Image and Vision Computing 32 (2014) 860–869

Contents lists available at ScienceDirect

Image and Vision Computing

journal homepage: www.elsevier.com/locate/imavis

Real-time 3D face tracking based on active appearance model constrained by depth data ☆

Nikolai Smolyanskiy *, Christian Huitema, Lin Liang, Sean Eron Anderson

Microsoft, Redmond, USA

ARTICLE INFO

Article history:
Received 5 February 2014
Received in revised form 31 May 2014
Accepted 4 August 2014
Available online 10 August 2014

Keywords:
Face tracking
Active Appearance Models
Morphable models
Fitting
Gradient descent
Kinect

ABSTRACT

Active Appearance Model (AAM) is an algorithm for fitting a generative model of object shape and appearance to an input image. AAM allows accurate, real-time tracking of human faces in 2D and can be extended to track faces in 3D by constraining its fitting with a linear 3D morphable model. Unfortunately, this AAM-based 3D tracking does not provide adequate accuracy and robustness, as we show in this paper. We introduce a new constraint into AAM fitting that uses depth data from a commodity RGBD camera (Kinect). This addition significantly reduces 3D tracking errors. We also describe how to initialize the 3D morphable face model used in our tracking algorithm by computing its face shape parameters of the user from a batch of tracked frames. The described face tracking algorithm is used in Microsoft's Kinect system.

1. Introduction

Many computer vision applications require accurate and real-time tracking of human faces. This includes gaming, teleconferencing, surveillance, facial recognition, emotion analysis, etc. A good face tracking system should track most human faces in a variety of lighting conditions, head poses, environments, and occlusions. A typical algorithm takes RGB, infrared, or depth frames as input and computes facial landmarks, head pose, facial expression parameters in 2D or 3D. Some of these parameters are correlated with each other and therefore are difficult to compute reliably. For example, errors in mouth size estimation contribute to inaccurate lip alignment. Real-time face tracking is challenging due to the high dimensionality of the input data and non-linearity of facial deformations.

Face tracking algorithms fall into two main classes. The first class consists of feature-based tracking algorithms, which track local interest points from frame to frame to compute head pose and facial expressions based on locations of these points [12,13,15,16]. Global regularization constraints are used to guarantee valid facial alignments. Local feature matching makes these algorithms less prone to generalization, illumination, and occlusion problems. On the down side, errors in interest point tracking lead to jittery and inaccurate results. The second class consists of algorithms that use appearance-based generative face models, such as Active Appearance Models (AAM) [1–3,5,7,8,10] and 3D morphable models [4]. These algorithms are more accurate and produce better global alignment since they compute over the input data space. However, they have difficulty generalizing to unseen faces and may suffer from illumination changes and occlusions.

Both classes of face tracking algorithms use generative linear 3D models [4] for 3D face alignment. They either fit a projected 3D model to video camera input [7,10] or combine projected model fitting with 3D fitting to depth camera input [12–14]. Xiao et al. [7] introduced 2D + 3D AAM, which uses a projected linear 3D model as a regularization constraint in 2D AAM fitting. Constrained 2D + 3D AAM produces only valid facial alignments and estimates 3D tracking parameters (head pose, expressions). It still suffers from typical AAM problems with generalization, illumination, and occlusions. Zhou et al. [10] reduced these problems by introducing a temporal matching constraint and a color-based face segmentation constraint in 2D + 3D AAM fitting. The temporal matching constraint improves generalization properties by enforcing inter-frame local appearance. The color-based face segmentation reduces AAM divergence over complex backgrounds. They also initialize AAM close to the final solution to improve its convergence. The resulting system is stable, tracks faces accurately in 2D, and has good generalization properties. The 3D alignment is still based on fitting a projected 3D model to input images. Model inaccuracies lead to significant errors in 3D tracking. In general, all monocular solutions suffer from the same problem. Several teams tried to overcome it by introducing depth-based tracking. Fua et al. [9] used stereo data, silhouette edges, and 2D feature points combined with least-squares adjustment of a set of progressively finer triangulations to compute head shapes. This required many parameters and therefore is computationally expensive. Others [12–14] used

☆ This paper has been recommended for acceptance by Stefanos Zafeiriou.

