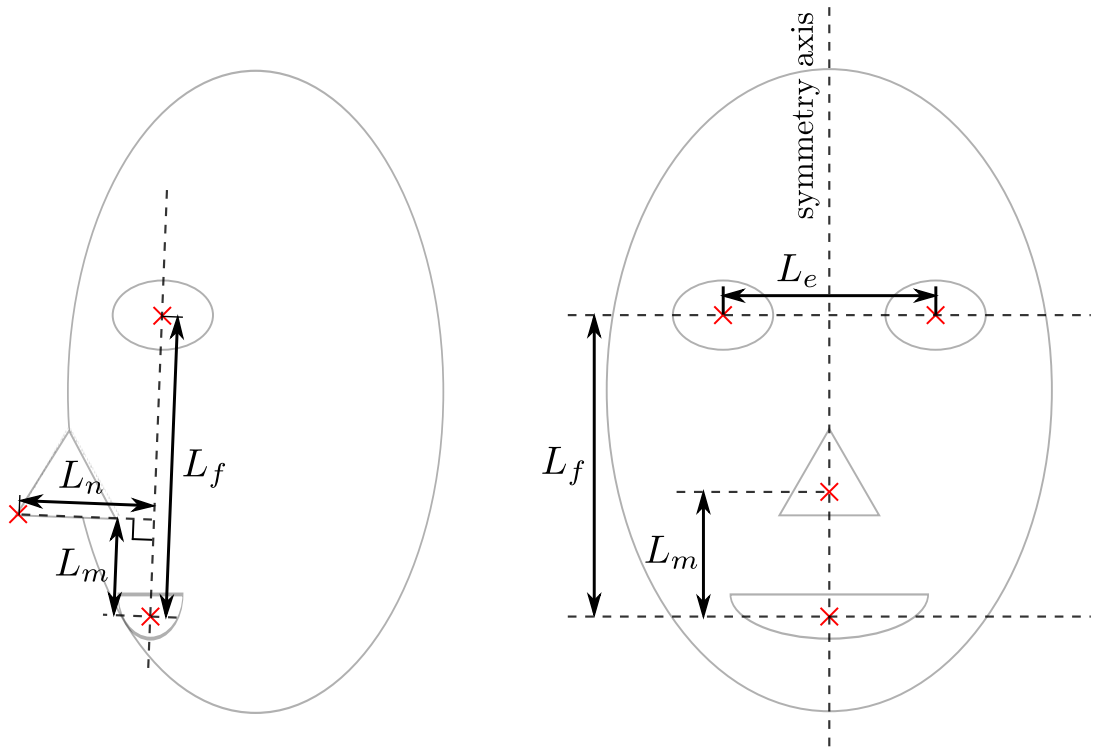

Figure 3: Profile (left) and frontal (right) views of a face. The face model is composed of distance ratios between the eye, nose and mouth positions of a typical face. The facial plane is defined by the locations of the eyes and the mouth. The pitch and yaw angles are estimated by assuming that the face is planar, and the nose is a vector normal to this plane [2].

Table 1: Average results for *jam* and *ssm* videos.

	Roll (deg)	Yaw (deg)	Pitch (deg)
jam *(MAE)*	2.74	5.34	3.31
jam *(Std)*	1.02	2.38	2.36
ssm *(MAE)*	3.31	5.20	4.49
ssm *(Std)*	2.00	2.57	3.83

conditions, skin colour, facial hair and the distance of the user from to camera differ between the two sets. The first set called the *jam* videos, features a light skin coloured subject with no facial hair in a typical office with poor lighting conditions, as shown in Fig. 6a. The second set known as the *ssm* videos, features a dark skin coloured subject with a moustache in the same setting but with good lighting conditions, as shown in Fig. 6b. In all videos, the face pixels amount to approximately 10-15% of the total frame pixels. The model parameters and the template image sizes were estimated manually from the subject's face appearance and size. For all videos, the system was allowed to start automatically as soon as the face features were detected. For all video sequences, the tracker was initialised automatically as described in section 3.1, and the head pose estimation system was allowed to run for the whole video sequence. The global results for all sequences may be seen in Table 1.

