EURASIP Journal on Applied Signal Processing 2002:10, 1039–1052

Model-Based Real-Time Head Tracking

Jacob Ström

Multimedia Technologies Department, Ericsson Research, Torshamnsgatan 23, 164 86 Stockholm, Sweden
Email: jacob.strom@era.ericsson.se

Received 31 August 2001 and in revised form 29 May 2002

This paper treats real-time tracking of a human head using an analysis by synthesis approach. The work is based on the Structure from Motion (SfM) algorithm from Azarbayejani and Pentland (1995). We will analyze the convergence properties of the SfM algorithm for planar objects, and extend it to handle new points. The extended algorithm is then used for head tracking. The system tracks feature points in the image using a texture mapped three-dimensional model of the head. The texture is updated adaptively so that points in the ear region can be tracked when the user's head is rotated far, allowing out-of-plane rotation of up to 90° without losing track. The covariance of the x- and the y-coordinates are estimated and forwarded to the Kalman filter, making the tracker robust to occlusion. The system automatically detects tracking failure and reinitializes the algorithm using information gathered in the original initialization process.

Keywords and phrases: face tracking, modeling, real-time, EKF, structure from motion, planar objects, analysis by synthesis.

1. INTRODUCTION

Automatic tracking and modeling of human faces from image sequences is an important and challenging task in computer vision. Applications include face recognition, model-based coding for video conferencing, avatar control, and computer graphics. The main goal of this paper is to present a tracking system built on the extended Kalman filter based SfM algorithm in [1], thus extending the foundations of Azarbayejani and Pentland and Jebara and Pentland [2] to achieve robust performance. Since points on the face might lie in a nearly planar constellation, the stability of the SfM algorithm is investigated for planar surfaces. The theory of Triggs for SfM of planar objects will be used as a starting point, and simulations on both noise free and noisy data are carried out to investigate how often the algorithm converges. The results are compared to general three-dimensional objects. The algorithm is also extended to handle new points, that is, points that are not visible in the first frame. This poses a problem since the first frame is used as a reference frame in the error function that the Kalman filter is minimizing. Three ways to handle these new points are investigated, and the resulting solution is to keep the old reference frame for the old points and use the new reference frame for the new ones.

The results will be applied to a face tracking system. The core idea is to select a dense set of feature points (essentially, optical flow at all the most information bearing points). Figure 1 illustrates how the system works. Patches around the feature points taken from the rendered three-dimensional model (lower left corner) are matched against the incoming video, and the two-dimensional trajectories of these feature points are then fed through an extended Kalman filter (EKF) to update the pose information of the three-dimensional model. In addition, the projection of the estimated structure serves as a starting point for the tracking in the next frame. The three-dimensional model helps in several ways. First, it compensates for rotation and scale, making it possible to use fast two-dimensional block-matching to track the feature points. Second, it helps in assessing how reliably a certain feature point can be tracked. For instance, a feature point depicted at a steep angle to the camera is hard to track and a feature point on the back side of the model cannot be tracked at all. The estimate of the reliability of a point can then be forwarded to the SfM Kalman filter as a covariance of the noise in the two-dimensional point measurements. Third, by tracking features on the side of the head, the system can cope with large out-of-plane rotations.

1.1. Previous work

The literature on head tracking is rich. Black and Yacoob [3] model the face as a plane in three-dimensional space. By comparing the optical flow of the image and that of the planar model, the face can be tracked. Basu et al. [4] use an ellipsoid model instead of a planar one and report improved results. To get a better fit to the head than what an ellipsoid can provide, Zhang and Kambhamettu [5] use an extended superquadric. Many authors use three-dimensional triangle meshes to model the head. In [6], Roivainen uses the CANDIDE model [7] to parameterize the optical flow. Using a feedback system, Roivainen avoids drift and can also resolve

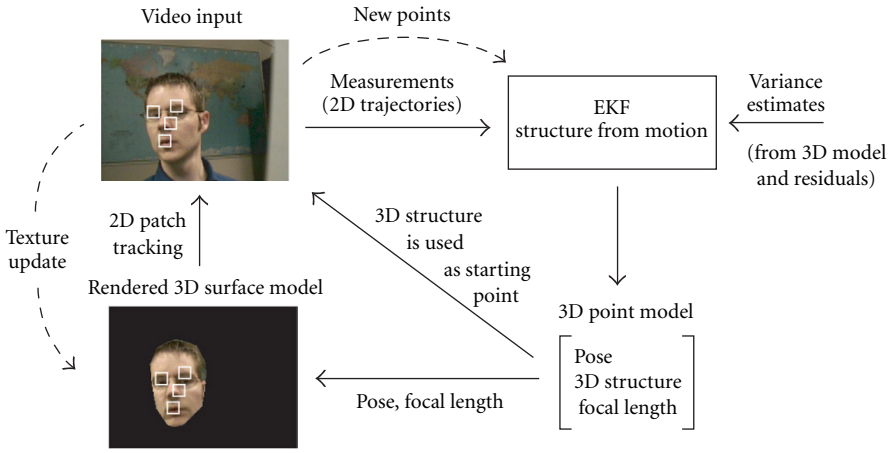


Figure 1: Patches from the rendered image (lower left corner) are matched with the incoming video. The two-dimensional feature point trajectories are fed through the SfM extended Kalman filter that estimates the pose information needed to render the next model view. For clarity, only four patches out of 24 are shown.

local motion. Li et al. [8] extend the system of Roivainen to work directly on image gradients, without the need to calculate the optical flow. In a later paper, Li and Forchheimer [9] modify their algorithm to include M-estimation, which is a robust version of the least squares algorithm. DeCarlo and Metaxas [10] use a very detailed parameterized head model constructed using anthropological data. The parameters are changed using a deformable model framework, where edges in the image give rise to forces on the parameters. Cootes et al. [11] use Active Appearance Models (AAMs), where a face is modeled using a combined eigenspace for texture and shape. By learning a function from the residual error to the eigenspace parameters, the algorithm can be iterated on a face image until convergence. Cootes et al. have used their algorithm on image sequences, where the shape from the previous frame is used as a starting value in the next one. Cascia et al. [12] use a three-dimensional version of the Active Appearance Model method. Instead of modeling the shape using a two-dimensional PCA, a cylinder model of the head is used, and an illumination eigenspace is added to the error function to gain robustness. Cascia et al. reports real-time performance (15 Hz) on an SGI O2 machine. Ahlberg [13] uses the three-dimensional CANDIDE model [7] for face tracking using AAMs in real time.

The method presented in this paper is a continuation of the work by Azarbayejani and Pentland [1], Jebara and Pentland [2], and Ström et al. [14]. In their SfM paper [1], Azarbayejani and Pentland show a head tracking application. The feature points are obtained by normalized correlation, and the tracker resolves rotation and translation of the head as well as point depths and focal length. Jebara and Pentland [2] extend the work to a real-time system, and use the estimated three-dimensional structure as a starting point for the normalized correlation in the next frame. This greatly enhances the robustness since an incorrectly tracked feature point can be corrected by the other points, so that its starting position in the next frame will be reasonable. Jebara and Pentland also estimate the covariance of the noise in the measured point positions from the residual error, which gives robustness to partial occlusion. Furthermore, each point is tracked for rotation and scale, doubling the degrees of freedom in the measurements. The work by Ström et al. [14] adds a three-dimensional head model, which is useful for (a) projectively transforming all templates that are used in the normalized correlation at once, reducing the cost of adding another point, and (b) predicting when points on the face are turning away from the camera or are self occluded. This paper further extends the work in [14] by incrementally updating the texture to track points on the side of the head. In this manner, the head can rotate farther before losing track.

Azarbayejani and Pentland [1] evaluate the stability of the SfM algorithm for general three-dimensional objects for different types of motion at various noise levels. No attempt is made to evaluate the algorithm with planar objects. Indeed, this is an important special case, since points on the surface of a face can be close to planar. Furthermore, many SfM algorithms have shown to fail for planar objects [15, 16, 17]. Triggs provides a theory for how many frames are needed to solve the SfM problem in the planar case, and also proposes an algorithm for autocalibration [18]. Tentative results on the stability of the algorithm of Azarbayejani and Pentland have been presented in [19], but only for the noise-free case. The reinitialization algorithm was first proposed in [20].