* Corresponding author at: 1112 4th ave N, Seattle, WA, 98109, USA. Tel.: + 1 206 383 8430.
E-mail addresses: nikolays@microsoft.com (N. Smolyanskiy), huitema@microsoft.com (C. Huitema), lliang@microsoft.com (L. Liang), seana@microsoft.com (S.E. Anderson).

http://dx.doi.org/10.1016/j.imavis.2014.08.005

depth data from commodity RGBD cameras to enable 3D tracking. These researchers used visual feature-based trackers (ASM, optical flow) to compute 2D alignment and combined it with the Iterative Closest Point (ICP) algorithm [11] to fit 3D face models to input depth data. The resulting systems track faces in 3D with high precision, but still may suffer from the shortcomings of feature-based trackers. Errors in interest point tracking may lead to jittery or unstable alignment. These systems may also require complex initialization of their 3D models, including manual steps to collect various face shapes [13].

In this work, we introduce *a new depth-based constraint into 2D + 3D AAM energy function to increase 3D tracking accuracy*. Our work extends the energy function described in the work of Zhou et al. [10]. We use the same set of terms, including: 2D AAM, 2D + 3D constraint, temporal matching constraint, face segmentation constraint. We also use optical feature tracking to initialize AAM fitting each frame close to the target, improving its convergence. These terms, combined with good initialization, improve AAM generalization properties, its robustness, and convergence speed. We improve 3D accuracy by introducing depth fitting to the energy function. We add a new constraint based on depth data from commodity RGBD camera (Kinect). This constraint is formulated similar to the energy function used in the Iterative Closest Point algorithm [11]. In addition, we replace the color-based face segmentation with the depth-based face segmentation and add an L2-regularization term. The resulting system is more accurate in 3D facial feature tracking and in head pose estimation than 2D + 3D AAM with additional constraints described in Zhou et al. [10].

We initialize our 3D face model by computing realistic face shapes from a set of input RGBD frames. This further improves tracking accuracy since the 3D model and its projection are closer to the input RGBD data. We use a linear 3D morphable model as a face shape model. It is computed by applying Principal Component Analysis (PCA) to a set of 3D human faces collected with help of a high-resolution stereo color rig. Initially, the face tracker uses an average face as the 3D face model. Our system computes a personalized face model once it collects enough tracking data. We use 2D facial landmarks along with corresponding depth frames for face shape computation. The shape modelling algorithm deforms the 3D model so that it fits to all the collected data. After computing a personalized 3D face shape model, we subsequently improve the face tracker accuracy by using that 3D face model in the tracking runtime.

2. AAM-based face tracking with depth fitting

2.1. Motivation

We observed that the face tracking algorithms based on 2D + 3D AAM described in Baker et al. [7] and Zhou et al. [10] are not accurate in 3D. The latter's system has good generalization properties and is robust in 2D video tracking, but is not accurate enough for real life applications in 3D tracking. Solutions are aligned with video data in 2D, but misaligned with range data in 3D. Fig. 1 shows a typical example of misalignment.

The 2D + 3D AAM energy minimization has many solutions when fitting 3D mask to a face. When a face moves from left to right, the 3D mask may move closer or further from the camera. These movements can be as large as the head size in 3D space. At the same time, the projection-based AAM energy terms have small residuals and the tracked face stays aligned in 2D. One can easily see that tracking an object in 3D with a monocular camera is an ill-defined optimization problem if the precise 3D shape of the object is not known. To resolve this, we introduced a new constraint into 2D + 3D AAM fitting that minimizes a distance between 3D face model vertices and depth data coming from a RGBD camera. The extended energy function provides more accurate solutions in 3D space. This addition also allows easier tracking of volumetric facial features such as cheek puffs, lip pucks, etc., which are not always visible on 2D video frames that lack depth.

