Robust 3D Head Tracking using Camera Pose Estimation

Shay Ohayon  
Computer Science Department  
Israel Institute of Technology  
Haifa, Israel 32000

Ehud Rivlin  
Computer Science Department  
Israel Institute of Technology  
Haifa, Israel 32000

Abstract

In this report we present a robust method to recover 3D position and orientation (pose) of a moving head using a single stationary camera. Head pose is recovered by formulating the problem as a camera pose estimation problem. 3D feature points (artificial or natural occurring) are acquired from the head prior to tracking and used as a model. Pose is estimated by solving a robust version of “Perspective n Point” problem. The proposed algorithm can handle self occlusions, outliers and recover from tracking failures. Results were validated by simulations and by a comparison to an accurate magnetic field 3D measuring device. Our contribution is a system that is not restricted to track only human heads, and is accurate enough to be used as a measuring device. To demonstrate the applicability of our method, three types of heads (human, barn owl, chameleon) were tracked in a series of biological experiments.

1. Introduction

An accurate estimation of head position and orientation (pose) in 3D is important in many applications. Knowledge about gaze direction can be used in human-computer interfaces, video compression and face recognition systems. Furthermore, possible applications are not limited solely to computer science domains. Our motivation is driven by many biological experiments, in which a human observer needs to monitor an experiment and operate accordingly. Such experiments may be fully automated if information about the animal’s gaze was available. Reporting all 6 degrees of freedom (DOF) of the head is important since it enables analysis of movements in the head intrinsic coordinate system, revealing preferred directions of movement.

In this paper, we present a robust approach for tracking a head in video sequences. While previous approaches give only a rough estimate to head pose, and use complex representations such as a triangular mesh [34], a textured cylinder [6, 5] or superquadric [23], we represent the head as a sparse set of 3D points. This set is acquired prior to tracking and used to model head geometry. During tracking, a camera pose estimation problem is solved using the known model points and the
available visible features. Since the camera is static, the recovered camera pose is equivalent to head pose. The two problems of head tracking and camera pose estimation have not been linked before, thus we provide a novel approach of head tracking. The point representation enables an accurate recovery of head pose and the tracking of head with arbitrary geometry (not necessarily human), as will be shown subsequently.

There is a growing increase of interest in gaze tracking systems from many areas, such as movement analysis [24], behavior analysis [13], [18], attention [17] and more. Although the main thrust of this paper is to present a head tracking algorithm, a full gaze tracking system can be implemented easily, since knowledge about head position simplifies the eye tracking problem (see the chameleon experiment in section 7.2).

The rest of the paper is organized as follows. In section 2 we review the relevant previous work on head tracking. In section 3 we present an overview of our system. Section 4 explains how we represent head geometry, and how we acquire such model. In section 5 we formulate the problem as a camera pose paradigm, and present our robust solution to the PnP problem. Methods of tracking features are discussed in section 6. We present three experiments and their tracking results in section 7. A short error analysis is given in section 8. Finally, the conclusions and future plans are discussed in section 9.

2. Previous work

The problem of tracking a human head has received much attention in past years. A coarse classification of algorithms can be made according to the way in which the head is represented. In [3], [25] a 2D ellipse is used to approximate head position in the image. Head position is obtained using color histogram or image gradients. The advantage of such approach is that it is extremely fast, and can be implemented in Real-Time. However, light changes and different color skins result in tracking failures. Another drawback with such approaches is the inability to report head orientation. In [31], [7], [20], [28], partial orientation information, such as tilt or yaw is available. However, the accuracy of those systems is low (up to 15 degree error in estimating rotation). More accurate systems use 3D geometrical models to represent the head as a rigid body. In [34], Ruigang and Zhang use a rough triangular mesh with semantic information. Stereo is used to obtain 3D information, which is matched against the known model. A major shortcoming of this method is the amount of time one need to spend to create a precise model. Recent approaches, such as [6] [5], [26], use a cylinder to approximate both head underlying geometry and texture. A linear combination of motion adaptive templates is used to match the model against current image. Since a cylinder is only a rough approximation to head geometry, those methods suffer from inaccuracies in estimating rotation, and have difficulties differentiating between small rotations and translations. Brolly et al [4] use Nurbs surface with texture to synthesize both appearance and pose, but could not report pose accuracy since ground truth was unavailable. Several researches [23], [9], [35] introduced the notion of extended superquadric surface, or fourier synthesized representation of a surface, which has high degree of flexibility to encompass face structure. They use model induced optical flow to define pose error function. The usage of a parameterized surface enables them to resolve ambiguities caused by self occlusion.