The method for adding new feature points to the algorithm of Azarbayejani and Pentland was presented in a technical report in [21]. Independently, Dell'Acqua et al. [22] presented a solution where multiple Kalman filters are running simultaneously, filtering different subsets of the points. The Kalman filters are then merged together to one Kalman filter. While interesting, the solution by Dell'Acqua et al. is of batch type and is thereby not directly applicable to real-time tracking.

1.2. Overview

Section 2 will recapitulate the SfM algorithm, investigate its stability for planar surfaces and extend it to allow new points. Section 3 will go through the tracking; initialization, individual point tracking, estimation of the measurement noise covariance and texture update. Section 4 will treat how the system is reinitialized once it loses track. This is followed by a system evaluation in Section 5. The paper is concluded in Section 6.

2. KALMAN FILTER BASED SfM

Azarbayejani and Pentland reformulate the SfM problem into a stable recursive estimation problem that has been shown to converge reliably [1, 23]. A three-dimensional point is parameterized with its image coordinates in the first frame, (u^0, v^0), and its depth α, using

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} (1+\alpha\beta)u^0 \\ (1+\alpha\beta)v^0 \\ \alpha \end{bmatrix}, \tag{1}$$

where β is 1/focal length. This is shown in the dashed line in Figure 2. The unknown parameters constitute the state vector $\mathbf{x}$; the rotation and translation from the first frame to the current, the inverse focal length β and the point depths α. To predict the image plane projection of the points in the kth frame, the function $(\hat{u}_1, \hat{v}_1, \hat{u}_2, \hat{v}_2, \ldots, \hat{u}_N, \hat{v}_N) = \mathbf{h}_k(\bar{\mathbf{x}})$ is used. As shown in Figure 2, the effect of $\mathbf{h}_k(\bar{\mathbf{x}})$ is to calculate (X, Y, Z) using (1) and then rotate, translate, and finally project the points using the parameters from the state vector $\mathbf{x}_k$. If $\mathbf{z}_k$ is the measured image coordinates of the feature points, the movement, the focal length, and the depths can be updated using

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k-1} + K_k(\mathbf{z}_k - \mathbf{h}_k(\hat{\mathbf{x}}_k)), \tag{2}$$

where K_k is the Kalman gain [24].

2.1. Planar objects

Planar objects are important for two reasons. First, man-made structures contain many planar objects such as walls, floors, desks, and so forth. Second, some objects might be close to planar. In the case of face tracking for instance, a small number of randomly picked points on the surface of the face might be close to coplanar. If the SfM algorithm fails for planar objects, it is likely to do poor on such nearly planar objects. Furthermore, it is a well-known fact that planar objects constitute a singular case that many SfM algorithms have problems with. For instance, it is not possible to calculate the coefficients of either the bifocal [17] or the trifocal tensor [15] for planar objects in the uncalibrated case. Stein and Shashua [16] show that problems also occur for objects made up of several planes that intersect in a single line, and also point to several real world objects that fit this description. Triggs shows that when n intrinsic camera parameters are unknown (but constant), $m = \lceil (n+4)/2 \rceil$ images are needed for uniqueness. In the uncalibrated case when $n = 5$, $m = \lceil (5+4)/2 \rceil = 5$ images are thus needed for uniqueness. If fewer intrinsic camera parameters are estimated, it should still be possible to use multilinear constraints. If, for instance, only the (constant) focal length is unknown, $m = \lceil (1+4)/2 \rceil = 3$ images are sufficient, and the trifocal tensor could be used. However, the constraints on the remaining four intrinsic camera parameters turn into polynomial constraints on the tensor coefficients that are not easily solved.

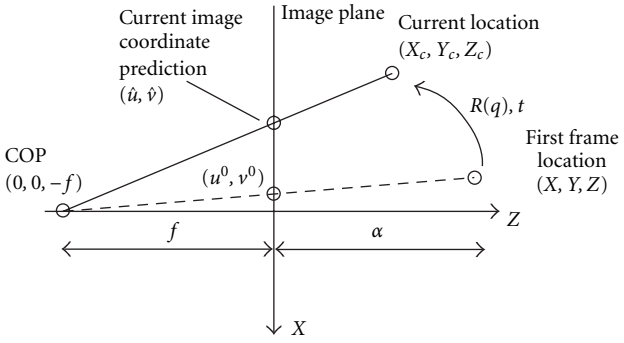


FIGURE 2: Using the original frame coordinates (u^0, v^0), the depth α and the focal length $1/\beta$, (X, Y, Z) is obtained. This vector is further moved to (X_c, Y_c, Z_c) and finally projected to the image coordinate prediction $(\hat{u}, \hat{v})$.

2.2. Several views of a planar object

The EKF approach to the SfM problem does not rely on only two or three images, but instead it takes an entire sequence of images into account. Whether the filter will converge for a planar object is not obvious. On the one hand, the previous section indicates that three images are enough for uniqueness. On the other hand, the constraints of all the images are not applied simultaneously; at each time step k, the problem will be under-determined and a manifold of false structures will be possible. To resolve this a Monte Carlo experiment is conducted. Random points are selected from a plane, which is randomly oriented. The object undergoes random motion and the point measurements are forwarded to the Kalman filter. The filter runs for 1000 frames and if the summed squared error of the estimated depth values is smaller than 0.1, convergence is declared. The experiment is repeated 1000 times using two types of motion: *rotation* around a random vector in the $z = 0$ plane (to avoid the degenerate rotation around the z-axis) and *Brownian motion*, that is, small incremental random steps in translation and rotation. The filter is initialized with $\beta = 2.0$ (the true value is $\beta = 1.3$). Three types of initialization procedures are tried for the depths $\alpha_{1,\ldots,N-1}$: in the first type, called *prior 1*, the α-values are set to random values in the interval ± 0.5 around the depth α_0 which is fixed to 1.0. In the second type, called *prior 2*, the filter is initialized with the true α-values plus random noise in the interval ± 0.5. The last one, *prior 3* is equivalent to prior 2 but with noise in the interval ± 0.25. The entire experiment

Table 1: Depth convergence frequency for noise free measurements.

	prior 1	prior 2	prior 3
2D rotational	0.673	0.953	1.000
3D rotational	0.796	1.000	1.000
2D brownian	0.751	0.972	1.000
3D brownian	0.762	1.000	1.000

Table 2: Depth convergence frequency for noisy measurements.

	prior 1	prior 2	prior 3
2D rotational	0.656	0.898	0.987
3D rotational	0.802	0.995	1.000
2D brownian	0.508	0.851	0.924
3D brownian	0.478	0.943	0.981

is then repeated for a random three-dimensional object. This time, prior 1 will mean that all the α-values are initialized to the same value 1.0, whereas the prior 2 and prior 3 will mean that the α-values are initialized to the correct value plus random noise in the intervals ± 0.5 and ± 0.25, respectively. The result is shown in Table 1 for the case of noise-free measurements.