We also considered building a tracker based on 3D AAM instead of 2D + 3D AAM. We abandoned this option due to the expensive training process. The 3D AAM training requires a large set of 3D facial expressions annotated with good precision. To cover all expressions, this task would involve collecting nearly ten times more data with our stereo capture rig. We would then need to compute all 3D head models with a time-consuming stereo-processing system and manually annotate these models in 3D with complex tools. Instead, we decided to capture only 500 neutral faces (no expressions) in 3D with our stereo capture rig and annotate them. We used Principal Component Analysis (PCA) to build a statistical linear 3D face shape model. Our 3D artist created a realistic set of facial expression deformations for this model. The 2D AAM model was trained from 500 face images with various expressions. The AAM was trained by using PCA for a set of annotated 2D images. This training process was less expensive than full 3D AAM training for real-world applications that require tracking many facial types.

2.2. 2D active appearance model and 3D morphable face model

The AAM 2D shape and appearance models are defined by the following 2D shape and grayscale image appearance linear equations:

$$s = s_0 + \sum_{i=1}^{n} p_i s_i \tag{1}$$

$$A(u) = A_0(u) + \sum_{i=1}^{g} \lambda_i A_i(u) \tag{2}$$

Fig. 1. A 3D face mask computed by tracking a face with 2D + 3D AAM extended as in Zhou et al. [10] and a precise point cloud computed by an independent stereo vision system. The projected mask is aligned with the projected point cloud (on the left), but they are misaligned in 3D space (view from above, on the right).

where s_0,A_0 are respectively the mean face shape (vector of 2D vertices) and mean appearance (as a grayscale image); u is a pixel location in the mean shape coordinate frame; s_i,A_i are shape and appearance basis vectors; and p_i,λ_i are shape and appearance parameters. Mean shape, mean appearance, shape and appearance basis vectors are computed from the training set of annotated images by the PCA process. The basis vectors are the first Eigen vectors corresponding to the largest eigenvalues of the shape or appearance data matrix. Appearance basis vectors are Eigen faces. See Matthews and Baker [5] or Zhou et al. [10] for more details.

The 3D linear face shape model is defined similarly to Zhou et al. [10] as:

$$\bar{s}_{3D} = \bar{s}_0 + \sum_{i=1}^{L} SU_i \bar{s}_i + \sum_{i=1}^{M} AU_i \bar{a}_i \tag{3}$$

$$S_{3D} = sR\bar{s}_{3D} + T \tag{4}$$

where $\bar{s}_{3D}$ is a 3D face shape in the model space and S_{3D} is a 3D model in the 3D camera space; $\bar{s}_0$ is a mean 3D shape; $\bar{s}_i, \bar{a}_i$ are the shape and animation deformation vectors that form a basis; $SU = \{SU_i\}_{i=1}^{L}$, $AU = \{AU_i\}_{i=1}^{M}$ are the shape and animation deformation parameters; s,R,T are the scale, 3D rotation matrix, and translation vector. Rotation matrix depends on θ_x,θ_y,θ_z—Euler rotation angles around X, Y, Z axes. Translation vector consists of T_x,T_y,T_z—translations in X, Y, and Z. Our 3D model has explicit face shape $(\bar{s}_i, SU_i)$ and facial animation $(\bar{a}_i, AU_i)$ parts to simplify creation of avatar animation systems. The 3D mean face topology was created by a 3D artist and includes 1347 vertices. The mean face vertices and shape basis were computed by the PCA process; the shape basis consists of 110 shape deformations. The animation basis was created by a 3D artist so that it is easier to use in face animation applications; it consists of 17 animations. The resulting animation basis is transformed to form an orthonormal basis together with the shape basis.

2.3. Energy function extended with depth term

Our goal is to improve 3D fitting accuracy of 2D + 3D AAM by using depth data from an RGBD camera like Kinect. To accomplish this, we introduce *a new depth-based term* to the 2D + 3D AAM energy function. The new term minimizes 3D Euclidean distances between face model vertices and closest depth points. We formulate this term similar to the energy function used in Iterative Closest Point (ICP) algorithm. This addition reduces the solution space and finds better 3D alignments.