The problem of recovering camera position and orientation relative to a known set of 3D points have been studied by both computer vision and photogrammetry communities. However, it has not been linked to the area of head tracking.
Numerous algorithms exist which recover camera pose, given a set of 3D points and their matched 2D features [14], [8], [21],[12],[30],[36]. The "Perspective n Point" (PnP) problem [30] is an exception, since it uses relative distances between 3D features to estimate pose. This attribute renders the method especially suitable for head tracking since relative distances between 3D points on the head are invariant under rigid body transformation. Furthermore, face deformation can change the 3D spatial configuration of features only by a limited amount. Camera pose can be recovered from as little as three points, although four are needed to resolve ambiguities [33]. Recently, new linear algorithms were developed for the four point problem (P4P), that do not depend on a complex coefficient representation [2]. However, error analysis showed that they have larger error than the basic P3P algorithms. In principle, any solution to the camera pose estimation algorithm that uses only the relative distances, can be used to track rigid body objects.

3. System overview

The information flow in our tracking algorithm is presented in figure 1. Prior to tracking, a set of 3D points is acquired from the head. The relative distances between these points is saved and is used in the actual tracking process. This set of 3D points will be referred as the acquired model. During the actual tracking sequence, a stationary camera views the moving head, and images are passed to the 2D feature detector. Features are extracted from the image according to their spatial characteristics and their predicted locations. The 2D features correspond to 3D points on the moving head. This set of points will be referred as the recovered model. The distance from camera center to 3D points on the head are recovered by solving a robust version of the PnP problem. Head pose is finally obtained by aligning the acquired model and the recovered model. Essentially, the best rigid body transformation, which maps two set of 3D points, is found by solving the Absolute Orientation Problem (see section 5.4). 2D features which were lost due to occlusions are recovered by projecting their 3D position onto the image using the recovered head pose. Ambiguities, resulting from perspective distortion, are resolved by a non linear minimizing. Finally, the 2D positions of features in the next frame are predicted according to a simple motion model, which takes into account the rigid body motion of the head. In the following sections, a detailed description of each module of our system will be given.

4. Model acquisition

While previous approaches use complex surface representations to model head geometry (often, difficult and time consuming to acquire), our representation is sparse and easy to obtain. Only a few 3D points (typically \( M \leq 20 \)) are used to model the head. They can be acquired using one of the many available 3D scanning devices (Stereo, XXX, Multiple view geometry, Laser based techniques, etc). We have selected to acquire the points using several different views [14]. The method is simple, robust, and results in a highly accurate spatial description of the model (up to a few millimeters). A short video sequence of the head is taken, while a checkerboard pattern is visible in the background (figure 2). The camera intrinsic (\( K_{3x3} \) matrix) and extrinsic parameters (\( R_{3x3} \) rotation mateix, \( T_{3x1} \) translation vector) can easily be recovered from the checkerboard pattern [36] Let us denote the \( k \)th 3D point on the head as \( P^k = [X^k, Y^k, Z^k] \). This point is viewed from \( 2 \leq i \leq N \) views. The
Figure 1: Information flow in our tracking algorithm. A model is acquired prior to tracking and saved. During tracking, markers are detected according to their spatial appearance and predicted positions. Head pose is estimated by solving a robust version of the PnP problem. Missing features are recovered and ambiguities due to merge events are solved. Finally, predicted position of features are extrapolated using previous head poses.
corresponding 2D feature is denoted as \( p^k_i = [x^k_i, y^k_i] \). Our goal is to recover 3D position \((P^k)\) of \(M\) features. Note that both intrinsic and extrinsic parameters in each frame are known. Let \(R_i\) and \(T_i\) denote orientation and position of the camera in frame \(i\) respectively. The projection matrix of each frame is defined by:

\[
M_i = K \begin{bmatrix} R_i & T_i \end{bmatrix}
\]

