Abstract—In this paper, we propose a hybrid model-based tracker for simultaneous tracking of 3D head pose and facial actions in sequences of texture and depth frames. Our tracker utilizes a generic wireframe model, the Candide-3, to represent facial deformations. This wireframe model is initially fit into the first frame by an Iterative Closest Point algorithm. Given the result after the first frame, our tracking algorithm combines both Iterative Closest Point technique and Appearance Model for head pose and facial actions tracking. The tracker is capable of adapting on-line to the changes in appearance of the target and thus the prior training process is avoided. Furthermore, the tracking system works automatically without any intervention from human operators.

I. INTRODUCTION

Tracking human faces has remained an active research area among the computer vision community for a long time due to its usefulness in a number of applications, such as video surveillance, expression analysis and human-computer interaction. An automatic vision-based tracking system is desirable and such a system should be capable of recovering the head pose and facial features, or facial actions. It is a non-trivial task because of the highly deformable nature of faces and their rich variability in appearances.

Face tracking has been performed accurately by feature-based trackers [1][2]. In [1] a two-stage approach was developed for 3D head pose and facial features tracking in monocular video sequences. 3D facial deformations are learned from stereo data and the features are tracked by optical flow. In [2], tracking is obtained via a set of linear predictors modeling pixel intensities and a statistical method is used to identify relevant areas to locate feature points.

Also based on feature points and their local information, Active Shape Models (ASMs) [3] use a point distribution model to capture shape variations. The shape consists of a set of landmarks with their local appearance distributions. Shape parameters are updated by iteratively searching the best nearby match for each landmark point. ASMs are simple and fast. However, they require a large amount of annotated training data to learn shape models, and by their inherent approach ASMs only use image information sparsely.

Active Appearance Models (AAMs) were proposed to overcome the limitation of ASMs by adopting statistical texture models. In [4][5], both shape and texture distribution models are learned from training data. Alternatively, in [6][7], a template is used in place of shape models and texture distribution models are mapped onto that template. In contrast to ASMs, parameters are updated iteratively by minimizing the difference between the incoming texture and the texture mean which was learned from training data. To avoid the training process of AAMs, some authors have proposed On-line Appearance Models (OAMs) [8][9] for tracking. In these works, the appearance distribution is constantly updated to reflect changes over time and compensate for drifting. However, a good initialization is required, and often it is performed manually.

In recent years, affordable range sensors have been introduced, such as the Microsoft™ Kinect and similar depth cameras. Face tracking research has since adopted 3D deformable model fitting techniques to take advantage of the now available depth information [10][11]. These works were based on Iterative Closest Point (ICP) procedure [12][13][14]. In [10], an ICP-based regularized maximum likelihood deformable model fitting algorithm was developed. Feature points are tracked across frames using optical flow and integrated into the framework to track facial movements. This method, while efficient, relies on optical flow tracking of feature points which often deteriorates over time, causing inaccurate deformations. Following a more graphics-based approach, the authors of [11] developed a face tracking system aiming toward facial animation. Initially, a set of specific facial poses are acquired from the human subject. These poses are then combined to form the final pose in the tracking process. Weights of shapes are learned from maximum a posteriori estimate. Their tracking algorithm is robust, however it requires a set of specifically designed blendshapes, and there is still a short off-line training procedure before it can work.

In this paper, we propose a hybrid on-line 3D face tracker to take advantages of both texture and depth information, which is capable of tracking 3D head pose and facial actions simultaneously. First, we employ a generic deformable model, the Candide-3, into our ICP fitting framework. Second, we introduce a strategy to automatically initialize the tracker using the depth information. Lastly, we propose a hybrid tracking framework that combines ICP and OAM to utilize the strengths of both techniques. The ICP algorithm, which is aided by optical flow to correctly follow large head movement, robustly tracks the head pose across frames using depth information. It provides a good initialization for OAM. In return, the OAM algorithm maintains the texture model of the face, adjusts any drifting incurred by ICP and transforms the 3D shape closer to correct deformation, which then provides ICP with a good
initialization in the next frame. Throughout this paper, it is assumed that we know the alignment and projection between color frame, depth frame and 3D world coordinates.

II. THE CANDIDE-3 WIREFRAME MODEL

A. Face Modeling and Parameterization

Candide-3 is a simple, generic wireframe model (WFM) developed by J. Ahlberg [15]. The Candide-3 WFM consists of 113 vertices, and 184 triangles constructed from these vertices that define its surface, as shown in Fig.1(a).