We define our energy function as an extension of the energy function used in Zhou et al. [10] (which extends basic 2D + 3D AAM energy described in Xiao et al. [7]):

$$E = w_{2D}E_{2D} + w_{2D3D}E_{2D3D} + w_{depth}E_{depth} + w_{temp}E_{temp} + w_{fseg}E_{fseg} + w_{reg}E_{reg} \tag{5}$$

where E_{2D},E_{2D3D},E_{depth},E_{temp},E_{fseg},E_{reg} are L2 terms and w_{2D},w_{2D3D},w_{depth}, w_{temp},w_{fseg},w_{reg} are the corresponding scalar weights. E_{2D} is the 2D AAM term responsible for aligning facial features to input video frames. We formulate it as in the prior art [5,7,10]. E_{2D3D} minimizes distances between vertices of 2D AAM shape and projected 3D face model. E_{2D3D} is formulated as in the 2D + 3D AAM paper [7]. $\boldsymbol{E_{depth}}$ *is the new term we introduce*. It minimizes the 3D Euclidean distances between face model vertices and the closest depth points to improve 3D fitting accuracy. E_{temp},E_{fseg} are temporal matching and face segmentation constraints. They improve AAM's generalization and robustness properties (see Zhou et al. [10] for details). We also add a simple L2 regularization term E_{reg} that minimizes changes in model parameters. Weights w_{2D},w_{2D3D},w_{depth},w_{temp},w_{fseg},w_{reg} are the scalar weights that are chosen such that they normalize corresponding terms numerically and weigh them based on the quality of incoming data.

In general, we may follow a Bayesian framework to derive formula (5) and set weights to be the inverse covariance of the corresponding term's residuals. This approach is not always practical due to complexities in estimating residuals and the size of the residual vectors. So to simplify, we assume that all *measurement channels are independent* and that the noise is *isotropic*. We also set the weight of the 2D AAM term to 1.0 for computational convenience. Under these assumptions, other weights are set to be the variances of their term residuals divided by the 2D AAM term's variance.

Fig. 2 shows the processing flow in our face tracking system. We initialize our face tracking system by finding a face rectangle in a video frame using a face detector. Then we use a neural network to find five points inside the face area—eye centers, mouth corners, and tip of the nose. We pre-compute the scale of the tracked face from these five points un-projected to 3D camera space and scale our 3D face model appropriately. When depth data is unavailable (some depth frames may have no data around face areas), we assume the user's head scale to match the mean face size. We also align a 2D AAM shape to these five feature points. This improves initial convergence when we minimize formula (5). We also follow Zhou et al. [10] in terms of tracking initialization—we initialize the next frame's 2D face shape based on the correspondences found by a robust local feature