Each feature point is projected using the following non-linear projective transformation:

\[
\Phi(x, y, z) = \frac{1}{z}(x, y).
\]

Therefore, point \(p^k_i\) is linked to its 3D description \(P^k\) by:

\[
p^k_i = \Phi(M_i P^k_c).
\]

Since we only know the approximations to \(M_i\) and \(p^k_i\), we denote them as \(\tilde{M}_i\) and \(\tilde{p}^k_i\). The approximation suffers from noise due to errors in solving the external camera parameters and detection of features in the images. The minimum square solution to the reconstruction problem is obtained by optimization over the following function:

\[
\argmin_{P^k} \sum_{k=1}^{N} \sum_{i=1}^{M} \|\tilde{p}^k_i - \Phi(\tilde{M}_i P^k_c)\|^2.
\]

We wish to find the best estimation of \(P^k_c\) which minimizes equation 4. This problem can be solved using non-linear optimization methods such as Levenberg-Marquardt or Gauss-Newton. Gauss-Newton is more suitable for this problem, since it is formulated as a min-squares problem. The main disadvantage of using these methods is that they require initial solution. The initial solution should be very close to the optimum, as this problem is non-convex.
4.1 Linear initial guess

Linear triangulation method is used to obtain an initial solution to the iterative optimization process. A coefficient matrix is constructed such that the sought for solution lies in its null space. First, equation 3 is rewritten in the following form:

\[ \tilde{p}_i^k \times \Phi(\tilde{M}_i P^k) = 0, \quad (5) \]

or alternatively, in matrix form:

\[
\begin{bmatrix}
\tilde{x}_i^k \tilde{M}_i^3 - \tilde{M}_i^1 \\
\tilde{x}_i^k \tilde{M}_i^3 - \tilde{M}_i^2
\end{bmatrix}
\begin{bmatrix}
X^k \\
Y^k \\
Z^k \\
1
\end{bmatrix} = A_i [P^k, 1]^T = 0, \quad (6)
\]

where \( \tilde{M}_i^q \) marks the q’th row of matrix \( \tilde{M}_i \). We can stack all \( A_i \) into one big matrix \( A \), obtaining the following constraint on \( P^k \):

\[
\begin{bmatrix}
A_1 \\
\vdots \\
A_M
\end{bmatrix} [P^k, 1]^T = A [P^k, 1]^T = 0. \quad (7)
\]

The result is an over determined system where \( P^k \) define the null space. The trivial method of solving this system is the Singular Value Decomposition of A. Observe that A is of rank 4. Let us denote the eigen-vector which correspond to the smallest eigen-value as \( V_k = [v_k^1, v_k^2, v_k^3, v_k^4] \). The the best linear estimation to \( P^k \) is obtained by normalizing \( v_k^4 \) to unity:

\[ \tilde{P}^k = \frac{1}{v_k^4} [v_k^1, v_k^2, v_k^3]. \quad (8) \]

This is actually, the least-squares solution to equation 7. The disadvantage of this method is that it is highly noise sensitive. If a single row has a very large error the quality of solution will be poor. A more robust (non-linear) method is to solve the following robust minimization problem:

\[ \min_{\tilde{P}^k} \phi \left( A \begin{bmatrix} P^k & 1 \end{bmatrix}^T \right), \quad (9) \]

where \( \phi \) is a robust estimator, like Huber function.

4.2 Model alignment

The set of points \( P^k \) describes the model relative to a world reference frame, located at one of the checkerboard squares. This representation is not intrinsic. It would be more meaningful to represent the head in a head-centered coordinate system (figure 3). Selecting such system is not a trivial task, since in most cases, the rotation point is not visible, or there is no such point. Nevertheless, a close approximation to it can be obtained from visible head features (two eyes and mouth, or beak in the case of the owl). It is important to describe the model in a head-centered coordinate system, of meaningful measurements and conclusions are to be drawn from the data. We denote the head-centered acquired model as \( P_M^k \).
Figure 3: The model is represented in a head-centered coordinate system. The owl’s intrinsic coordinate system was determined according to visible head elements. The pitch axis passes through the center of both eyes. The roll axis was determined from behavior experiments, and was found to be approximately 25 degrees elevated from the line joining the center between the eyes and the tip of the beak (marked as V).