The Candide model was originally designed for model-based face coding and its deformations are directly related to face biometry and facial expressions. The 3D shape of the model is controlled by a set of Action Deformation Units (AUs) and Shape Deformation Units (SUs). Since every vertex can be transformed independently, each vertex of the model is reshaped according to:

$$g = p_0 + S \sigma + A \alpha$$

where $p_0$ is the base coordinates of a vertex $p$, $S$ and $A$ are shape and action deformation matrices associated with vertex $p$, respectively. $\sigma$ is the vector of shape deformation parameters and $\alpha$ is the action deformation parameters vector. In general, the transformation of a vertex given global motion including rotation $R$ and translation $t$ is defined as:

$$p' = R(p_0 + S \sigma + A \alpha) + t$$

Notice that we exclude the scaling factor $s$ due to: 1) human heads have similar size in most cases thus we can calibrate the scale beforehand, and 2) shape deformations will take care of micro-scaling through the fitting procedure. The geometry of the model is parameterized by vector

$$u = [\theta_x, \theta_y, \theta_z, t_x, t_y, t_z, \sigma^T, \alpha^T]^T$$

where $\theta_x$, $\theta_y$, $\theta_z$ are three rotation angles, $t_x$, $t_y$, $t_z$ are three translation values corresponding to three axes x, y and z.

B. Shape Deformation Formulation

Let us assume that the current shape is defined by a transforming function $T$: $P = T(P_0, u)$. Given new data, we would like to find a new parameters vector $u'$ to make the new shape $P' = T(P_0, u')$ fit new data best. As in any fitting procedures, we want to find a displacement vector $\Delta u$ such that $u' = u + \Delta u$. Substituting $u'$ into Eq. (2) we have:

$$P' = R'[\sum_i S(\sigma + \Delta \sigma) + A(\alpha + \Delta \alpha)] + t + \Delta t$$

We make another assumption that the rotation from the current shape $P$ to the new shape $P'$ is relatively small, and we can use the following approximation:

$$R(\theta_x + \Delta \theta_x, \theta_y + \Delta \theta_y, \theta_z + \Delta \theta_z) = \Delta R \times R(\theta_x, \theta_y, \theta_z)$$

$$\Delta R \approx I + \Delta \theta_x \times R_x + \Delta \theta_y \times R_y + \Delta \theta_z \times R_z$$

Substituting (4) and (5) back into (3), we can compute the new shape $P'$ as:

$$P' \approx P + \left( \sum_{i} \Delta \theta_i \times R_i \right) \times P + S \Delta \sigma + A \Delta \alpha$$

From now on when finding the transformation from shape $P$ to shape $P'$, we will find the unknown vector $(\Delta \theta^T, \Delta t^T, \Delta \sigma^T, \Delta \alpha^T)^T$, whose initial values are zeros.

C. Initialization and Tracking Steps

As being stated in Section II-A, the deformations of the Candide model are designed to cover face biometry (SUs) and facial expressions (AUs). Each person has a unique set of corresponding shape units, whereas facial expressions, or action units, are person-independent. However, according to the transformation in Eq.(6), we optimize both shape and action parameters simultaneously, and it is difficult if not impossible to find the unique set of shape parameters for the test subject. Thus, we use the first frame to calculate shape parameters; from the second frame onward, we keep shape parameters fixed and only optimize on action parameters. **Initialization:** set action parameters to zeros and optimize on $(\Delta \theta^T, \Delta t^T, \Delta \sigma^T, \Delta \alpha^T)^T$. It requires neutral expression in the first frame. **Tracking:** keep shape parameters unchanged and optimize on $(\Delta \theta^T, \Delta t^T, \Delta \sigma^T, \Delta \alpha^T)^T$. In this paper, we track 7 facial actions as shown in Figure.1(b-h).

III. INITIALIZATION

In this section, first, we take advantage of the depth information to efficiently find the initial head pose using an SVD-based registration technique and a set of feature points detected from the input texture frame. Next, we introduce our ICP fitting procedure to optimize transformation parameters on the input point cloud.
update the transformation by calculating new values of correspondences set $d$. In general, the ICP algorithm iterates two steps: starting from initial estimate of $u$, it repeats the C-step: compute the correspondences set $d_i$ from input point cloud; and the T-step: update the transformation by calculating new values of $u$; until convergence.