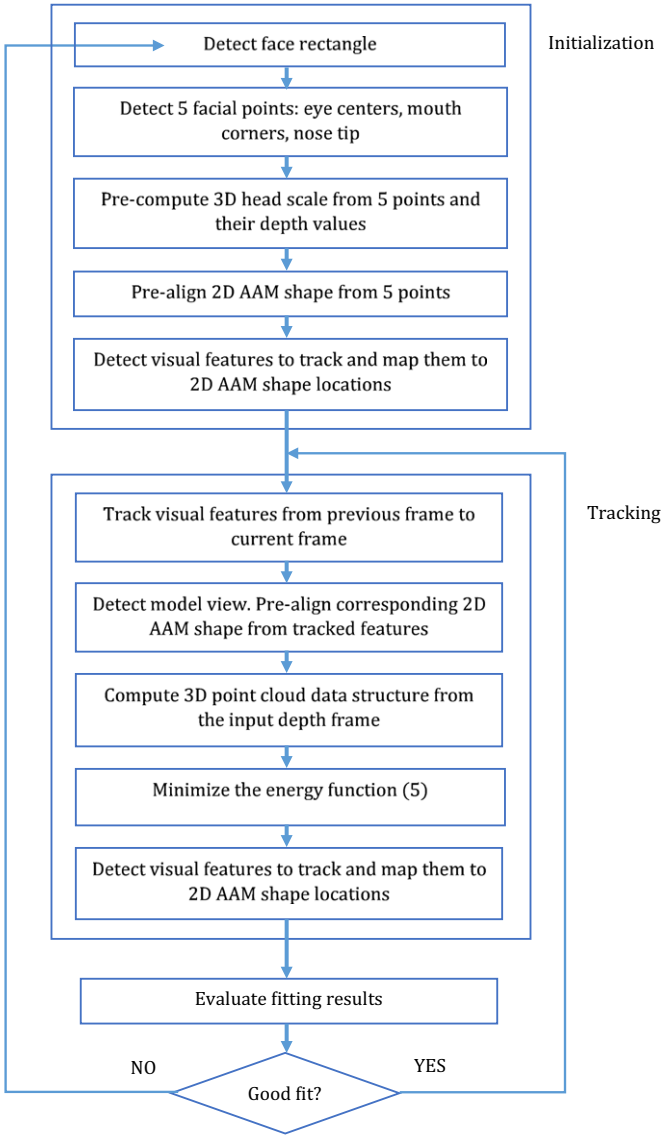

Fig. 2. The flowchart of our face tracking system.

matching between that frame and the previous frame. This improves the stability in tracking fast face motions and reduces number of Gauss–Newton iterations required to minimize formula (5).

To handle large angles of rotation, we use a view-based approach [17] and train three 2D AAMs for different view ranges—frontal, left and right. We switch to the left and right models from the frontal model when the angle between the line connecting the bridge of the user's nose and the camera reaches $\pm 75°$.

2.4. Video data based energy terms

Next we briefly describe the formulation of the video data-based terms E_{2D},E_{2D3D},E_{temp},E_{fseg}. They are described elsewhere [5,7,10] in full detail. We subsequently formulate and derive our new depth data-based term E_{depth}, and provide information about regularization term E_{reg}.

E_{2D} is defined as in 2D AAM with inverse compositional image alignment [5,6,10]. Formulas (1) and (2) define the shape and appearance models. The 2D AAM energy term is defined as:

$$E_{2D} = \sum_{u} \left[A_0(N(W(u;\Delta p);\Delta q)) - I(N(W(u;p);q)) \right]^2_{span(A_i)} \quad (6)$$

where $span(A_i)$ is a projection from the full parameter space (p,q,λ) onto the subspace orthogonal to the span of the appearance bases. We use this projection to simplify and speed up the optimization—we first optimize for p,q and then compute λ (see Matthews and Baker [5] for details). We instantiate AAM with help of two 2D transforms: 1) a piecewise affine warp $W(u;p)$ which transforms a pixel location in the mean shape coordinate frame according to given shape parameters p; 2) $N(u;q)$ global normalizing 2D transform that maps a pixel location from the mean shape coordinate frame to the image coordinate frame. The 2D face model (1,2) is instantiated from a set of parameters (p,q,λ) by generating appearance A in the mean shape coordinate frame and then warping it to the image frame with a piecewise affine warp composed with global normalizing transform $N(W(u;p);q)$. We use the inverse compositional image alignment and therefore optimization parameters Δp,Δq define the inverse delta warp $N(W(x;\Delta p);\Delta q)$ in the mean shape coordinate frame. This warp needs to be properly composed with the known $N(W(x;p);q)$ warp defined in the absolute image coordinate frame with currently known warp parameters p,q. See Matthews and Baker [5] for more details on how to compose these warps to get updated p, q parameters. To make AAM more robust, we use four metrics as pixel feature descriptors: pixel grayscale intensity, grayscale gradient in X, grayscale gradient in Y, and corner/edge metric. See Scott et al.[3] or Zhou et al. [10] for further details.