5. Head pose estimation

In this section, we describe the process of estimating the head pose. The 3D model points \( P^k_M \), which are represented in a head-centered coordinate frame, are searched in the image according to their 2D spatial appearance (chroma or any other cue). Once the 2D image features \( p_k \) are matched to 3D model points, the distances from camera center to the 3D head points are recovered by solving the a robust version of the PnP problem. The coordinates of the recovered 3D head points \( P^k_R = [X^k_R, Y^k_R, Z^k_R] \) are finally matched to the 3D model points. The rigid body transformation which maps between the acquired model and the recovered model is found by solving the absolute orientation problem. This yields the rotation \( R \) and translation \( T \), that represents the head pose in the current frame.

5.1. The perspective 3 point problem

The problem of obtaining the distances from the camera center to 3D points has been studied by both the computer vision and the photogrammetry community. In the following we give a review over the problem and the common solution, obtained using elimination theory. We assume that the Euclidean distance between each pair of model points \( (u, v) \) is known:

\[
d_{u,v} = \|P^u_M - P^v_M\|.
\]  (10)

Camera center is denoted by \( C \), and we will assume from hereon that \( C = [0, 0, 0] \). Therefore, distances are measured relative to the camera center. According to the cosine theorem

\[
\|P^u_R - P^v_R\|^2 = \|P^u_R\|^2 + \|P^v_R\|^2 - 2 \|P^u_R\| \|P^v_R\| \cos \angle P^u_R CP^v_R.
\]  (11)
Left hand side of equation 11, can be expressed in terms of the acquired model
\[
\|P^u_R - P^v_R\| = \|P^u_M - P^v_M\| = d_{u,v}. \tag{12}
\]
We assume that camera internal parameters are known and represented as a 3x3 matrix \(K\). The direction vector, from camera center \(C\) to feature \(u\) is given by:
\[
N(u) = K^{-1} [pu, 1]^T. \tag{13}
\]
Therefore, the angle \(\angle_{P^u_RCP^v_R}\) can be found from the two direction vectors \(N(u)\) and \(N(v)\):
\[
\cos \angle_{P^u_RCP^v_R} = \frac{N(u) \cdot N(v)}{\|N(u)\| \|N(v)\|}. \tag{14}
\]
We can reorganize the equations and obtain a single polynomial in two unknowns of the form:
\[
f_{u,v}(x_u, x_v) = x_u^2 + x_v^2 - 2x_u x_v \cos \theta_{u,v} - d_{u,v}^2 = 0, \tag{15}
\]
where \(x_k = \|P_k\|\). When three features \((u, v, w)\) are available, the cosine theorem (equation 11) can be formulated for pairs \((u, v), (u, w), (v, w)\), to obtain three polynomials in three unknowns \((x_u, x_v, x_w)\):
\[
\begin{cases}
f_{u,v}(x_u, x_v) = 0 \\
f_{v,w}(x_v, x_w) = 0 \\
f_{u,w}(x_u, x_w) = 0
\end{cases}. \tag{16}
\]
We end up with a set of three quadratic equations in three unknowns \((x_u \equiv \|P^u_R\|, x_v \equiv \|P^v_R\|, x_w \equiv \|P^w_R\|)\). This system can be solved using a method from elimination theory, which will be described next.

