Accurate Single View Model-Based Head Pose Estimation

Pedro Martins, Jorge Batista*
Institute for Systems and Robotics
Dep. of Electrical Engineering and Computers, FCT-University of Coimbra, Portugal
{pedromartins,batista}@isr.uc.pt

Abstract

A framework for automatic human head pose estimation from single view images is proposed. The 6DOF head pose was estimated using Pose from Orthography and Scaling with ITerations (POSIT) where a statistical anthropometric 3D rigid model its used as an approximation of the human head, combined with Active Appearance Models (AAM) for facial features extraction and tracking. Several experiences were performed comparing results with a planar ground truth. These experiments shows that orientations and distances were, on average, found within 2° *or 1cm error standard deviations respectively.*

1. Introduction

For Human Computer Interface (HCI) applications, knowledge about face pose, $i.e.$ position and oriention, is an important issue, enabling build smart interactive systems such as: face recognition systems, teleconference, knowledge about gaze direction, video compression, etc.

This work deals with the problem of estimate the tridimensional orientation and position of faces (6DOF) using a non-intrusive system. A statistical anthropometric 3D rigid model is used with Pose from Orthography and Scaling with ITerations (POSIT) [2] algorithm for pose estimation. Since POSIT estimates pose by a set of 3D model points and 2D image projections correspondences, a way to extract facial characteristics and perform the associations is required.

A model-based approach for the interpretation of face images, Active Appearance Models (AAM) [9] its used. The AAM fitting procedure provides an effective way to locate facial features on 2D images. Given a sufficient representative training set, AAM has the key advantage of fit an unseen person, modeling non-rigid deformations fully describing facial characteristics.

Monocular head pose estimation is achieved combining tracking of facial features [1] with the approximation of the human head by an 3D anthropometric model using POSIT. Since the AAM landmark-based nature, the 3D model / 2D correspondences registration problem is easily solved.

This paper is organised as follows: section 2 gives a introduction to the standard AAM theory, section 3 describes the POSIT algorithm, section 4 explains the combined methodology used to perform human head pose estimation. Section 5 shows experimental results and section 6 describes an Augmented Reality (AR) application where 3D virtual glasses were inserted on the subjects face using the pose estimation approach described. Section 7 discusses the results.

2. Active Appearance Models

Active Appearance Models (AAM) [9] is a statistical based template matching method, where the variability of shape and texture is captured from a representative training set. Principal Components Analysis (PCA) on shape and texture data allow building a parametrized face model that fully describe with photorealistic quality the trained faces as well as unseen. For futher details refer to [8].

2.1. Shape Model

The shape is defined as the quality of a configuration of points which is invariant under Euclidian Similarity transformations [8]. This landmark points are selected to match borders, vertexes, profile points, corners or other features that describe the shape. The representation used for a single n-point shape is a $2n$ vector given by $\mathbf{x} = (x_1, y_1, x_2, y_2, \ldots, x_{n-1}, y_{n-1}, x_n, y_n)^T$. With N shape annotations, follows a statistical analysis where the shapes are previously aligned to a common mean shape using a Generalised Procrustes Analysis (GPA) removing location, scale and rotation effects. Optionally, we could project the shape distribution into the tangent plane, but omitting this projection leads to very small changes [7]. Applying a Principal Components Analysis (PCA), we can model the statistical variation with

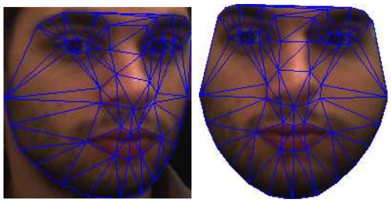
$$\mathbf{x} = \overline{\mathbf{x}} + \Phi_s \mathbf{b}_s \quad (1)$$

*This was funded by FCT Project POSC/EEA-SRI/61150/2004

where new shapes $\mathbf{x}$, are synthesised by deforming the mean shape, $\bar{\mathbf{x}}$, using a weighted linear combination of eigenvectors of the covariance matrix, Φ_s. $\mathbf{b}_s$ is a vector of shape parameters which represents the weights. Φ_s holds the t_s most important eigenvectors that explain a user defined variance.