Term E_{2D3D} links 2D AAM and 3D face models together. We formulate it as in Xiao et al. [7]:

$$E_{2D3D} = \left\| S_{2D}(p,q) - P\left(S_{3D}\left(\theta_x,\theta_y,\theta_z,T_x,T_y,T_z,AU\right)\right)\right\|_2^2 \quad (7)$$

where $S_{2D}(p,q)$ is a vector with 2D vertices of the 2D AAM mean shape s_0 (1) deformed by warp $N(W(x;p);q)$; and $S_{3D}(\theta_x,\theta_y,\theta_z,T_x,T_y,T_z,AU)$ is a vector with 3D face shape vertices defined by formulas (3 and 4). $P(S_{3D})$ is a projection function that can be defined as in a pinhole camera model:

$$P(x,y,z) = \begin{cases} \frac{fx}{z} + \frac{frame\ \ width}{2} \\ -\frac{fy}{z} + \frac{frame\ \ height}{2} \end{cases} \quad (8)$$

In a real production system, we use the camera model that includes radial distortions and other camera intrinsic parameters (see as in Hartley and Zisserman [18]. We freeze all 3D shape parameters SU in formula (3) and scale in formula (4) at the tracking time, so we achieve more accurate animation results. Face shape and scale parameters are computed by the face shape modelling algorithm when it collects enough tracking data for its batched computation (see Section 2.7 below).

E_{temp} is a temporal matching constraint that improves AAM's generalization. We formulate it as in Zhou et al. [10], and further details maybe found there.

$$E_{temp} = \sum_{j\in\Omega_t} \sum_{x\in R^j} \left[A_{t-1}(x)/\bar{g}^j_{t-1} - I_t(W(x;p_t))/\bar{g}^j_t(p_t) \right]^2 \quad (9)$$

where Ω_t is a set of feature points, including some interesting points selected by a corner detector and some semantic points, such as the eyes' corners. A_{t-1} is the face appearance of frame $t-1$ in the 2D model coordinate frame, R^j is the local patch corresponding to the j^{th} feature point. We set R^j size to 9x9. $\bar{g}^j_{t-1}$ and $\bar{g}^j_t(p_t)$ are the average intensity of the j^{th} patches of frame $t-1$ and t respectively. They are used to normalize the illuminations of the two patches.E_{fseg} is a face segmentation constraint. We define it similarly to Zhou et al. [10] with one exception—we use depth data to segment a human face instead of color data. Depth-based segmentation with Kinect is easy and reliable. We convert a segmented head mask into a cost map I_D. In the map, the pixel values are set to zero inside the face region and increase with distance from the boundary. This constraint forces the face outline points to stay inside the segmented face region. Mathematically, we define face segmentation constraint as:

$$E_{fseg} = \sum_{k=1}^{K} I_D(W(x_k;p))^2 \quad (10)$$

where $\{x_k\}$ are the locations for the face outline points in the 2D model coordinate frame.

2.5. Depth term and 3D regularization term

In this section, we introduce *our new depth constraint* to improve 3D accuracy of 2D + 3D AAM. The constraint is similar to the energy function used in the Iterative Closest Point (ICP) algorithm. We formulate it as:

$$E_{depth} = \left\| D - S_{3D}\left(\theta_x,\theta_y,\theta_z,T_x,T_y,T_z,AU\right)\right\|_2^2$$

where S_{3D} is a column vector with x,y,z coordinates of 3D face model vertices and D is a column vector with x,y,z coordinates of corresponding nearest depth points. The 3D face model is defined by formulas (3) and (4). S_{3D} contains only model vertices that are currently visible from the depth camera perspective (not occluded). The depth points are computed from a current input depth frame by un-projecting each depth pixel to the 3D camera space (in Kinect, it is aligned with depth camera space). Each 3D point in D is selected such that it is a nearest point to a corresponding 3D vertex in S_{3D} (with the same index).

Computing D in formula (11) is a relatively expensive operation. First, we need to compute a 3D point cloud from the input depth frame and then find nearest points for all non-occluded vertices in S_{3D}. Nearest point search is an expensive operation for real-time systems even when done with the help of KD-trees [11]. So to make this operation faster, we utilize a property of the depth frames—they are organized as a grid of pixels since depth values are computed for each pixel projected to IR camera (in case of Kinect). Therefore, even when a pixel is un-projected to 3D camera space, we know its index in the depth frame grid. We use this property for fast nearest-point lookup based on uniform sampling.