5.2. Elimination theory and sylvester resultant
Algebra elimination theory provides mathematical tools to eliminate a specific variable from a set of polynomials. Let us assume we have two polynomials of the form:
\[
\begin{cases}
f(x) = \sum_{i=0}^{m} a_i x^i \\
g(x) = \sum_{j=0}^{n} a_j x^j
\end{cases}. \tag{17}
\]
The Sylvester matrix of \(f(x)\) and \(g(x)\) is:
\[
S(f, g) = \begin{bmatrix}
a_m & a_{m-1} & \ldots & a_1 & a_0 & 0 & \ldots & 0 \\
0 & a_m & a_{m-1} & \ldots & a_1 & a_0 & \ldots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \ldots & 0 & a_m & a_{m-1} & \ldots & a_1 & a_0 \\
b_n & b_{n-1} & \ldots & b_1 & b_0 & 0 & \ldots & 0 \\
0 & b_n & b_{n-1} & \ldots & b_1 & b_0 & \ldots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \ldots & 0 & b_n & b_{n-1} & b_1 & b_0 & 0
\end{bmatrix}. \tag{18}
\]
The Sylvester Resultant is the determinant of the Sylvester matrix. A basic theorem about the Sylvester Resultant states that \( f(x) \) and \( g(x) \) have a common root, if and only if \( |S| = 0 \). We can use this theorem for polynomials with more than one variable, by selecting one variable with coefficients polynomials in the other variables. The variable we selected will be eliminated from the two polynomials, and we will get a new polynomial, of degree \( n + m \), that is constructed from the coefficients of the other variables.

Let us return to the original problem we were facing. We had a set of three quadratic polynomials, and each polynomial had two unknowns. Using the resultant twice, we can eliminate two of the variables \((x_v, x_w)\) and reduce the problem to a single, forth degree polynomial in single new variable \( x \):

\[
g(x) \equiv \sum_{i=0}^{4} a_i(x)^i,
\]

such that:

\[
g(x_u) = 0, \quad x_u \equiv \|P^u_R\|^2.
\]

The root \( x_u \) of the computed polynomial corresponds to the squared distance from camera center to the 3D point \((\|P^u_R\|^2)\). In the same manner, we can eliminate variables \( x_u, x_w \) and solve for \( x_v \), or eliminate \( x_u, x_v \) and solve for \( x_w \). Note that there are four possible solutions to this polynomial. A geometrical interpretation to the ambiguity in the solution can be found in [33]. Additional information must be given to select the correct solution from the four possible ones.

5.3. The perspective n point problem (PnP)

Three points are not sufficient to solve the PnP problem, since the forth degree polynomial can have up to four different real roots, all correspond to possible true positions of the head. Four points are sufficient for an unambiguous solution, unless they lie in a degenerate configuration \((C, P^u_R, P^v_R, P^w_R)\) are on a plane). However, our goal is to use all possible points, and to obtain the most accurate solution. If \( N \geq 4 \) un-occluded points are available, then \( \lambda \equiv \binom{N-1}{2} \) forth degree polynomials can be constructed, as was explained above. The unique solution is the common root of all these polynomials.

The polynomial coefficients will suffer from noise that is originated from inaccuracies in detecting feature points, and inaccurate model. This noise will result in a drift of the true common root. Quan and Lan proposed in [30] a linear solution which stacks the \( \lambda \) forth degree polynomial coefficients into a single matrix. Their solution was obtained using the SVD of the coefficient matrix. There are two drawbacks of this approach. It is not stable when outliers are present and a positive solution is not guaranteed. Inspired by [29], we use robust M-estimator \( \rho \), such as Huber, to overcome the above difficulties. A robust estimation of \( x_u \) is obtained by minimizing

\[
x_u = \arg\min x_u \sum_{j=1}^{\lambda} \rho \left( \sum_{i=0}^{4} a_i^j x^j \right).
\]

This problem is non convex and has multiple local minima. Therefore, an iterative method (Gauss-Newton) is applied with Quan and Lan’s [30] linear solution as initial guess.
Once \( x_u \) is obtained, the 3D coordinate of \( P^u_R \) can be calculated by:

\[
P^u_R = \frac{N(u)}{\|N(u)\|} \sqrt{x_u} = \frac{N(u)}{\|N(u)\|} \|P^u_R\|.
\]  

(22)

This process is repeated for each point \( 1 \leq u \leq N \), until all visible points of the recovered model are obtained.