In the *jam8* video, the subject undergoes x, y, and z translations, as well as *yaw*, *pitch* and *roll* rotations, as can be seen in the frames of Fig. 6(a). The estimated angles were plotted against the ground truth, as shown in Fig. 7(a). The *roll* angle rotation achieved the least error, and this was expected since it is being extracted directly from the 2D image plane. The *yaw* rotation attained the largest error and this can be attributed to the fact that the eye appearance undergoes significant change in this rotation direction causing a poor match when compared to

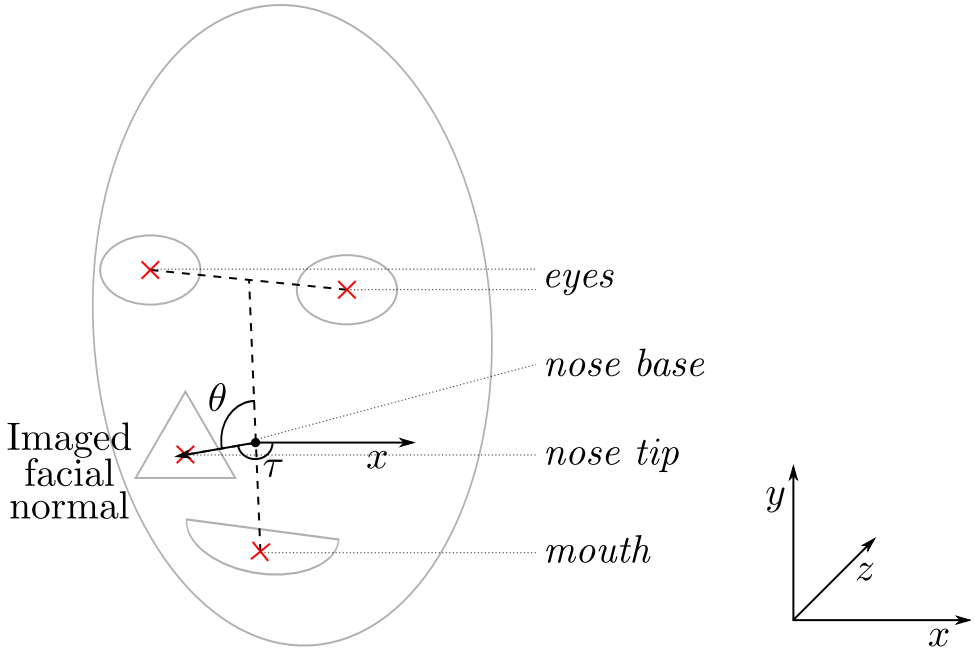

Figure 4: Illustration of a face image. The symmetry axis is found by connecting the midpoint between the eyes to the mouth, and the nose base is located on this line by using the model ratio $R_m = L_m/L_f$. The facial normal is found by joining the nose base to the nose tip. The image angles θ and τ can easily be extracted from the 2D image [2].

the fronto-parallel template captured at initialization.
In the *ssm8* video, the subject undergoes large yaw rotations, as shown in Fig.6(b). The resulting plots are shown in Fig. 7(b). The tracker was unable to track the large positive yaw rotation fully, and overestimated the roll angle towards the end of the sequence. The overall orientation of the face was still tracked despite such a low resolution of the face features. Errors in pose may also be attributed to the uncertain face model ratios of the subject, errors in automatic feature localization, and the simple camera model assumption. Despite the inherent inaccuracies of the system, the results look promising and are well within the error margin for several practical applications.