**C-step**: in this step, with the point cloud $D$, we find the correspondence $d_i$ of each vertex $p_i$ such that $d_i = \arg \min_{d_j} \|d_j - p'_i\|_2$, $\forall d_j \in D$ with $p'_i = T(p_i, u_t)$, $u_t$ are current parameters. We make use of the matching method in [13]. Searching for closest points is accelerated using K-D tree. After we have found all correspondences $d_i$, we discard pairs of outliers which has $\|d_i - p'_i\|_2 \geq D_{\text{max}}$. The threshold $D_{\text{max}}$ is updated after each ICP iteration; for more details please refer to [13]. At the end of this process we retrieve a subset of $N_d$ pairs of the initial correspondences set. This filtering method makes ICP more robust and we do not have to treat vertices hidden by head pose in any special way, they are rejected automatically since their corresponding pairs will have larger distances than the rest.

**T-step**: in this step we find the optimal parameters: $\tilde{u}_{t+1} = \arg \min_u \sum_{i=1}^{N_d} (d_i - T(p_i, u))^2$.

Both steps try to reduce the error, thus convergence to a local minima is guaranteed.

In our framework, minimizing the error $E(u)$ in the T-step is a non-linear least-squares problem as $T$ is a non-linear function. We choose the Levenberg-Marquardt algorithm (LMA)[20] to solve the optimization in Eq.(8). It is more robust than the standard Gauss-Newton method, and LMA can find the solution even when it starts far off the final optima, which suits our tracking problem when changes between frames may be drastic in some cases. Interested readers are encouraged to refer to [12] and [20] for more details. Note that in our ICP procedure, the Jacobian matrix can be computed analytically using the formulation in Eq.(6).

**Algorithm 1**: The initialization algorithm.

<table>
<thead>
<tr>
<th>Data:</th>
<th>input point cloud, texture image of the first frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result:</td>
<td>parameters vector $u_1$</td>
</tr>
<tr>
<td>while not converged do</td>
<td></td>
</tr>
<tr>
<td>Find correspondences set from point cloud;</td>
<td></td>
</tr>
<tr>
<td>Discard outliers from the set;</td>
<td></td>
</tr>
<tr>
<td>Update Transformation;</td>
<td></td>
</tr>
<tr>
<td>end</td>
<td></td>
</tr>
</tbody>
</table>

**IV. TRACKING**

We introduce a two-step optimization procedure for our model-based tracking framework. First, an ICP algorithm calculates the deformation parameters from the point cloud of the current frame. After that, an appearance-based algorithm optimizes the deformation parameters once more over the current texture frame to rectify any error incurred by ICP.

**A. Iterative Closest Point for Tracking**

The ICP procedure in the tracking phase is basically as same as in the initialization phase, the only difference is in the first iteration. The rigid motion, especially translation of the head from frame $t-1$ to frame $t$ may be large and if we apply parameters retrieved from the previous frame as initialization to ICP, the algorithm would likely not converge...
to the correct solution. To quickly compensate for this possibly large movement, we use the following method: Given the transformed shape in frame \( t-1 \), we project every 3D vertex of the Candido model to the color frame \( t-1 \) to form a set of \( N_f \) 2D points \( z^{2D}_i \) \( (N_f \leq 113) \). We then track this set of feature points from color frame \( t-1 \) to frame \( t \) using optical flow. After positions of these feature points in the current frame are recovered, we back-project them to 3D world coordinates to form a set of \( N_f \) 3D correspondences \( x^{3D}_i \) and use them in the first iteration of ICP, instead of searching for closest points from the point cloud. The ICP tracking procedure is given in Algorithm 2.

In this tracking step, we keep the facial action deformations remain human-like when drifting happens by enforcing upper and lower bounds on seven action deformation parameters. We use a variant of the Levenberg-Marquardt algorithm with box-constraints [21] for optimization.

However, optical flows of these feature points deteriorate after each frame. After several frames, these feature points may drift far away from their supposedly actual positions. Thus, we implement an appearance-based algorithm which is explained further in Sec. IV-B to adjust the transformation parameters closer to correct values. If the feature points in frame \( t \) are placed correctly, there will be very little drift in frame \( t+1 \), and the ICP algorithm will likely converge to the right solution. Furthermore, since ICP has provided a good initial estimate for the appearance-based algorithm, it will also converge faster.

### B. On-line Appearance Model

The On-line Appearance Model (OAM) [8][9] is developed upon the principles of the original Active Appearance Models [4][5][6][7]. In contrast to traditional AAMs, the mean texture is defined by a fixed-sized template and it is updated over time by employing a particle filter-like approach.

Given a new input image \( I \) and the corresponding deformation parameters vector \( u \), we apply a piece-wise affine warping function \( \psi(I,u) \) similar to AAMs to map pixels of each triangle of the 3D wireframe model onto the corresponding triangle of the 2D template. The function results in a fixed-sized observation texture vector \( \chi \). This texture vector is then normalized to \( \chi \sim \mathcal{N}(0,1) \) to partially compensate for different lighting conditions. The warping process is demonstrated in Figure 3.

\[
\chi \leftarrow \psi(I,u) \tag{9}
\]

We do not track eyelids and irises, thus we exclude their pixels from the template to eliminate their effects on optimization.