2.2. Texture Model

For m pixels sampled, the texture is represented by the vector $\mathbf{g} = [g_1, g_2, \ldots, g_{m-1}, g_m]^T$. Building a statistical texture model, requires warping each training image so that the control points match those of the mean shape. In order to prevent holes, the texture mapping is performed using the reverse map with bilinear interpolation correction. The texture mapping is performed, using a piece-wise affine warp, $i.e.$ partitioning the convex hull of the mean shape by a set of triangles using the Delaunay triangulation. Each pixel inside a triangle is mapped into the correspondent triangle in the mean shape using barycentric coordinates, see figure 1. This procedure removes differences in texture due shape changes, establishing a common texture reference frame. The effects of differences in illumination are reduced performing a histogram equalization independently in each of the three color channels [3]. A texture model can be obtained by applying a low-memory PCA on the normalized textures,



(a) Original (b) Warped texture

Figure 1. Texture mapping example.

$$\mathbf{g} = \bar{\mathbf{g}} + \Phi_g \mathbf{b}_g \tag{2}$$

where $\mathbf{g}$ is the synthesized texture, $\bar{\mathbf{g}}$ is the mean texture, Φ_g contains the t_g highest covariance texture eigenvectors and $\mathbf{b}_g$ is a vector of texture parameters.

2.3. Combined Model

The shape and texture from any training example is described by the parameters $\mathbf{b}_s$ and $\mathbf{b}_g$. To remove correlations between shape and texture model parameters a third PCA is performed to the following data, $\mathbf{b} = \begin{pmatrix} \mathbf{W}_s \mathbf{b}_s \\ \mathbf{b}_g \end{pmatrix} = \begin{pmatrix} \mathbf{W}_s \Phi_s^T (\mathbf{x} - \bar{\mathbf{x}}) \\ \Phi_g^T (\mathbf{g} - \bar{\mathbf{g}}) \end{pmatrix}$, where $\mathbf{W}_s$ is a diagonal matrix of weights that measures the unit difference between shape and texture parameters. A simple estimate for $\mathbf{W}_s$ is to weight uniformly with ratio, r, of the total variance in texture and shape, $i.e.$ $r = \sum_i \lambda_{gi} / \sum_i \lambda_{si}$, where λ_s and λ_g are shape and texture eigenvalues, respectively. Then $\mathbf{W}_s = r\mathbf{I}$ [6]. As result, using again a PCA, Φ_c holds the t_c highest eigenvectors, and we obtain the combined model, $\mathbf{b} = \Phi_c \mathbf{c}$. Due the linear nature for the model, is possible to express shape, $\mathbf{x}$, and texture, $\mathbf{g}$, using the combined model by

$$\mathbf{x} = \bar{\mathbf{x}} + \Phi_s \mathbf{W}_s^{-1} \Phi_{c,s} \mathbf{c} \tag{3}$$

$$\mathbf{g} = \bar{\mathbf{g}} + \Phi_g \Phi_{c,g} \mathbf{c} \tag{4}$$

where $\Phi_c = \begin{pmatrix} \Phi_{cs} \\ \Phi_{cg} \end{pmatrix}$ and $\mathbf{c}$ is a vector of appearance controlling both shape and texture. An AAM instance is built by generating the texture in the normalized frame using eq. 4 and warping-it to the control points given by eq. 3.