5 A practical application

In order to prove the applicability of the pose estimation system as a hands-free pointing device controlled by the users head orientation, a simple game was created. The aim of the game is to shoot down blue circles that appear in random positions on the screen with a black cross-hair style pointer controlled by the user. The position of the black cross-hair is determined from the yaw and pitch angles of the user's head, and can be moved about in real time. Once the user has positioned the black cross-hair over the blue circle, it will disappear and another one will appear in another random position on the screen. A counter then informs the user how many blue circles he has managed to hit. With a 2ms processing time on a standard laptop, the pose estimation game may run at up to 500fps, however in this case, the frame rate is limited to the 30fps afforded by the webcam. Typical screenshots of the game can be seen in the accompanying video[1].
The angles which are generated by the head pose estimation are discrete owing to the face feature locations which reside at discrete pixel locations. The possible pointing positions on

[1] http://www.youtube.com/watch?v=6MfKMT-tfMs

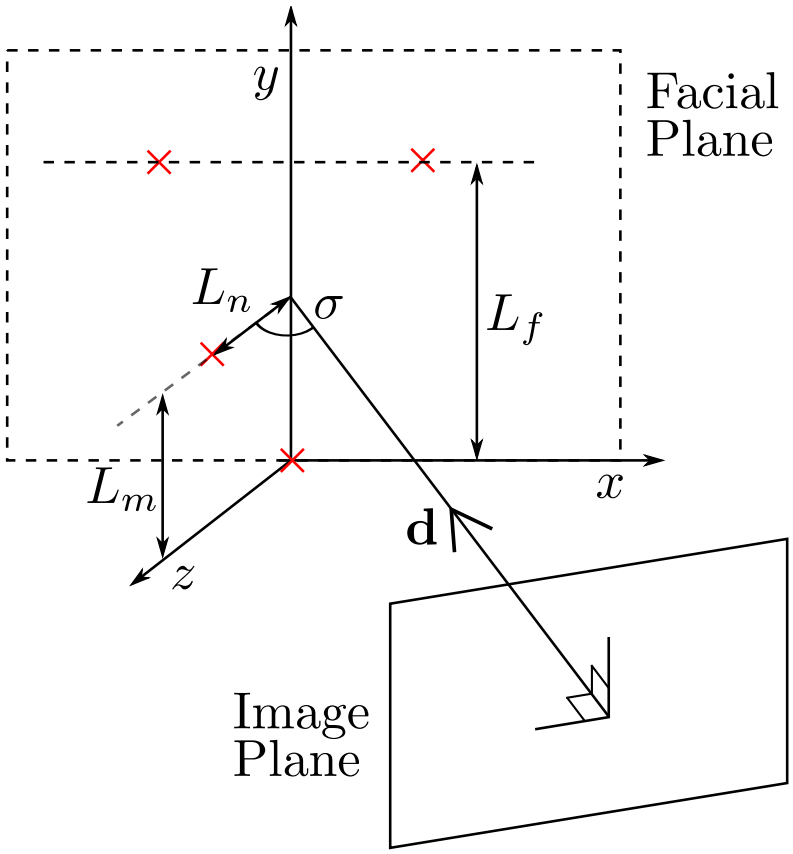

Figure 5: The face centred coordinate system used to calculate the slant angle σ. The facial slant σ is the angle between the vector d, normal to the image plane, and the facial normal located along the z axis of the facial plane [2].

the screen are therefore also discrete and will jump between locations when the user's head is undergoing a rotation. One way of increasing the resolution of the pose angles would be to increase the image resolution, however this is undesirable. Another way would be to estimate the position of the cursor in the light of the current image measurements and a simple motion model. This was achieved by using a 2D Kalman filter on the x and y positions of the cursor position and had the effect of adding a more natural motion to the cursor's movement, at the expense of adding a slight delay in response.

6 Discussion

The results from the *jam* videos consistently achieved a lower MAE when compared to the *ssm* videos as can be seen from Table 1, which indicates that a higher face resolution and proper feature localization improves the accuracy of the tracking. One must stress that the pose estimation is directly reliant on the tracking of the salient features in the face and any tracking error will immediately corrupt the pose estimates. Moreover, any inaccuracy in the face detection stage will result in a constant offset error in the pose estimation.
Overall, the 18 test videos in the BU dataset were tracked successfully and were representative of the whole range of movements in the 6 DOF. The system also showed robustness to the ethnicity, scale, and illumination variations present between the *jam* and *ssm* sequences. Global mean absolute errors of 3.03, 5.27 and 3.91 degrees were achieved for roll, yaw and pitch angles respectively. The rotation angles are all constrained to approximately ± 35 degrees with respect to the image normal, as this is the angle at which the features start becoming self occluded.