As can be seen, there are differences between the planar (2D) and the general (3D) objects, but as the quality of the depth prior improves, the convergence frequency goes to unity for both types of objects. In the case of noisy measurements, the difference is larger. In Table 2, uniformly distributed noise of ± 1 pixel is added to the measurements. Still, the difference is quantitative, not qualitative. One example is when tracking a planar object undergoing Brownian motion, with depth prior 3 (third row, third column). In this case, one has as much to gain by changing the motion to rotational as changing the planar object to a general three-dimensional one (convergence frequency rises to 0.987 compared to 0.981). Hence, a planar object is not a catastrophic situation that means that the SfM algorithm will unconditionally fail. Rather, planar objects can be seen as something that should be avoided if possible, just like measurement noise, poor depth priors, or low excitation in the motion. A way to understand why the filter converges for planar views is the following. The SfM problem at image k can be formulated as finding the state vector $\mathbf{x}$ which minimizes the (scalar) error function

$$J_k(\mathbf{x}) = \left(\mathbf{z}_k - \mathbf{h}_k(\mathbf{x})\right)^T \left(\mathbf{z}_k - \mathbf{h}_k(\mathbf{x})\right), \tag{3}$$

where $\mathbf{h}_k(\mathbf{x})$ consists of the Kalman filter's estimate of the projected points $(\hat{u}_1, \hat{v}_1, \hat{u}_2, \hat{v}_2, \ldots, \hat{u}_N, \hat{v}_N)$. Each new image k gives a new constraint $J_k(\mathbf{x}) = 0$. Since the object is planar and since the focal length is constant (not varying over time) but its value unknown. $m = \lceil (1+4)/2 \rceil = 3$ images are needed for uniqueness, but only two images are part of $J_k(\mathbf{x})$. Therefore, an entire curve $x_k(t)$ of false solutions will satisfy $J_k(\mathbf{x})$. If this curve is projected down

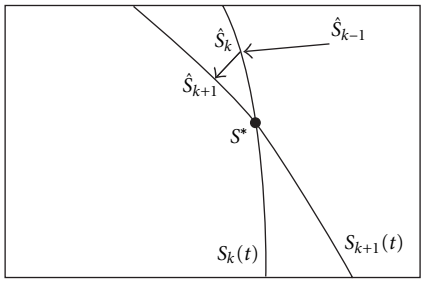


Figure 3: Structure convergence for planar objects.

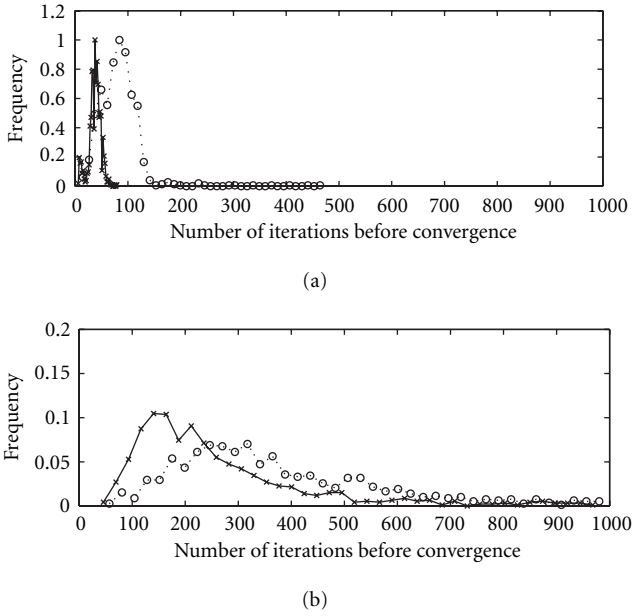


Figure 4: (a) Histogram over convergence time for rotational motion with a general object (solid) and a planar object (dotted). (b) Ditto for Brownian motion.

to only the structure components of x, a new curve $s_k(t)$ is obtained, where $\mathbf{s} = (\beta, \alpha_1, \alpha_2, \ldots, \alpha_N)$. Since the solution is unique for three views, $s_k(t)$ and $s_{k+1}(t)$ cannot be the same curve, and must meet in the point $\mathbf{s}^*$ that represents the correct structure. This is illustrated in Figure 3. The estimate $\hat{s}$ will move towards the curves, but for each new image the curve has moved and $\hat{s}$ will continue moving until it has found the optimum. This should result in a slower convergence than in the over-determined case, and this has also been verified experimentally: the top diagram in Figure 4 shows a histogram over how many frames are needed to have a sum squared depth error of 0.05 for rotational motion. In the bottom diagram, a similar histogram is drawn for Brownian motion. It is clear that the general three-dimensional objects (solid curve with crosses) converges faster than the planar ones (dotted curve with circles), but in this case the type of motion seems to have a bigger impact on the convergence time than has the shape of the object.

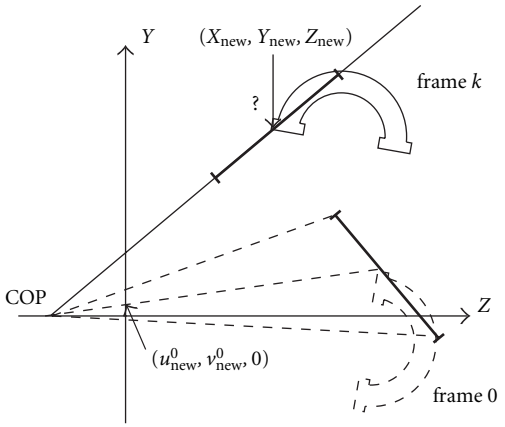


FIGURE 5: Expected bias in (u^0_{new}, v^0_{new}) when the depth of the new point $(X_{new}, Y_{new}, Z_{new})$ is unknown.

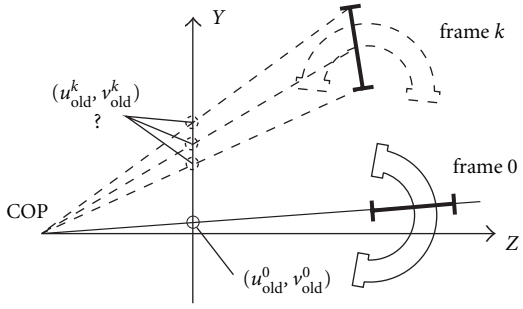


FIGURE 6: The bias in the position (u^k_{old}, v^k_{old}) in the new reference frame when the depth of an old point is not known exactly (lower black interval).

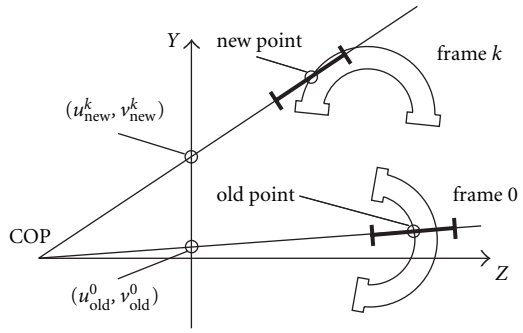


FIGURE 7: The proposed method—the old points will keep the old reference frame, whereas the new points will use the new reference frame. No bias due to depth will occur.

2.3. Appearing points

In a tracking scenario, it is advantageous to be able to track points that were not visible in the first image. In the example of head tracking, for example, it is desirable to be able to track points on the ear when the head turns sideways. It is thus essential that the algorithm is able to include some of the new points that have appeared. The last part of this section will treat how points can be added to the Kalman filter based SfM algorithm. The work described here was presented in [21]. Independently, Dell'Acqua et al. [22] later published similar work. However, their solution differs from the one presented here and will be treated at the end of this section.

As described in the beginning of Section 2, the function $\mathbf{h}_k(\cdot)$ uses the image coordinates in the first frame, (u^0, v^0), to calculate (X, Y, Z), (X_C, Y_C, Z_C), and then the estimate $(\hat{u}, \hat{v})$. The fact that (u^0, v^0) must be known in order to calculate the filter estimates poses a problem when adding feature points at a later stage. If a new feature point becomes visible first at frame k, its image plane projection in the original reference frame (u^0_{new}, v^0_{new}) is not known.

2.3.1 Old reference frame

One way to solve the problem is to rotate and translate back the measurements from the kth frame to the 0th frame. As can be seen in Figure 5, however, the lack of knowledge of the depth α will result in a large bias in the estimation of (u^0_{new}, v^0_{new}).

2.3.2 New reference frame

Another solution would be to change the reference frame and restart the Kalman filter at the kth frame. As can be seen in Figure 6 a similar problem to that of Figure 5 occurs: for each old point, the image coordinates from the first frame (u^0_{old}, v^0_{old}) (solid circle) are replaced with estimates (u^k_{old}, v^k_{old}) in the kth frame (middle dashed circle). The depth α is not known exactly, and this creates a bias in the position of (u^k_{old}, v^k_{old}) (other dashed circles). Since the filter has had some time to converge, there is an estimate of α, and the bias of (u^k_{old}, v^k_{old}) is thus smaller than with the *old reference frame* method of Figure 5. On the other hand, bias is added to *all* the old points, compared to only the new points, which are assumed to be fewer.