2.4. Model Training

An AAM search seek to minimize the texture difference between a model instance and the beneath part of the target image that it covers. It can be treated as an optimization problem where $arg_{min|_{\mathbf{c}}} |\mathbf{I}_{image} - \mathbf{I}_{model}|^2$ updating the appearance parameters $\mathbf{c}$ and pose. This nonlinear problem can be solved by learning offline how the model behaves due parameters change and the correspondent relations between the texture residual [9]. Additionally, are considered the similarity parameters to represent the 2D pose, $\mathbf{t} = (s_x, s_y, t_x, t_y)^T$. To maintain linearity and keep the identity transformation at zero, these parameters are redefined to $s_x = (s\cos(\theta) - 1)$, $s_y = s\sin(\theta)$ which represents a combined scale, s, and rotation, θ, while the remaining parameters t_x and t_y are translations. The complete model parameters include also pose, $\mathbf{p} = (\mathbf{c}^T | \mathbf{t}^T)^T$. The initial AAM formulation uses the Multivariate Linear Regression (MLR) approach over the set of training texture residuals, $\delta\mathbf{g}$, and the correspondent model perturbations, $\delta\mathbf{p}$. Assuming that the correlation of texture difference and model parameters update is locally linear, the goal is to get the optimal prediction matrix, in the least square sense, satisfying the linear relation, $\delta\mathbf{p} = \mathbf{R}\delta\mathbf{g}$. Solving it involves perform a set experiences, building huge residuals matrices and perform MLR on these. It was suggested [9] that appearance parameters, $\mathbf{c}_i$, should be perturbed in about $\pm 0.25\sigma_i$ and $\pm 0.5\sigma_i$. Scale around 90%, 110%, rotation ±5°, ±10° and translations ±5%, ±10%, all with respect to the reference mean frame.

The MLR was later replaced by a simpler approach [9], computing the gradient matrix, $\dfrac{\partial \mathbf{r}}{\partial \mathbf{p}}$, requiring much less memory and computational effort. The texture residual vector is defined as $\mathbf{r}(\mathbf{p}) = \mathbf{g}_{image}(\mathbf{p}) - \mathbf{g}_{model}(\mathbf{p})$, where the goal is to find the optimal update at model parameters to minimize $|\mathbf{r}(\mathbf{p})|^2 = \mathbf{r}^T\mathbf{r}$. Expanding the texture residuals, $\mathbf{r}(\mathbf{p})$, in Taylor series around $\mathbf{p}$ and holding the first order terms, $\mathbf{r}(\mathbf{p}+\partial\mathbf{p}) \approx \mathbf{r}(\mathbf{p}) + \mathbf{J}\partial\mathbf{p}$ where $\mathbf{J} = \dfrac{\partial\mathbf{r}(\mathbf{p})}{\partial\mathbf{p}}$ is the Jacobian matrix. Differentiating $w.r.t$ $\mathbf{p}$ and equalling to zero leads to $\partial\mathbf{p} = -(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{r}$. Normally Steepest Descent (SD) approaches require the Jacobian evaluation for each iteration. Since the AAM framework works on a normalize reference frame, the Jacobian matrix can be considered fixed over the training set and can be estimated once on the training phase.

2.5. Iterative Model Refinement

The model parameters are updated over texture residuals by,

$$\mathbf{p}_k = \mathbf{p}_{k-1} - \alpha(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\delta\mathbf{g} \tag{5}$$

which is a damped Gauss-Newtow modification on Steepest Descent methods where $\mathbf{J}$ is the Jacobian matrix and α is the damping factor. Starting with a given estimate for the model, $\mathbf{p}_0$, and a rough estimate of the location of the face. An AAM model can be fitted following the algorithm 1. As better is the initial estimate minor the risk of being trap in a local minimum, in this work AdaBoost [10] method its used. Figure 2 shows a successful AAM search.