In the OAM framework, the appearance model at time \( t \), or the mean texture \( \mu_t \), varies over time and models the appearance presentation in all observations \( \chi \) up to frame \( t-1 \). All normalized texture vectors are modelled using a multivariate Gaussian distribution with independent variables and diagonal covariance matrix \( \Sigma \). Each pixel follows a Gaussian distribution \( \chi_i \sim \mathcal{N}(\mu_i, \sigma_i) \). The observation likelihood at time \( t \) is defined as:

\[
p(I_t|u_t) = p(\chi_t|u_t) = \prod_{i=1}^{n} \mathcal{N}(\chi_{it}; \mu_{it}, \sigma_{it}) \tag{10}
\]

After recovering the deformation parameters vector \( u_t \) of the current frame, we also get the new observation \( \chi_t \). We update the Gaussian distribution from frame \( t \) to frame \( t+1 \) using recursive filtering technique:

\[
\mu_{it+1} = (1 - \alpha) \mu_{it} + \alpha \chi_{it} \tag{11}
\]

\[
\sigma_{it+1}^2 = (1 - \alpha) \sigma_{it}^2 + \alpha (\chi_{it} - \mu_{it})^2 \tag{12}
\]

The initial mean texture \( \mu_1 \) is constructed at the first frame, after we have recovered the shape deformation parameters. \( \alpha \) is set to 0.1 in our experiments. To find the optimal deformation parameters given a new frame, we minimize the Mahalanobis distance between the warped texture and the current appearance mean:

\[
\hat{u}_t = \arg \min_{u_t} \sum_{i=1}^{n} \left( \frac{\chi_{it} - \mu_{it}}{\sigma_{it}} \right)^2 \tag{13}
\]

This is a non-linear least-squares problem, and we apply the same LMA optimization as in ICP. The only difference is that we cannot compute the Jacobian matrix analytically, thus we use numerical approximations, similar to [9]. Moreover, in OAM the computation of the Jacobian matrix is expensive, we keep it unchanged during the optimization and only update it once in each frame after the deformation parameters are recovered.

### V. Experiments

For quantitative and qualitative results, we carry out experiments on synthetic RGBD sequences that we render from the BU-4DFE dataset. We also demonstrate the tracking performance on real RGBD sequences captured from the Microsoft\textsuperscript{TM} Kinect and Creative\textsuperscript{TM} Senz3D depth cameras.
Figure 5(c) demonstrates fine tracking performance on a Kinect sequence. Figure 6(c) shows the result on the Senz3D sequence; considering that both the point cloud and texture are noisy and slightly misaligned, the performance of the hybrid tracker is acceptable.

VI. Conclusion

In this paper, we presented a hybrid 3D tracking framework that can be used to simultaneously track head pose and facial actions from RGBD video sequences. The tracker can start automatically from an arbitrary face pose given a few available feature points. Our framework combined Iterative Closest Point and On-line Appearance Model which complement each other and make the tracker more stable and accurate. In future work, we plan to work on a 3D head pose estimator to initialize the tracker from any arbitrary pose, extend our framework with eyelids and irises tracking, and improve speed, robustness and performance of our tracker.

REFERENCES

Fig. 4. Tracking results on 4 sequences of 4 different expressions from 4 subjects. Odd and even columns are results of the hybrid tracker and the ICP tracker respectively. In sequence (a), eyebrows are pulled out of place (frame 40), and mouth deforms incorrectly (frame 85) with the ICP tracker; the hybrid tracker performs well. In sequence (b) which is relatively difficult, the hybrid tracker tracks the facial actions correctly. Frame 20 of sequence (c) shows really good deformation of the mouth area. In (d), the model returned by the ICP tracker drifts upward whereas the hybrid tracker performs reliably.

Fig. 5. Tracking results on a Kinect sequence. a) The model is drawn upon the first color frame. b) The model fits to the point cloud in the first frame. c) First and second rows show performance of the hybrid tracker and ICP tracker respectively. ICP tracker does not track the mouth deformation correctly.

Fig. 6. Tracking results on the Senz3D sequence. The fitting is skewed toward the right side of the face due to color/depth misalignment, but the tracker can still manage to model facial actions. a) and b) The fitting result in the first frame c) Performance of two trackers. ICP tracker fails to model the mouth correctly.