5.4. Absolute orientation and head pose

Head pose is represented as the rigid body transformation which rotates \((R)\) and translates \((T)\) the acquired model, to align with the recovered model. The two models are represented using a set of 3D Points and are connected via the following equation:

\[
P^u_R = RP^u_M + T.
\]  

(23)

The problem of finding the best \( R \) and \( T \) which align the two models is a well known problem in computer vision and is called Absolute Orientation problem. The minimum square formulation is given by

\[
\arg\min_{R,T} \sum_{u=1}^{M} (P^u_R - (RP^u_M + T))^2.
\]  

(24)

Several solutions are known ([10], [15],[16], [19]). We have selected the method of Faugeras, who found a close form solution using quaternions. Let us first assume that the two models are centered about their centroid. In that case, the problem is reduced to finding only the rotation. The 3x3 orthogonal rotation matrix \( R \) can be represented as a unit quaternion \( q \). Therefore,

\[
\arg\min_{q,s.t.\|q\|=1} \sum_{u=1}^{M} (q * P^u_M - P^u_R)^2
\]

(25)

where \( \bar{q} \) represents the quaternion conjugate and \(*\) denotes quaternion multiplication (for more details on quaternion algebra, please refer to [32]). The term \( q * P^u_M - P^u_R * q \) is linear in \( q \) and can be written as:

\[
\begin{align*}
\arg\min_{q,s.t.\|q\|=1} \sum_{u=1}^{M} (A_i q)^2 &= \arg\min_{q,s.t.\|q\|=1} \sum_{u=1}^{M} q^T A_i^T A_i q = \arg\min_{q,s.t.\|q\|=1} q^T \left( \sum_{u=1}^{M} A_i^T A_i \right) q \\
&= \arg\min_{q,s.t.\|q\|=1} q^T B_{4x4} q
\end{align*}
\]

(26)

Differentiation by \( q \) results in:

\[
\arg\min Bq = 0
\]

\[
s.t. \quad \|q\| = 1
\]

(27)

This can be solved using the Singular Value Decomposition of \( B \):

\[
\arg\min_{s.t.\|q\|=1} \|Bq\|^2 = \arg\min_{s.t.\|q\|=1} \|USVq\|^2 = \arg\min_{s.t.\|Vq\|=1} \|SVq\|^2 = \arg\min_{s.t.\|y\|=1} \|Sy\|^2.
\]  

(28)

The solution is \( y = (0, .., 1, 0, ...0) \), where the position of the which is the eigen-vector which corresponds to the smallest eigen-value.
Once rotation is known, we are left with the problem of recovering the best fit translation between the two models. A closed form solution for the translation, adopted from [15], is given by:

\[ T = \bar{P}_R - R \bar{P}_M, \]  

(29)

where \( \bar{P}_R \equiv \frac{1}{M} \sum_{i=1}^{M} P^i_R \) and \( \bar{P}_M \equiv \frac{1}{M} \sum_{i=1}^{M} P^i_M \).

The method above is linear, and has the advantage of being extremely fast. However, it is noise sensitive. Therefore, a more accurate solution can be obtained by projecting the 3D recovered model to the image and minimizing the re-projection error:

\[ \min_{R,T} \sum_u \rho \left[ \rho_u - \Phi_{R,T}(P^u_M) \right], \]  

(30)

where \( \Phi_{R,T} \) is the perspective projection operator and \( \rho \) is a robust M-Estimator. This is a non-linear minimization, therefore, the initial guess used is the linear solution (described above).

### 6. Feature tracking and recovery

Up until now, we have neglected the issue of detecting and registering low level features that are used in the pose estimation algorithm. However, for the algorithm to work properly, it must keep track of at least three visible features at any given moment. The features are hand-marked in the first frame of the video sequence, and are automatically tracked from thereon. Features are detected according to their spatial appearance. The algorithm can use either artificial markers, or natural occurring. When artificial markers are used, they are detected according to their color (hue, saturation), size and shape properties. The feature tracking module find candidates according to their spatial appearance and selects the most probable one according to the predicted positions and proximity to their positions in the previous frame.