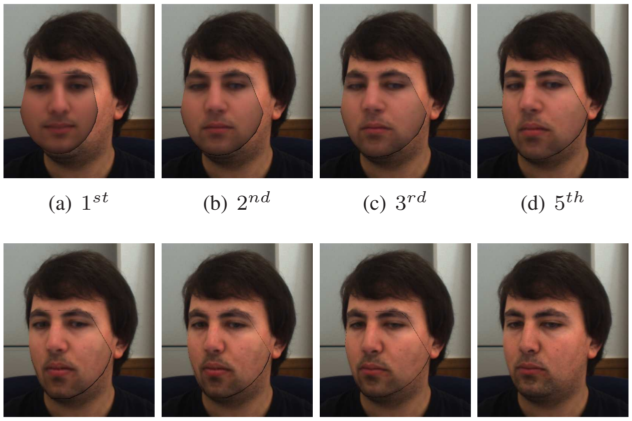
Algorithm 1 Iterative Model Refinement.

```
while MaxIterations reached or no improvement is made to error $E_0$ do
    Sample image at $(\mathbf{x},\mathbf{y}) \rightarrow \mathbf{g}_{image}$
    Build an AAM instance AAM($\mathbf{p}$) $\rightarrow (\mathbf{x}_{model}, \mathbf{y}_{model}, \mathbf{g}_{model})$
    Compute residual $\delta\mathbf{g} = \mathbf{g}_{image} - \mathbf{g}_{model}$
    Evaluate Error $E_0 = |\delta\mathbf{g}|^2 = \delta\mathbf{g}\delta\mathbf{g}^T$
    Predict model displacements $\delta\mathbf{p} = (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\delta\mathbf{g}$
    Set $\alpha = 1$
    Update Model Parameters $\mathbf{p}_k = \mathbf{p}_{k-1} - \alpha\delta\mathbf{p}$
    Update sample control points from $(\mathbf{x}_{model}, \mathbf{y}_{model})$ with similarity pose
    correction $\rightarrow (\mathbf{x}_k, \mathbf{y}_k)$
    Sample image at $(\mathbf{x}_k, \mathbf{y}_k) \rightarrow \mathbf{g}_{image_k}$
    Compute residual $\delta\mathbf{g}_k = \mathbf{g}_{image_k} - \mathbf{g}_{model}$
    Evaluate Error $E_k = \delta\mathbf{g}_k\delta\mathbf{g}_k^T$
    if $E_k < E_0$ then
        Accept model parametes, $\mathbf{p}_k$
        Accept control points $(\mathbf{x},\mathbf{y}) = (\mathbf{x}_k, \mathbf{y}_k)$
        Update current error $E_0 = E_k$
    else
        Try $\alpha = 1.5, \alpha = 0.5, \alpha = 0.25, \alpha = 0.125$
    end if
end while
```

3. POSIT

Pose from Orthography and Scaling with ITerations (POSIT) [2] is a fast and accurate, iterative algorithm for finding the 6DOF pose (orientation and translation) of a 3D model or scene with respect to a camera given a set of 2D image and 3D object points correspondences.

Figure 3 shows the pinhole camera model, with its center of projection O and image plane at the focal length f (focal length and image center are assumed to be known). Unit vectors in the camera frame are i, j and k. A 3D model with feature points $M_0, M_1 \ldots, M_i, \ldots, M_n$ is positioned at camera *frustrum*. The model coordinate frame is centered at M_0 with unit vectors u, v and w. A M_i point has known coordinates (U_i, V_i, W_i) in the model frame and unknown coordinates (X_i, Y_i, Z_i) in the camera frame. The projections of M_i are known and called m_i, having image coordinates (x_i, y_i). The pose matrix $\mathbf{P}$ gives the rigid transformation between the model and the camera frame



(a) 1^{st} (b) 2^{nd} (c) 3^{rd} (d) 5^{th}

(e) 8^{th} (f) 10^{th} (g) Final (h) Original

Figure 2. Iterative model refinement.

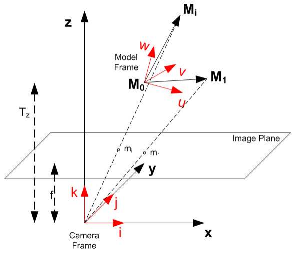


Figure 3. Perspective projections m_i for model points M_i.

$$\mathbf{P} = \left[\begin{array}{c|c} \mathbf{R} & \mathbf{T} \\ \hline 0 & 1 \end{array}\right] = \begin{bmatrix} i_u & i_v & i_w & T_x \\ j_u & j_v & j_w & T_y \\ k_u & k_v & k_w & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{P}_1 \\ \hline \mathbf{P}_2 \\ \hline \mathbf{P}_3 \\ \hline \mathbf{P}_4 \end{bmatrix}. \tag{6}$$