2.3.3 Bias estimation

Both the *old reference frame* and the *new reference frame* methods will thus suffer from bias. One solution to this is to include this bias in the state vector and estimate it using the Kalman filter. Bias estimation was proposed in the original paper by Azarbayejani and Pentland [1], but not specifically as a solution to the problem of adding points. Equation (1) is then modified to

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} (u^0 + b_u) \\ (v^0 + b_v) \\ 0 \end{pmatrix} + \alpha \begin{pmatrix} \beta(u^0 + b_u) \\ \beta(v^0 + b_v) \\ 1 \end{pmatrix}, \tag{4}$$

and b_u and b_v are added to the state vector $\mathbf{x}$. However, introducing these extra degrees of freedom will make the filter less over-determined.

2.3.4 Two reference frames

The method proposed in this paper is to maintain two reference frames, one for the old points and one for the new ones.

As illustrated in Figure 7, the old points will continue to be restricted to the line from the center of projection (COP) to $(u^0_{\text{old}}, v^0_{\text{old}})$, whereas the new points will lie on the line from the COP to $(u^k_{\text{new}}, v^k_{\text{new}})$. The prediction $(\hat{u}, \hat{v})$ for the old points will continue to be calculated using (the projection of) (5), whereas the new points will be calculated according to the projection of

$$\begin{bmatrix} \hat{X} \\ \hat{Y} \\ \beta\hat{Z} \end{bmatrix} = BR(q)R_k^T \left(\begin{pmatrix} (1+\alpha\beta)u^k \\ (1+\alpha\beta)v^k \\ \alpha \end{pmatrix} - T_k \right) + \begin{bmatrix} t_x \\ t_y \\ \beta t_z \end{bmatrix}, \quad (5)$$

where $B = \text{diag}[(1, 1, \beta)]$ and R_k, T_k represent the rotation and translation between frame 0 and frame k.

This approach is still not bias free—estimation errors in the rotation quaternion and in the translation vector will give rise to a shift in the texture map and hence a bias in $(u^k_{\text{new}}, v^k_{\text{new}})$. However, this problem also occurs with the other methods. Moreover, in the proposed scheme the rotation and translation bias only applies to the new points, as opposed to all the old points in the *new reference frame* method. As mentioned above, these biases in u and v can be estimated. Alternatively, the motion parameters R_k^T and T_k can be estimated. This amounts to only 6 extra degrees of freedom for the filter, which is advantageous to estimating b_u and b_v if more than 3 points are added at once. The implementation of the real system (described in Section 3), did not seem to suffer from these biases, and the *two reference frames* method could be used without extending the feature vector for bias estimation.

2.3.5 Related work

Recently, the problem of adding points to the EKF based SfM algorithm has been investigated by Dell'Acqua et al. [22]. Their solution is to start an independent Kalman filter each time a new point occurs. After collecting data from the entire sequence, a single Kalman filter is stitched together from all the others. Each time a point disappears from the *master* Kalman filter, all the *slave* filters that have been created up to that point are examined for replacement candidates. The point that will survive the longest is then used to replace the old point. The *old reference frame* method is used, and the bias of the new point can be reduced since the depth α can be obtained from the *slave* Kalman filter, that has been converging for a while. The *master* filter is then continued until a new point disappears, and the procedure is repeated. No attempt is made to reacquire old points—when they reappear they are treated as new, unknown points.

Since the above-mentioned method is of batch type it is not well suited for real-time tracking. It can obviously be modified so that no look-ahead is used, but even so the use of multiple Kalman filters (one filter for each new point) makes it a bit computationally expensive in a real-time scenario. Furthermore, just as in the *new reference frame* method, any remaining error in the estimated depth α will result in some bias. On the other hand, there are advantages to having all points in the same reference system. For instance, the extra matrices for rotation and translation (R_k and T_k) of (5) are avoided.

Figure 8: A generic three-dimensional polygonal head model is aligned with a head-on shot of the video sequence, and the corresponding pixels are texture mapped to the surface of the face model.

3. HEAD TRACKING

In this section, the extended SfM algorithm from Section 2.3.4 will be used in a head tracking application. Recalling Figure 1, the head is rendered using a generic polygonal face model[1] [7] in the predicted pose (lower left). Patches from this rendered image are matched against the incoming video (top left). The two-dimensional measurements are fed into the SfM Kalman filter that calculates the three-dimensional structure, pose and focal length for a point configuration defined by the center of each patch. The pose and the focal length are then used to render the polygonal model, and the structure is used to predict the positions of the patches' centers in the next frame. Whereas the solid arrows in Figure 1 represent information flows that occurs every frame, the dashed arrows are invoked only at texture update. When the head has turned sufficiently, texture is grabbed, and both the three-dimensional model (bottom left) and the extended Kalman filter (upper right) are updated.

3.1. Initialization

The system is initialized from a frontal position as seen in Figure 8. The face model (left) is aligned to match with a head-on view of the face in the video sequence (middle). The pixels from the video are then texture-mapped onto the model (right). Our system uses a manual alignment—the user has to put the head in a prespecified position on the screen, and make sure that she/he is in a frontal position before initiating tracking.

After alignment has been performed, the system selects which feature points to use. The part of the video input containing the face is cropped out (first image of Figure 9) and further processed. The cropped image is lowpass filtered and subsampled once to avoid locking on to features that are too vague to be reliably tracked. The determinant of the Hessian $\begin{vmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{vmatrix}$ is calculated at every point. To avoid selecting points on parts of the face surface perpendicular to the camera, the determinant is weighted with the cosine of the angle between the surface normal and the camera direction. These values

[1] The three-dimensional model is a modified version of CANDIDE.

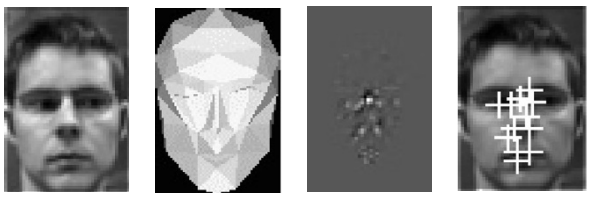


Figure 9: From left: The lowpass filtered incoming video, the weighting (the cosine of the angle between the surface normal and the camera direction), the final rating, and the extracted feature points.

can easily be obtained from the computer graphics hardware by rendering a gray shade version of the three-dimensional model with lighting from the camera direction (Figure 9, second image). The resulting rating of each pixel is shown in the third image in Figure 9, where brighter pixels indicate a higher score. The 24 points with the highest ratings are then selected with a minimum-distance constraint between points. The fourth image in Figure 9 shows 12 of the 24 points selected. Each point is given a three-dimensional position on the surface of the three-dimensional model. Again, the computer graphics hardware can be used, this time reading out the value of the depth buffer of the rendered image in the corresponding pixel location, to calculate the depth of the feature point.

3.2. Feature point tracking

As shown in Figure 1, the tracking is carried out between the rendered frame and the video input. Both images are subsampled in order to track larger and more robust features. Since the feature points are fixed with respect to the three-dimensional model, their two-dimensional coordinates in the rendered image are known. A 7×7 pixel patch around each feature point is cropped out. This patch is then matched with patches from the video input image in a 17×17 pixel search window using weighted normalized correlation. More specifically, if $\mathbf{a}$ and $\mathbf{b}$ are the vectors obtained by raster scanning the patch in the rendered image and the video input respectively, then the $\hat{\mathbf{b}}$ that maximizes

$$\varrho = G(x, y)\cos(\theta) = G(x, y)\frac{\hat{\mathbf{a}} \cdot \hat{\mathbf{b}}}{||\hat{\mathbf{a}}||||\hat{\mathbf{b}}||} \tag{6}$$

is selected, where $\hat{\mathbf{a}} = \mathbf{a} - \mu_{\mathbf{a}}$, $\hat{\mathbf{b}} = \mathbf{b} - \mu_{\mathbf{b}}$, $\|\cdot\|$ represents the norm, and $G(x, y)$ is a Gaussian weighting function that punishes large jumps of the patches. The search window is centered around the position that is estimated from the structure from motion algorithm. An exhaustive search is carried out in the search window and the candidate with the lowest error is selected.