Figure 6: Video frames from the sequence (a) *jam8* and (b) *ssm8*. The automatic initialisation stage can be seen in 'Frame 1' where the detected face region is represented by a red box. The eyes, nose, and mouth are represented by blue, green, and yellow circles respectively. In the remaining frames, the estimated face feature locations have a red box drawn around them, and the pose is depicted using a drawing pin representation.

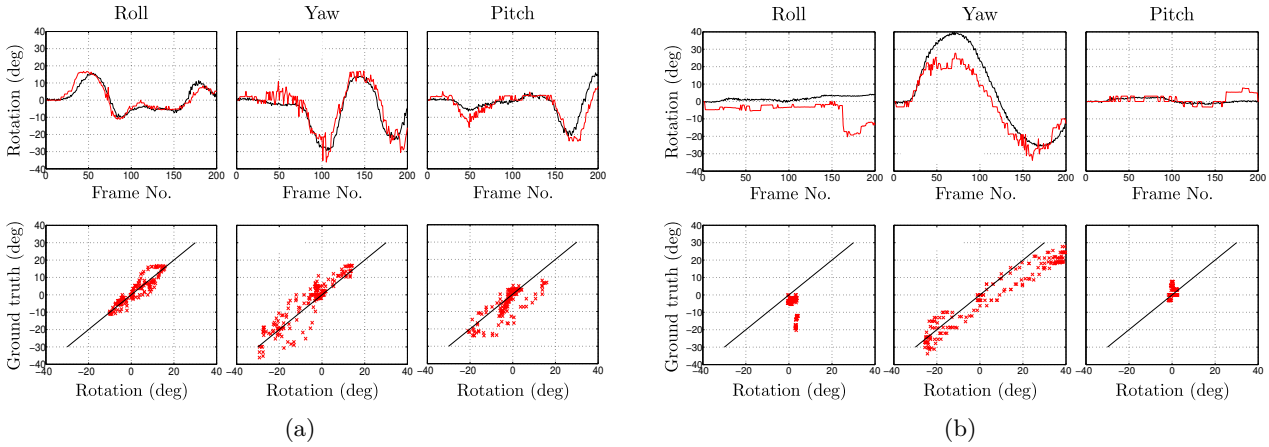

Figure 7: (a) results for video *jam8* and (b) for video *ssm8*. The first row of plots show the variation of the head rotation angle against the video frame number. The red solid line, which represents the estimated pose, should ideally follow the black solid line which represents the ground truth. The second row of plots show the ground truth plotted against the estimated pose angles. The black diagonal line shows the ideal case in which the estimated pose matches exactly to the ground truth.

These restrictions however are acceptable when playing a mobile game in which the user would need to face the camera/screen to get feedback on his/her performance. Finally a simple game was created to test the practical usability of the pose estimation system as a hands-free, head controlled pointing device. The positive results demonstrate that the system is indeed capable of being used practically by any unskilled user.

7 Conclusion

The results obtained from the head tracking and pose estimation system dubbed *fasthpe* demonstrate its applicability to a real-world gaming scenario that may operate on a mobile device. The monocular image capture device does not require calibration and easily achieves real time performance (2ms), starts automatically and recovers from failure automatically without any previous knowledge of the user's appearance or location. It can operate in a wide range of constant lighting conditions, but handling drastic changes in lighting conditions is still a problem since template images change considerably with varying illumination. In the current system,

this problem is overcome when the system detects failure and automatically re-initialises to capture new template images under the changed illumination. A simple game was created to demonstrate the practical applicability of the head pose estimation system. The use of this game proved to be entertaining, and also provided insight into the strengths and weaknesses of the system.
In future work, a head motion model can be used to improve the pose estimation by predicting the next pose position from past measurements. This would have the effect of smoothing the pose measurements and removing outliers. More robust feature tracking may be achieved by matching the template images against the regions of interest at various scales and orientations. This approach could be extended to any affine transformation which would further improve the possibility of finding a good match, thus enhancing the robustness of the tracker.