$\mathbf{R}$ is the rotation matrix representing the orientation of the camera frame with respect to the model frame, $\mathbf{T} = (T_x, T_y, T_z)$ is the translation vector from the camera center to the model frame center. $\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$ and $\mathbf{P}_4$ are defined as the pose matrix rows. The rotation matrix $\mathbf{R}$ is the matrix whose rows are the coordinates of the unit vectors (i, j, k) of camera frame expressed in the model coordinate frame (M_0u, M_0v, M_0w). $\mathbf{R}$, transforms model coordinates of vectors M_0M_i into coordinates defined in the camera system, for instance, the dot product $M_oM_i \cdot i$ between the vector M_0M_i and the first row of the rotation matrix, provides the projection of this vector on the unit vector of the camera system, $i.e.$ the coordinate X_i. To full compute $\mathbf{R}$ is only needed to compute i and j since $k = i \times j$. The translation vector $\mathbf{T}$ is the vector OM_0, has coordinates (X_0, Y_0, Z_0) and is aligned with the vector Om_o, so, $\mathbf{T} = \frac{Z_0}{f} Om_0$. To compute the model translation form the camera center its just need Z_0 coordinate. Knowing i, j and Z_0 the model pose becomes fully defined.

In a perspective projection model, a 3D point (X_i, Y_i, Z_i) is projected in the image by $(x_i, y_i) = (f\frac{X_i}{Z_i}, f\frac{Y_i}{Z_i})$. Under *weak perspective* (or also known *scaled orthographic*) projection model which make the assumption that the depth of an object is small compared to distance of the object from the camera, and that visible scene points are close to the optical axis [5], a 3D image point projection can be written as $(x_i, y_i) = (\frac{f}{(1+\epsilon)}\frac{X_i}{T_z}, \frac{f}{(1+\epsilon)}\frac{Y_i}{T_z})$.. In scaled orthographic projection, a vector M_0M_i in the model frame is projected by an orthographic projection over the plane $z = T_z$ followed by a perspective projection. The projected vector in the image plane has a scaling factor equals to $\frac{f}{Z_0}$.

3.1. Fundamental Equations

Defining the 4D vectors $\mathbf{I} = \frac{f}{T_z}\mathbf{P}_1, \mathbf{J} = \frac{f}{T_z}\mathbf{P}_2$ and knowing that $(1+\epsilon_i) = \frac{Z_i}{T_z}$, the fundamental equations that relate the row vectors $\mathbf{P}_1, \mathbf{P}_2$ of the pose matrix, the coordinates of the model features M_0M_i and the coordinates (x_i, y_i) from the correspondent images m_i are

$$M_0M_i \cdot \mathbf{I} = x'_i, \qquad M_0M_i \cdot \mathbf{J} = y'_i \tag{7}$$

with

$$\mathbf{I} = \frac{f}{T_z}\mathbf{P}_1, \qquad \mathbf{J} = \frac{f}{T_z}\mathbf{P}_2 \tag{8}$$

$$x'_i = x_i(1+\epsilon_i), \qquad y'_i = y_i(1+\epsilon_i) \tag{9}$$

and

$$\epsilon_i = \mathbf{P}_3 \cdot M_0M_i/T_z - 1. \tag{10}$$

If values are given for ϵ_i, eqs. 7 provide a linear system of equations with unknowns $\mathbf{I}$ and $\mathbf{J}$. Unit vectors i and j are found by normalizing $\mathbf{I}$ and $\mathbf{J}$. T_z is obtained by the norms of either $\mathbf{I}$ and $\mathbf{J}$. This approach is called Pose from Orthography and Scaling (POS) [2], $i.e.$ finding pose for fixed values of ϵ_i. Once i and j have been computed, more refined values for ϵ_i can be found using again POS. The steps of this iterative approach called POSIT (POS with Iterations) [2] are described in algorithm 2.

This method does not require an initial pose estimate, is very fast (it converges in about four iterations) and robust with respect to image measurements and camera calibration errors. The image registration problem, $i.e.$ the image and model points correspondences will be a straightful problem as will be shown in the following section.