In our experiments we used natural occurring features as well. The advantage of using artificial markers is that you can easily detect them and position them at preferred positions. However, due to the nature of the application, it might not be feasible to use artificial markers. In this case, natural occurring features can be extracted from a single representative image. The natural occurring features are basically local maxima in scale-space. For each local maxima that passes a certain threshold, additional information is saved, such as at which scale it was detected, orientation of neighboring pixels in that scale, etc. Therefore, each feature has not only a gray-scale value, but an increased vector of characteristics. The high dimensionality vector makes registration of such features more robust. It is less likely that two features will have the same descriptor. In our implementation, we have used SIFT descriptors [22], to detect natural occurring features in a human face. Figure 4 shows all natural occurring features that were detected in a representative image of the head. These features represented the acquired model. The 3D position of all features marked in yellow were estimated using structured light technique, which is described elsewhere [1]. In each new video frame, SIFT features were matched against the acquired model.
6.1 Feature prediction

The predicted position of a 2D feature is calculated by projecting the corresponding 3D model point using an extrapolated pose. Let denote by \( P_{i-1} = [R_{i-1}, T_{i-1}], P_{i-2} = [R_{i-2}, T_{i-2}] \) the two previous poses of the head. The goal is to derive the predicted pose for the current frame \( P_i \). The translational component of the extrapolated pose is calculated using a simple linear model. Therefore,

\[
T_i = T_{i-1} + (T_{i-2} - T_{i-1}).
\]  

However, predicting rotation using extrapolation Euler angles yielded poor results. Interpolating Euler angles is known to give a non natural rotational movement, therefore, it can not be expected that extrapolation of Euler angles will results in a good prediction. Quaternions, on the other hand, have been used in computer graphics, to create a smooth movement of rigid objects. We have adopted the SLERP (Spherical Linear Interpolation) method of Shoemake [32], and have created a similar extrapolation scheme. Given \((Q_{i-1} \text{ and } Q_{i-2})\), the extrapolated orientation, represented as a quaternion, can be represented by:

\[
Q_i = \sin ((1 - 2t) \Theta) \frac{Q_{i-2}}{\sin \Theta} + \frac{\sin (2t\Theta)}{\sin \Theta} Q_{i-1}
\]

\(t = 1\).

This equation is actually a modified interpolation at time \( t = 2 \). The extrapolated quaternion lies on the great circle arc which connects \( Q_{i-1} \) and \( Q_{i-2} \), at a distance of \( \Delta Q = acos (Q_{i-2} \circ Q_{i-1}) \), from \( Q_{i-1} \).

6.2 Recovery of lost features

During tracking several features may be lost due to occlusion, self-occlusion, change of light, or many other factors. One of the key strengths of our algorithm is the ability to recover lost features. Since the 3D structure of feature points is known,
we can project the model using the recovered pose, and know exactly where they are in the image, even if they are occluded. At the end of each pose estimation iteration, features which were lost are recovered by searching at the known positions. When a lost feature is detected near the predicted position, it is considered a candidate. Only candidates which reduce the re-projection error (Equation 30) are further tracked.

When artificial features with same spatial appearance are used, additional problem may arise, which is ambiguity in the registration process. Such ambiguities arise when two tracked features get too close to each other. The two features "merge" into a single spatial position due to perspective distortion. Our algorithm identifies these merge events and solves the ambiguity by choosing the correspondence combination which minimizes the re-projection error.

7. Experimental results

Three types of heads were successfully tracked in a series of experiments. Implementation was done in MatLab. Video sequences resolution was 720x576 pixels. There is a tradeoff between the accuracy of the pose and the number of iterations used to solve the non-linear optimization problems. However, under reduced resolution and reasonable accuracy, we managed to obtain head pose at a frame rate of approximately 3-4 fps with a Pentium 1.4Ghz. However, when merge events were detected, performance dropped dramatically since ambiguities had to be resolved. We strongly believe that a better implementation under C++ will yield a significant improvement in running time.