3.2.1 Subpixel refinement

Since the two images are subsampled before matching, the accuracy of the tracking is only ± 1 pixel. However, since an exhaustive search is performed over the search window, the error is known in the adjacent positions. By approximating error derivatives with central differences, the error surface is approximated by a second degree Taylor polynomial

$$\varrho(\Delta\mathbf{x}) \approx \varrho_0 + \mathbf{c}^T\Delta\mathbf{x} + \Delta\mathbf{x}^T H\Delta\mathbf{x}. \tag{7}$$

The (subpixel) location of the minimum of the resulting paraboloid is then used as the feature location. If the error surface is very irregular, however, the minimum of the paraboloid can be outside the 2×2 pixel area. In this case, the Taylor polynomial is a bad approximation of the error surface and the centroid of the 2×2 pixel area is used instead.

3.3. Estimating measurement noise covariance

From the model and the image data it is possible to gather information about the quality of each point measurement. For instance, if the correlation coefficient ϱ is very small, there is a reason to believe that the point is not in the correct position, and we would like the Kalman filter to discard that measurement. The same should happen if the point on the model is facing away from the camera or is occluded by other parts of the model. Finally, instead of estimating the variances of the error in the x- and y-direction individually, the covariance matrix Σ^k of the error of each measurement (x_k, y_k) can be estimated from the Taylor expansion of ϱ.

In the case of template matching, there are basically two types of matching errors. Either the correct match is found, offset only slightly because of small changes in lighting, small errors in the projective transformation of the patch, and so forth. This is called the *small error case* in this paper. The other possibility is that the match is completely wrong, for instance due to occlusion or to that the normal of the feature point is almost perpendicular to the camera direction. This is named the *large error case*. Depending on the residual error, the angle to the camera and the visibility of the triangle, Σ^k will be selected to be Σ_s in the small error case, and Σ_l otherwise.

3.3.1 Small error case Σ_s

Inspired by Jebara and Pentland [2], the covariance could be made proportional to the Hessian of the Taylor expansion of the error (equation (7)),

$$\Sigma_s \sim (-H)^{-1}. \tag{8}$$

This means that all points that contribute to a certain correlation value ϱ will get the same probability. In other words, the iso-residual ellipses will correspond to iso-probability ellipses. We will assume that all measurements of the small error type will be of equal quality and should be given similar uncertainties. Thus the Hessian H should only contribute to the shape of the error distribution. If $(-H)^{-1}$ is symmetric and positive definite (which it should if a proper maximum has been found), it can be diagonalised into $(-H)^{-1} = RDR^T$ [25], where R is a rotation matrix and D is a diagonal matrix with positive elements. The radii (a, b) of the ellipse $\mathbf{x}^T(-H)\mathbf{x} = 1$ will then correspond to the square roots of the elements in D; $a = \sqrt{d_{11}}$, $b = \sqrt{d_{22}}$. If the rotation angle

FIGURE 10: Resulting estimation of Σ_s. The radii of the ellipses $\mathbf{x}^T\Sigma_s\mathbf{x} = 1$ are shown, where long radii correspond to large uncertainties.

and the thickness of the ellipse $\mathbf{x}^T\Sigma_s^{-1}\mathbf{x} = 1$ are set to be the same as for $\mathbf{x}^T(-H)\mathbf{x} = 1$, this means that the rotation matrix R and the ratio of the radii a/b should remain the same. Hence, Σ_s can be written as

$$\Sigma_s = R\begin{pmatrix} ca & 0 \\ 0 & cb \end{pmatrix}^2 R^T, \tag{9}$$

where the constant c can be set to give Σ_s the desired uncertainty, or *entropy*. Shannon [26] shows that the entropy of a Gaussian random variable of covariance Σ equals $\ln((2\pi e)^{d/2}|\Sigma|^{1/2})$. The constant c is chosen so that Σ_s has the same entropy as $\Sigma_{\text{ref}} = \left(\begin{smallmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{smallmatrix}\right)$, where $\sigma = 4$ pixels. A check is also made on ca and cb so that they are at least 1 pixel.

An example of estimation of Σ_s can be seen in Figure 10, where the radii ca and cb have been plotted. Just as expected, the uncertainty is bigger along elongated features such as the mouth and the side of the nose.

3.3.2 Large error case

In the large error case, the assumption is that the true position is somewhere in the search area region. A standard deviation of $10\sigma = 40$ pixels is used, which is in the same order of magnitude as the 22×22 pixel search window (11×11 in the subsampled image). Since the best match is assumed to be in the wrong position, the local Taylor expansion is not valid and no attempt is made to shape the noise. Hence, the covariance matrix $\Sigma_l = \left(\begin{smallmatrix} 100\sigma^2 & 0 \\ 0 & 100\sigma^2 \end{smallmatrix}\right)$ is used for the large error case.

3.3.3 Choosing covariance matrix

The choice between Σ_s and Σ_l is mainly decided by the correlation value ϱ. The small error model should be selected if $p(\text{small} \mid \varrho) > p(\text{large} \mid \varrho)$, which is equivalent to

$$f(\varrho \mid \text{small})\,p(\text{small}) > f(\varrho \mid \text{large})\,p(\text{large}) \tag{10}$$

using Bayes' rule. $p(\text{small})$ and $p(\text{large})$ vary a lot during tracking. For instance when the head is frontal, $p(\text{small})$ is almost one, whereas when the head is rotated far from the initialization pose, $p(\text{large})$ might be the bigger one. Assuming they are equally probable, the decision rule is now simplified to

$$f(\varrho \mid \text{small}) > f(\varrho \mid \text{large}). \tag{11}$$

To estimate $f(\varrho \mid \text{small})$ and $f(\varrho \mid \text{large})$, the following experiment was conducted. The tracker was allowed to run, and the user's head was rotated to a critical pose where the tracked point deviated from its correct position. The correlation value was measured for a number of frames, each measurement being an example of $f(\varrho \mid \text{large})$. Next, the head was moved until the feature point just moved back to the correct position. The correlation value was then measured continuously during a rotation from this critical pose to a frontal pose, in order to get correlation values from all types of poses. These correlation values were used as examples of $f(\varrho \mid \text{small})$. The procedure was repeated for all points, and $f(\varrho \mid \text{large})$ and $f(\varrho \mid \text{small})$ were estimated using normalized histograms. The same number of measurements was used from each point. The result is shown in Figure 11.

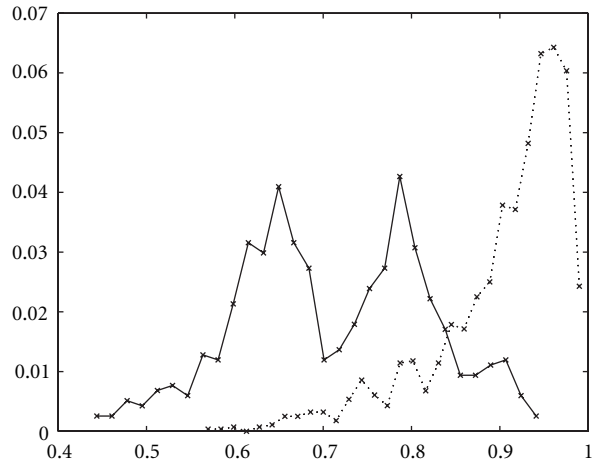


FIGURE 11: Estimated $f(\varrho \mid \text{small})$ (dotted) and $f(\varrho \mid \text{large})$ (solid).