Algorithm 2 POSIT

```
$\epsilon_i$ = best guess, or $\epsilon_i = 0$ is no pose information available
loop
    Solve for I and J: $M_0M_i \cdot \mathbf{I} = x'_i$ and $M_0M_i \cdot \mathbf{J} = y'_i$
    with $x'_i = x_i(1+\epsilon_i)$ and $y'_i = y_i(1+\epsilon_i)$
    $T_z = \frac{\|\mathbf{I}\|+\|\mathbf{J}\|}{2}$
    $\mathbf{P}_1 = \frac{T_z}{f}\mathbf{I}$; $\mathbf{R}_1 = (\mathbf{I}_1, \mathbf{I}_2, \mathbf{I}_3)$
    $\mathbf{P}_2 = \frac{T_z}{f}\mathbf{J}$; $\mathbf{R}_2 = (\mathbf{J}_1, \mathbf{J}_2, \mathbf{J}_3)$
    $\mathbf{R}_3 = \frac{\mathbf{R}_1}{\|\mathbf{R}_1\|} \times \frac{\mathbf{R}_2}{\|\mathbf{R}_2\|}$
    $\mathbf{P}_3 = [\mathbf{R}_3|T_z]$
    $\epsilon_i = \mathbf{P}_3 \cdot M_0M_i/T_z - 1$
    if $\epsilon_i \approx \epsilon_{i-1}$ then
        $\mathbf{P}_4 = (0, 0, 0, 1)$
        Exit Loop
    end if
end loop
```

4. Head Pose Estimation

The full automatic framework for head pose extraction is composed by the two parts previously described. An AAM model fitting is performed on a subject leading to shape model landmarks location tracking over time. Notice that no temporal filter is used. Since the model fitting, in certain occasions, fails especially when the movements of the head were particularly quick and a model is assumed a failure if any of the appearance parameters don't follow $-3\sigma_i \leq \mathbf{c}_i \leq 3\sigma_i$. The recovery from lost track is overcome by reinitialize the appearance-based detection process each time the model fails.

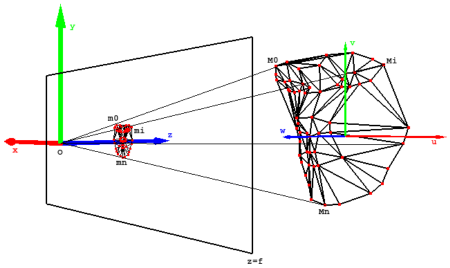


Figure 4. Anthropometric head used as POSIT 3D model.

The head pose estimation is performed using POSIT. As 3D model an anthropometric 3D rigid model of the human head, see figure 4, its used, since it is the best suitable rigid body model that describe the 3D face surface of several individuals. It was acquired by a frontal laser 3D scan of an physical model, selecting the equivalent 3D points of the AAM annotation procedure creating a sparse 3D model. Figure 5 illustrates this procedure.

Tracking features in each video frame combined with the landmark-based nature of AAMs, the image/3Dmodel registration problem required for the use of POSIT is easily solved.

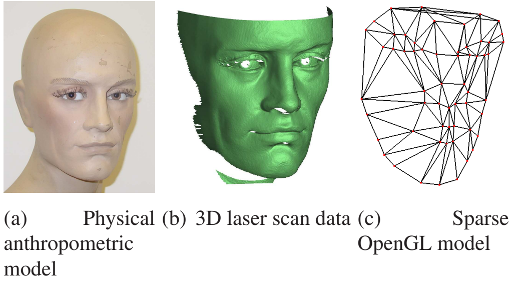
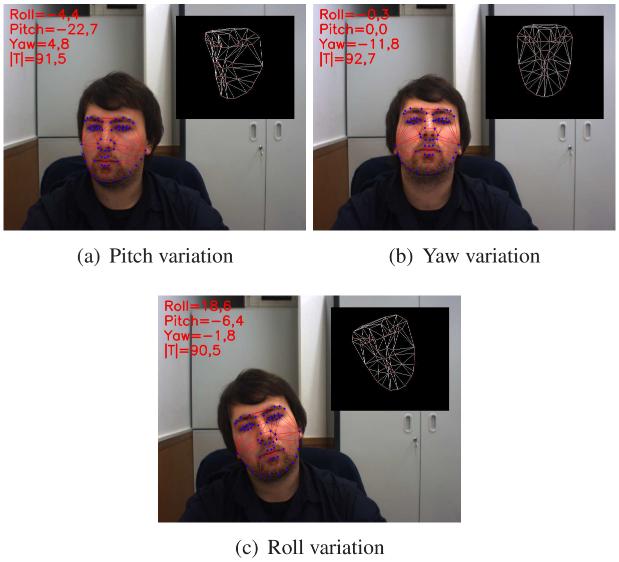

(a) Physical anthropometric model (b) 3D laser scan data (c) Sparse OpenGL model

Figure 5. a) Physical model used. b) Laser scan data acquired c) OpenGL built model using the AAM shape features.