7.1. Barn owl experiments

The motivation for head tracking came from a recent investigation of barn owls’ head movements. Barn owls can not move their eyes, thus gaze can be inferred solely from their head pose. Owls exhibit strange head movements prior to prey capture while they sit on a perch and observe their prey. These pre-attack head movements were successfully tracked and analyzed with our system in a recent biological study [27]. High pose accuracy (2 mm in position, 1 deg in orientation) was obtained with the usage of blue circular artificial markers. These could easily be detected using their Hue-Saturation values. More than 30 video sequences with an average length of 27 ± 16 s, were successfully analyzed. Although many of the sequences contained missing features due to self occlusions, our system successfully managed to track the head (figure 5). Features positions are marked in yellow dots. Notice that their 2D position is known even if they are occluded. For more information regarding the biological findings in this work, please refer to [27].

7.2. Chameleon experiments

In another series of experiments, both eyes and head of a chameleon were tracked (additional module was written to determine pupil’s position). The goal was to measure correlations between independent eye movements, relative to the head pose. Several artificial rectangular markers were attached to the head and used as markers. A 19 minute sequence was successfully with our system (figure 6). The direction of the eye is drawn as a yellow cone in the left most picture.
Figure 5: Several frames from a tracked sequence of a barn owl pre-attack behavior. The owls’ coordinate system is shown in black. The artificial markers are simple blue paper stickers. The yellow dots appear in the expected position of a feature, even if the feature is completely occluded.

Figure 6: Several frames from a tracked sequence of a chameleon. In this experiment, the features were small rectangles attached to the animal’s head. The coordinate system is shown in black. In the right most image the orientation of the eye, relative to head is shown as a yellow cone.
7.3. Human experiments

To demonstrate that head pose can be tracked using natural occurring features we tracked a human head without any attached markers. More than 100 SIFT descriptors were acquired and stored as a model (figure 4), along with their 3D relative distance. However, only 5-15 could be reliably matched during the actual tracked sequence (figure 7, detected SIFT descriptors are marked in yellow dots). The recovered pose was typically less accurate than the one obtained in the two experiments described above. This can be seen in figure 7, where the center of coordinate system shifts, and it not fixed on the tip of the nose.

8. Pose accuracy analysis

We have used three ways to evaluate the accuracy of our algorithm. First, by simulations with synthetic data. However, such simulations can be influenced by many factors, such as the shape of the tracked object, angle of view, etc. We therefore, decided to use a real calibrating object (figure 8), for evaluation of performance. The calibrating object was a small cup, with artificial markers attached. 18 Images were analyzed, at different (discrete) known positions and orientations of the object. Under these “ideal” conditions, the positional error had a standard deviation of 2.3mm and orientation error had a standard deviation of $0.47^\circ$. These values, however, might not reflect the best possible performance, since they refer to “absolute” error, from a known reference frame, at discrete poses. Further validation of our results was obtained by a comparison with one of the commercially available 3D magnetic field measuring devices (miniBIRD - Ascension Technologies). This highly accurate (1.8mm in position, $0.5^\circ$ orientation) system has a small wired sensor which measures the changes in the magnetic field induced by a near-by transmitter. We have attached a calibrating object to a moving platform which could move only
along the Y direction and recorded simultaneously. The results are summarized in figure 9, and indicate that our system has a comparable performance to the miniBIRD. Our system suffers more from noise along the Z direction, while the miniBIRD system is less accurate in reporting orientation.

9. Conclusion and outlook

In this paper we have proposed a novel method to track a head with arbitrary geometry in 3D by solving the PnP problem. High degree of accuracy was obtained using a robust solution to the PnP and recovery of lost features. One limitation of our system is that it lacks a proper temporal filter since extrapolated pose is computed only from two previous frames. Future additions may include a Kalman filter that might improve predicted pose. Furthermore, SIFT descriptors are only one option to detect natural occurring features. Future research will focus on detecting natural occurring features using more sophisticated methods. The development of a full gaze tracking application is also a possibility. The knowledge of head pose can give a rough estimate to eye position, making it easier to build eye tracking module. In the future we will try to implement an eye tracking module for humans, and hope to achieve the same high accuracy without the need of artificial markers.

10. Acknowledgements

The authors would like to thank Ofir Avni, for allowing us to use the results obtained in his experiments with the chameleons.
Figure 9: Simulations recording using our system and a 3D magnetic field measuring device
References