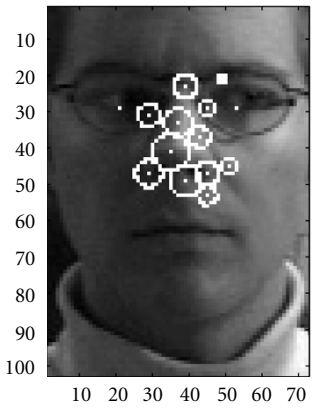


FIGURE 12: Converged depths estimates. Small circles indicate large depth.

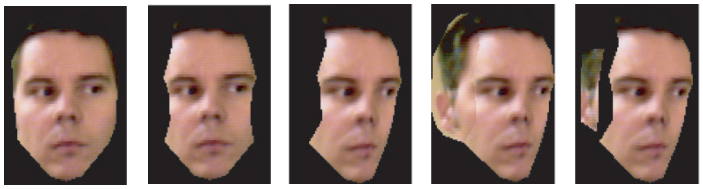
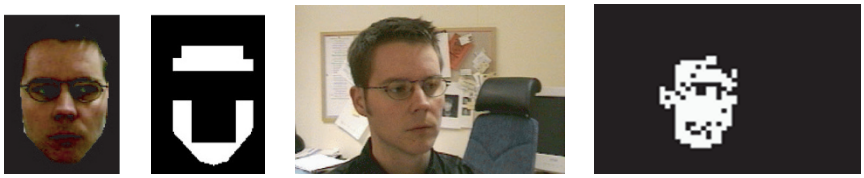
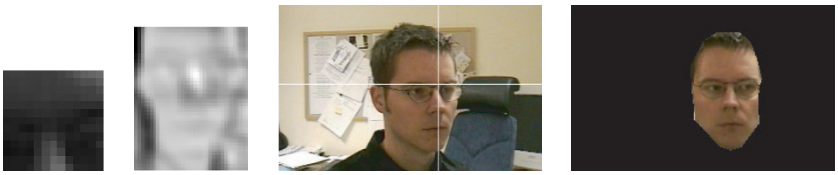


FIGURE 13: From left: head-on shot used for initialization, first tracked frame, frame just before texture update, frame just after texture update. The small area in last image shows where the system looks for new feature points.



(a) (b) (c) (d)

FIGURE 14: (a) Original texture, (b) mask, (c) input image, (d) color blob.



(a) (b) (c) (d)

FIGURE 15: (a) Between-eyes template, (b) correlation surface, (c) result, (d) model position.

The diagram shows that, using the rule in (11), the decision boundary should be put at around $\varrho = 0.8$. This value has also proven to work well in practice. For the angle θ between the triangle normal and the viewing direction, a decision boundary of $\cos(\theta) = 0.2$ was found. Thus Σ_s is selected if $\varrho > 0.8$, $\cos(\theta) > 0.2$, and the triangle is visible. Otherwise, Σ_l is selected.

3.4. Depths convergence

The tracked trajectories of the feature points are forwarded to the Kalman filter, which uses this information to infer the depths of the points. In Figure 12 the converged depths estimates are shown. Note the small circles near the center of the eyes. This converged structure is then used by the Kalman filter to constrain the motion of the individual feature points and thus improve the tracking of the head movements.

3.5. Texture update

Since the initialization procedure is done using a head-on shot of the head, all the feature points will be situated in the frontal face area. Thus, at large out-of-plane rotations, few or no feature points will be visible in the image and the system will inevitably lose track. Tracking points at the side of the head would solve this problem. However, since the texture is acquired from the zeroth frame during the initialization procedure, parts of the head that are not visible in a head-on shot will not get an accurate texture. Therefore, the system automatically extracts new texture when the head has rotated enough. More specifically, the texture for the entire side of the head will be acquired from the video when the scalar product between the camera direction and the triangle normal is greater than a certain constant value. Figure 13 illustrates this; the first image is the head-on shot used for initialization. The second image is the texture map acquired at the first frame. The third and the fourth image is the model just before and after the acquisition of the new textures on the side. The new feature points on the acquired texture are obtained the same way as described in Section 3, with one difference. Since points that are too close to the head boundary, or to the discontinuous "seam" between the two textures are undesirable, only a smaller area of the texture is searched for feature points. The small area in the last image in Figure 13 shows where the system looks for new feature points.

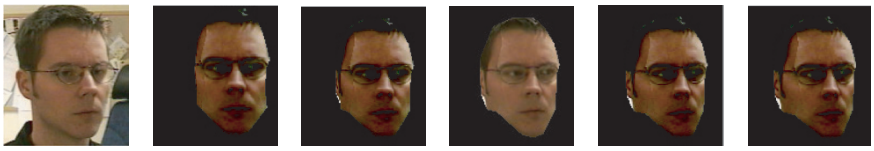


Figure 16: Tracker convergence at restart.

4. REINITIALIZATION

Sooner or later, the tracking will fail. Typical scenarios are that the head moves too quickly, that it is rotated out of the tracking range, or that the entire head is occluded, for example, by an arm. In situations like these, it is desirable to have the system detect the failure and start a reinitialization procedure. It should be noted that reinitialization is a simpler task than original initialization. This is due to the fact that a lot of information from the initialization can be used to simplify the search for the head, for example, the texture of the face and the converged Kalman filter. This means that simple methods such as skin color blobs and template matching can be reliably used, since they can be tuned for the specific appearance of the actual face.

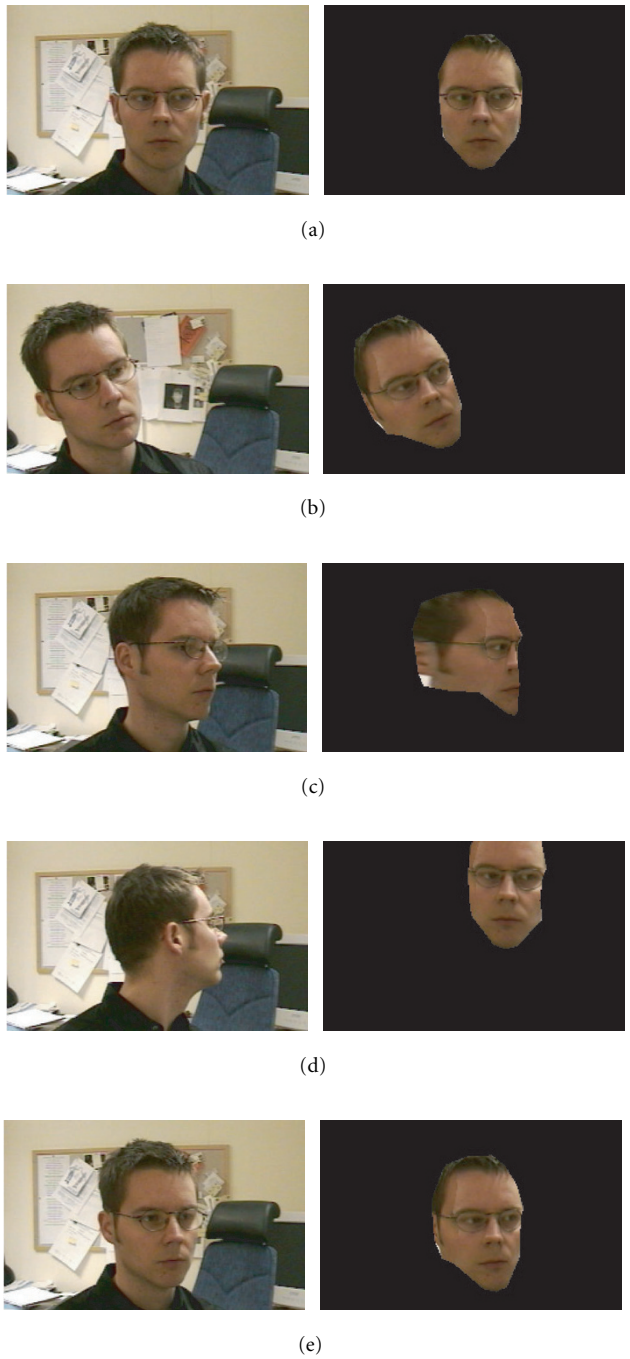
Prior attempts at reinitialization are not that plentiful in the literature. Jebara and Pentland [2] detect tracking failure by normalizing the face texture back to frontal position and measuring the distance from face space (DFFS). When the DFFS is larger than a threshold, a face detection process restarts the tracker from scratch, disregarding information gathered so far. Crowley and Berard [27] use three visual processes for two-dimensional tracking and continuous reinitialization; a skin-color blob model, a correlation tracker, and a blink detector. Perhaps the most similar approach to reinitialization is proposed by Matsumoto and Zelinsky [28]. They use the correlation values from the tracking to detect failure, and a coarse-to-fine template matching step of the entire face in order to find a good restarting position. Their face tracking system is using stereo camera input, in contrast to monocular camera input as proposed here.