5. Experimental Results

The orientation of the estimated pose is represented by the Roll, Pitch and Yaw (RPY) angles. Figure 6 shows some examples of pose estimation where the pose is represented by an animated 3DOF rotational OpenGL model showed at images top right. This model, used only for display, follows the subject head rotations, ignoring translation effects.



(a) Pitch variation (b) Yaw variation

(c) Roll variation

Figure 6. Example of pose estimation.

The evaluation of pose estimation accuracy is performed comparing the pose estimated with the one estimated from a planar checkerboard [4], used as ground truth reference values. Figure 7 presents results from the pose estimated during a video sequence where the subject performs several human head movements, ranging from yaw, pitch and roll head rotations of several degrees (during a total of 140 frames). The experience began by rotating head left, changing pitch angle, and recovering to frontal position, followed by a yaw angle, moving head up and down and again recovering to frontal position, and finally performing a head roll rotation. Near the end, after frame 95 the distance from camera is also changed. The individual parameters (Pitch, Yaw, Roll and distance) results are presented in figure 7-a, 7-b, 7-c and 7-d respectively. The graphical results show some correlations between Pitch and Yaw angles that result from the differences between the subject and the rigid 3D anthropometric model used. Table 1 displays the errors average standard deviations of the pose parameters for six similar performed experiences with different individuals.

Table 1. Errors standard deviations. The angle parametes are in degrees and the distance in centimeters.

Param.	Experiences error std						Avg std
Roll	1.92	1.86	1.87	2.15	2.14	1.69	1.95°
Pitch	1.91	2.46	2.10	2.94	3.23	2.81	2.57°
Yaw	3.0	1.47	1.48	1.64	1.49	1.14	1.70°
Distance	1.29	1.72	1.37	1.50	1.30	0.85	1.33cm

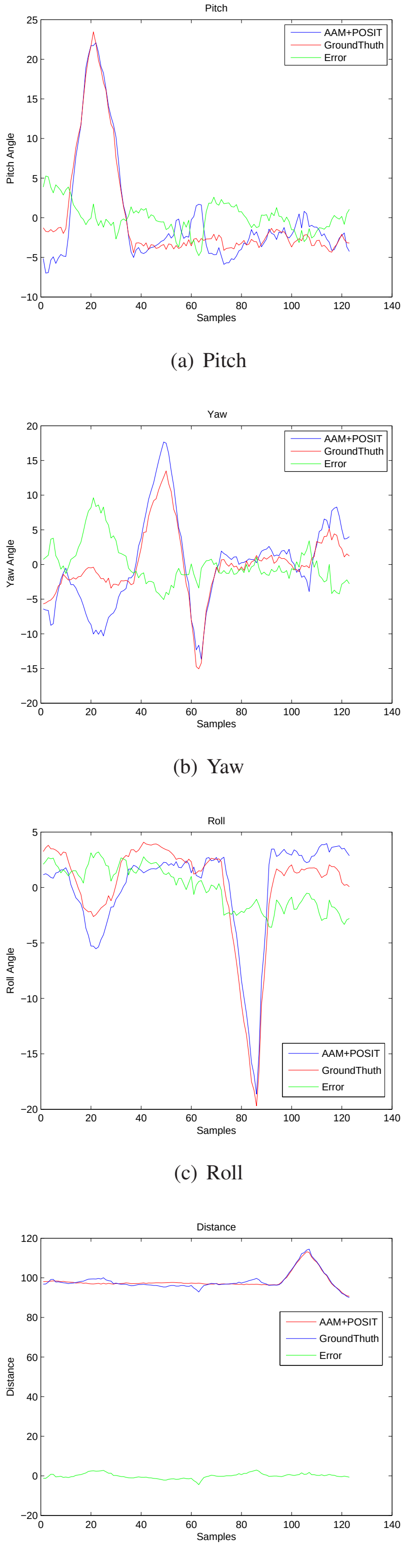
The application with AAM model fitting combined with POSIT for pose estimation runs at 5 frames/s on 1024×768 images using a 3.4 GHz P4 Intel Processor under Linux OS. AAM is based on a 58 landmark shape points ($N = 58$), sampling 48178 pixels with color information ($m = 48178 \times 3 = 144534$) by OpenGL hardware-assisted texture mapping using a Nvidia GeForce 7300 graphics board.