References

[1] E. Murphy-Chutorian and M. Trivedi, "Head pose estimation in computer vision: A survey," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 31, no. 4, pp. 607–626, 2009.

[2] A. Gee and R. Cipolla, "Determining the gaze of faces in images," *Image and Vision Computing*, vol. 12, no. 10, pp. 639–647, 1994.

[3] S. Niyogi and W. Freeman, "Example-based head tracking," in *Proc. 2nd Int. Conf. on Automatic Face and Gesture Recognition*, 1996, pp. 374–378.

[4] S. J. McKenna and S. Gong, "Real-time face pose estimation," *International Journal on Real Time Imaging*, vol. 4, pp. 333–347, 1998.

[5] S. Srinivasan and K. Boyer, "Head pose estimation using view based eigenspaces," in *Proc. 16th Int. Conf. on Pattern Recognition*, 2002, pp. 302–205.

[6] S. Li, F. Qingdong, G. Lie, B. Scholkopf, C. Yimin, and Z. Hongjiag, "Kernel machine based learning for multi-view face detection and pose estimation," in *Proc. 8th IEEE Int. Conf. on Computer Vision*, 2001, pp. 674–679.

[7] B. Raytchev, I. Yoda, and K. Sakaue, "Head pose estimation by nonlinear manifold learning," in *Proc. 17th IEEE Int. Conf. on Pattern Recognition*, 2004, pp. 462–466.

[8] Z. Liang, G. Pingali, and I. Carlbom, "Real-time head orientation estimation using neural networks," in *Proc. Int. Conf. on Image Processing*, 2002, pp. 297–300.

[9] L. Morency, J. Whitehill, and J. Movellan, "Generalized adaptive view-based appearance model integrated framework for monocular head pose estimation," in *Proc. Int. Conf. Automatic Face and Gesture Recognition*, 2008, pp. –.

[10] T. Cootes, C. Taylor, D. Cooper, and J. Graham, "Active shape models," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 681–685, 2001.

[11] M. L. Cascia, S. Scarloff, and V. Athitsos, "Fast, reliable head tracking under varying illumination: An approach based on registration of texture-mapped 3d models," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 4, pp. 322–336, 2000.

[12] J. Paterson and A. Fitzgibbon, "3d head tracking using non-linear optimization," in *Proc. of the British Machine Vision Conference*, 2003, pp. 609–618.

[13] T. Horprasert, Y. Yacob, and L. Davis, "Computing 3-d head orientation from a monocular image sequence," in *Proc. Int. Conf. on Automatic Face and Gesture Recognition*, 1996, pp. 242–247.

[14] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vision*, vol. 57, pp. 137–154, 2004.

[15] J. C and B. Tiddeman, "A real-time stereo head pose tracking system," in *Proc. Int. Conf. Computer Vision Theory and Applications (VISAPP)*, 2005, pp. 258–263.

[16] R. Feris, T. de Campos, and R. Junior, "Detection and tracking of facial features in video sequences," in *Lecture notes in Artificial Intelligence*, 2000, pp. 197–206.

[17] K. Huang and M. Trivedi, "Robust real-time detection, tracking, and pose estimation of faces in video streams," in *Proc. 17th Int. Conf. on Pattern Recognition*, 2004, pp. 965–968.

[18] P. Fitzpatrick, "Head pose estimation without manual initialization," Artificial Intelligence Lab, Massachusetts Institute of Technology, Technical Report, 2000.

[19] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[20] M. C. Santana, O. Déniz-Suárez, L. Antón-Canalís, and J. Lorenzo-Navarro, "Face and facial feature detection evaluation - performance evaluation of public domain haar detectors for face and facial feature detection," in *Proc. Int. Conf. Computer Vision Theory and Applications (VISAPP)*, 2008, pp. 167–172.