4.1. Failure detection

The tracking procedure described in Section 3.2 produces a correlation value ϱ for each point. A value of ϱ close to 1 indicates an accurate match, whereas a low ϱ means that the measurement is insecure. If the tracking has failed, it is unlikely that any point will produce a high ϱ. We declare a tracking error if the best ϱ is smaller than a threshold value for the entire duration of ten frames. The resulting detector is not fail proof, but since the ϱ measurements are a by-product of the tracking, it is virtually cost-free. It also works reasonably well in practice.

4.2. Finding the face

Once a tracking failure is declared, the reinitialization process starts with building a skin color model. This is a widely used method for finding faces and hands, introduced by



(a)

(b)

(c)

(d)

(e)

Figure 17: Typical tracking sequence. The tracking is initialized in (a), and continued in (b). The image (c) shows the maximum rotation that can be managed without losing track, whereas the track is lost in (d). In (e) the tracking is regained.

(a) (b) (c) (d)

FIGURE 18: Occlusion robustness due to estimation of the measurement noise covariance matrix. Left: fixed covariance, resulting in mismatched points (a) and lost track (b). Right: covariance estimated with the proposed method. Only points that are estimated to be small error cases are shown in (c). The tracking is almost unaffected (d).

Crowley and Berard [27] and by Olivier et al. [29]. By gathering statistics from the original texture (Figure 14a) in the areas likely to contain skin color (Figure 14b), the probability $p(\text{skin} \mid \mathbf{x})$ that a pixel with color $\mathbf{x}$ depicts skin can be modeled as a Gaussian in $YCrCb$ space. The skin color probability can then be calculated and thresholded for every pixel in a subsampled version of the image (Figure 14c). A connected components processing step is performed on the binary image, and the biggest connected skin color blob is assumed to be the face (Figure 14d).

The skin color blob provides a very robust but not entirely accurate position of the head. To refine this position, a 13×13 template (Figure 15a) from the area between the eyes is cropped out from a subsampled version of the original texture. The bounding box of the color blob is used as a search window for the template. Exhaustive search using normalized correlation is used in the search window, and the location of the maximum of the correlation (shown in Figure 15b) is used. Figure 15c shows a result of the template matching. Since the texture between the eyes is situated in the middle of the face, it is visible even for relatively large out-of-plane rotations. Empirically, it also works for comparatively large changes in scale. The model is then placed in the estimated position, facing the camera head-on and at the same z distance as in the original initialization (Figure 15d). The tracker is then restarted and the Kalman filter will resolve the rotation and scale of the head for a rather large set of head poses. This is shown in Figure 16. The left-most image shows the video input at the time of reinitialization. The tracker is restarted in the position shown in the second image. After five iterations (about 0.2 second) the tracker has converged to the correct pose.

5. SYSTEM EVALUATION

The tracking system performs in real time on a SGI O2 R12000 270 MHz workstation. The feature point finding algorithm (executed once at the start of the tracking process) takes about 100 milliseconds, and the rest of the tracking runs at 25 Hz. A typical tracking sequence can be seen in Figure 17. The system is initialized in Figure 17a, the head is tracked in Figures 17b and 17c, the track is lost in Figure 17d and regained in Figure 17e.

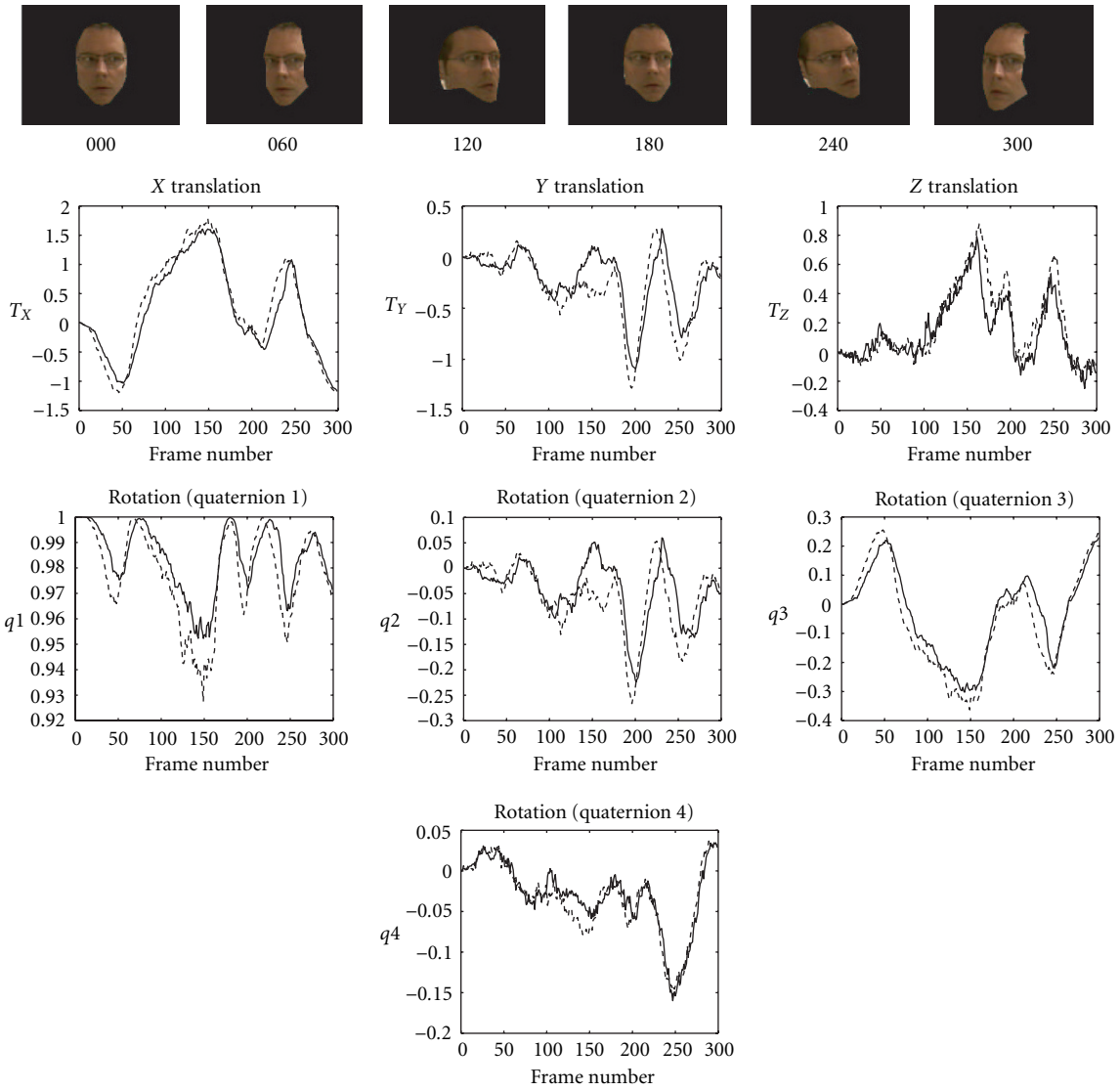


FIGURE 19: Example of tracking of a synthesized sequence where the motion is known. In each graph, the dashed curve represents the ground truth motion, and the solid curve represents the motion estimated from the tracker.