6. Augmented Reality Application

In Augmented Reality (AR) applications pose estimation is a critical issue. As accurate is the pose estimation, better is the model backprojection on the target image. Supported by head pose estimation approach described, an AR system where virtual glasses were inserted on the subjects face was develop. The monocular pose estimation system extracts the pose from the head model, animating the rigid 3D anthropometric model in a OpenGL application with 6DOF. Mapping the capture image combined with the model animation, a system where the rigid face model reacts to the user pose variation is achieved. Including a 3D glasses model requires drawing-it with respect to the head model. Similarly, the final AR application consists on animating only the glasses model. Image 8 shows several views of the subject with glasses augmentation with different head poses.

7. Conclusions

This work describes a single view solution to estimate the head pose of human subjects combining AAM and POSIT. AAM extracts in each image frame the landmarks position. These selected features are tracked over time and used in conjunction with POSIT to estimate head pose. Required the use of a 3D rigid model, a statistical anthropometric model is selected since is the most suitable one.



(a) Pitch

(b) Yaw

(c) Roll

(d) Distance

Figure 7. Angle Results.

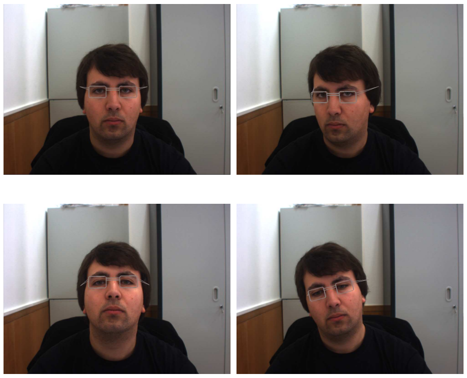


Figure 8. 3D glasses augmentation.

One of the major advantage of using combined AAM plus POSIT is that it solves directly the correspondences problem, avoiding the use of registration techniques. An accurate pose estimation is achieved with average standard deviations about 2 degrees in orientation and 1 centimeter in distance and subjects exhibiting a normal expression. The facial expression influence on pose estimation will be analyzed on future work.

References

[1] J. Ahlberg. An active model for facial feature tracking. *EURASIP Journal on Applied Signal Processing*, 2002. 1

[2] D. DeMenthon and L. Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 1995. 1, 3, 4

[3] G. S. G. Finlayson, S. Hordley and G. Y. Tian. Illuminant and device invariant color using histogram equalization. In *Pattern Recognition*. 2

[4] V. Lepetit and P. Fua. Monocular model-based 3d tracking of rigid objects: A survey. *Foundations and Trends in Computer Graphics and Vision*, 2005. 5

[5] R. D. Philip David, Daniel DeMenthon and H. Samet. Simultaneous pose and correspondence determination using line features. In *Computer Vision and Pattern Recognition*. 4

[6] M. B. Stegmann. Active appearance models theory, extensions & cases. Master's thesis, IMM Technical Univesity of Denmark, 2000. 2

[7] M. B. Stegmann and D. D. Gomez. A brief introduction to statistical shape analysis. Technical report, Informatics and Mathematical Modelling, Technical Univesity of Denmark, 2002. 1

[8] T.F.Cootes and C.J.Taylor. Statistical models of appearance for computer vision. Technical report, Imaging Science and Biomedical Engineering - University of Manchester, 2004. 1

[9] G. E. T.F.Cootes and C.J.Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2001. 1, 2

[10] P. Viola and M. Jones. Rapid object detection using a boosted cascate of simple features. In *Computer Vision and Pattern Recognition*. 3