5.1. Robustness to occlusion

Due to the estimation of the measurement noise covariance matrix, the system is robust to occlusion of a large number of the feature points by, for example, a hand, as shown in the sequence (Figure 18). Figures 18a and 18b show what happens when the covariance matrix is fixed; the estimated pose jerks severely and the track is often lost. With the proposed estimation of the covariance matrix, the tracking is almost completely unaffected by the occlusion (Figures 18c and 18d). The sequence shown in Figure 18 is one of many such examples.

5.2. Evaluation on synthetic data

To get an idea of the accuracy of the tracking, the following experiment has been conducted: first the tracker is run on a live image sequence to provide ground truth motion parameters (the first pass). Then a synthetic sequence is rendered using these motion parameters. The tracker is now run a second time on the synthetic sequence (second pass), and the estimated motion parameters are compared to the ground truth data. It should be noted that such an experiment does not measure how well the tracker can follow the motions in the original, live image sequence. The reason for this is that it is not possible to rule out that the original sequence is moving in a way that the tracker in the first pass does not follow, and that this complex motion is filtered out from the ground truth data. Tracking the synthetic sequence would then be an easier task than tracking the original live sequence. Still, the experiment gives an idea of how well the tracker can behave on a naturally moving head. The top row of Figure 19 shows a few frames from the synthetic sequence. The middle row and the bottom row show the translation and rotation parameters, respectively. In each chart the

ground truth is marked with a dashed curve and the tracked data is marked with a solid curve. The most striking systematic error is that there is a delay between the ground truth and the estimates from zero to about ten frames (0.4 second). Also, at around frame 150, the error between the estimated parameters and the ground truth widens temporarily, as best seen in the Y translation and in quaternion 1, 2, and 4. Around frame 160, however, the estimates have recovered.

6. CONCLUSIONS

The SfM algorithm by Azarbayejani and Pentland has been analyzed for planar surfaces. Although the performance of the algorithm decreases for such surfaces, the behavior is not catastrophic and can be compared with measurement noise, poor depth priors, or low motion excitation. The algorithm is extended to handle new points, and it is then used in a head tracking system. Due to estimation of the covariance of the measurement noise, the system is robust to simultaneous occlusion of a large number of feature points. Furthermore, the system automatically detects tracking failure and performs a reinitialization using a color model and a template learned from the model. An experiment on synthetic data has also been conducted, showing a good following of the estimated parameters.

REFERENCES

[1] A. Azarbayejani and A. Pentland, "Recursive estimation of motion, structure, and focal length," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 17, no. 6, pp. 562–575, 1995.

[2] T. Jebara and A. Pentland, "Parametrized structure from motion for 3D adaptive feedback tracking of faces," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 144–150, San Juan, Puerto Rico, June 1997.

[3] M. J. Black and Y. Yacoob, "Recognizing facial expressions in image sequences using local parameterized models of image motion," *International Journal of Computer Vision*, vol. 25, no. 1, pp. 23–48, 1997.

[4] S. Basu, Essa I., and A. Pentland, "Motion regularization for model-based head tracking," in *Proc. 13th Int'l. Conf. on Pattern Recognition*, vol. C, pp. 611–616, Vienna, Austria, August 1996.

[5] Y. Zhang and C. Kambhamettu, "Robust 3D head tracking under partial occlusion," in *4th International Conference on Face and Gesture Recognition*, pp. 176–182, Grenoble, France, March 2000.

[6] P. Roivainen, *Motion estimation in model-based coding of human faces*, Licentiate thesis, Linköping University, Sweden, 1990.

[7] M. Rydfalk, "CANDIDE, a parameterized face," Tech. Rep. LiTH-ISY-I-0866, Linköping University, Sweden, October 1987.

[8] H. Li, P. Roivainen, and R. Forchheimer, "3-D motion estimation in model-based facial image coding," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 15, no. 6, pp. 545–555, 1993.

[9] H. Li and R. Forchheimer, "Two-view facial movement estimation," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 4, no. 3, pp. 276–287, 1994.

[10] D. DeCarlo and D. Metaxas, "The integration of optical flow and deformable models with applications to human face shape and motion estimation," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 231–238, San Francisco, Calif, USA, June 1996.

[11] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," in *Proc. European Conference on Computer Vision 1998*, H. Burkhardt and B. Neumann, Eds., vol. 2, pp. 484–498, Springer-Verlag, 1998.

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

[13] J. Ahlberg, "Facial feature tracking using an active appearance-driven model," in *Proc. the EuroImage ICAV3D 2001 Conference*, pp. 116–199, Mykonos, Greece, May 2001.

[14] J. Ström, T. Jebara, S. Basu, and A. Pentland, "Real time tracking and modeling of faces: an EKF-based analysis by synthesis approach," in *Proc. the Modelling People Workshop at ICCV '99*, Corfu, Greece, August 1999.

[15] R. I. Hartley, "Ambiguous configurations for 3-view projective reconstruction," in *Proc. 6th European Conference on Computer Vision*, pp. 922–935, Dublin, Ireland, June–July 2000.

[16] G. P. Stein and A. Shashua, "On degeneracy of linear reconstruction from three views: linear line complex and applications," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 21, no. 3, pp. 244–251, 1999.

[17] P. Torr, W. Fitzgibbon, and A. Zisserman, "Maintaining multiple motion model hypotheses over many views to recover matching and structure," in *Proc. 6th International Conf. on Computer Vision*, pp. 485–491, Bombay, India, January 1998.

[18] B. Triggs, "Autocalibration from planar scenes," in *Proc. the 5th European Conference on Computer Vision*, pp. 89–105, Freiburg, Germany, June 1998.

[19] J. Ström, "Structure from motion of planar surfaces, (analysis of an EKF based approach)," in *Proc. Symposium on Image analysis, Swedish Society for Automated Image Analysis*, pp. 13–16, Norrköping, Sweden, March 2001.

[20] J. Ström, "Reinitialization of a model-based face tracker," in *Proc. the EuroImage ICAV3D 2001 Conference*, Mykonos, Greece, May 2001.

[21] J. Ström, T. Jebara, and A. Pentland, "Model-based real-time face tracking with adaptive texture update," Tech. Rep. LiTH-ISY-R-2342, Linköping University, Sweden, March 2001.

[22] A. Dell'Acqua, A. Sarti, and S. Tubaro, "Effective analysis of image sequences for 3D camera motion estimation," in *Proc. the EuroImage ICAV3D 2001 Conference*, pp. 307–310, Mykonos, Greece, May 2001.

[23] T. Jebara, A. Azarbayejani, and A. Pentland, "3D structure from 2D motion," *IEEE Signal Processing Magazine*, vol. 16, no. 3, pp. 66–84, 1999.

[24] A. Gelb, *Applied Optimal Estimation*, MIT Press, Cambridge, 1996.

[25] S. Spanne, *Lectures in Matrix Theory*, Department of Mathematics, Lund Institute of Technology, Lund, Sweden, 1993.

[26] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423, July, October 1948.

[27] J. L. Crowley and F. Berard, "Multi-modal tracking of faces for video communications," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 640–645, San Juan, Puerto Rico, June 1997.

[28] Y. Matsumoto and A. Zelinsky, "Real-time face tracking system for human-robot interaction," in *Proc. 1999 IEEE International Conference on Systems, Man, and Cybernetics*, pp. II–830–II–835, Tokyo, Japan, October 1999.

[29] N. Oliver, S. Pentland, and F. Berard, "LAFTER: a real-time lips and face tracker with facial expression recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 123–129, San Juan, Puerto Rico, June 1997.

Jacob Ström received his M.S. degree in computer science and engineering at Lund Institute of Technology, Sweden, in 1995. He was a visiting Ph.D. student at University of California, San Diego in 1995–1996, working for Prof. Pamela Cosman. In 1998, Ström received a Fulbright grant and spent one year as a visiting Ph.D. student at Prof. Alex Pentland's Vision and Modeling group at the MIT Media Lab. In February 2002, he received the Ph.D. degree in image coding Linköping University, Sweden. He is currently at the Multimedia Technologies Department at Ericsson Research, Sweden.