The algorithm works as follows:

1) The depth frame is organized as a 2D array with rows and columns. For each element in this array, un-project corresponding depth pixel (by using its 2D coordinates and depth value) to 3D camera space and then convert the resulting 3D point to the world space where 3D tracking occurs (could be either depth camera or RGB camera space). The resulting 3D point cloud is organized as a 2D array.
2) Split the array into several uniformly distributed square cells with several array elements in each. Compute an average 3D point for each cell and save them as "sampling points".

To find the 3D point nearest to a 3D model vertex (during optimization):

1) Find the nearest "sampling point" to a given 3D vertex (linear search)
2) Find the nearest 3D depth point to this 3D vertex by searching in the cell that corresponds to the found nearest "sampling point" (linear search)

This algorithm can also utilize a pyramid search to further improve its speed.

We also add a simple 3D parameter regularization term to the energy function (5) to improve its fitting robustness:

$$E_{reg} = \|\Delta\Theta_{3D}\|_2^2 \tag{12}$$

where $\Theta_{3D} = [\theta_x,\theta_y,\theta_z,T_x,T_y,T_z,AU]^T$ is a vector of 3D face model parameters and $\Delta\Theta_{3D}$ is its change from the last frame.

2.6. Fitting with depth constraint

We now describe how to fit 2D + 3D AAM extended with the new depth constraint. The goal of fitting is to minimize the energy function (5) simultaneously with respect to 2D AAM and 3D face model parameters. We can combine 2D and 3D parameters in one vector:

$$\Theta = [\Theta_{2D},\Theta_{3D}];\Theta_{2D} = [\Delta p,\Delta q]^T;\Theta_{3D} = \left[\theta_x,\theta_y,\theta_z,T_x,T_y,T_z,AU\right]^T \tag{13}$$

To find the best position of the 2D features and 3D mask that fits the input RGBD data, we need to find the value of Θ that minimizes the non-linear function (5). We employ a Gauss–Newton method to find the optimum parameters. We perform a first order Taylor expansion of each term in formula (5) and then minimize the expanded energy function to find parameter updates $\Delta\Theta$. The parameter updates are found by computing Jacobians and residuals for each term, deriving a set of normal equations, and solving them to find parameter update.

The derivations for 2D AAM term E_{2D} are given in Matthews and Baker [5]. We use an inverse compositional image alignment algorithm as in [5] to improve the speed of computing the 2D AAM term E_{2D}. $\frac{\partial N \circ W}{\partial \Theta_{2D}}$ is a derivative of the combined shape and global warps at point $p = 0;q = 0$. Both of these derivatives are constant and are pre-computed at 2D AAM training time. We use the approximation $\frac{\partial S_{2D}^{\dagger}}{\partial \Theta_{2D}}$ instead of $\frac{\partial S_{2D}}{\partial \Theta_{2D}}$ since S_{2D} depends on p,q parameters defined in the image coordinate space, but we optimize for their inverse compositional update, $\Delta p,\Delta q$, defined in the mean shape coordinate space. We use $S_{2D}^{\dagger}$ from equation (60) in Matthews and Baker [5] to approximate S_{2D}.

Taylor expansion, computation of Jacobians, and computation of residuals for terms E_{2D3D},E_{temp},E_{fseg} is well described in by Xiao et al. [7] and Zhou et al. [10] and is trivial for term E_{reg}, so we omit their derivation to focus on the new depth term E_{depth}. A first order Taylor expansion of E_{depth} gives:

$$E_{depth} = \left\| D - \left(S_{3D} + \frac{\partial S_{3D}}{\partial \Theta_{3D}} \Delta\Theta_{3D} \right) \right\|_2^2 \tag{14}$$

where D is a vector with x,y,z coordinates of the nearest 3D depth points as in (11); S_{3D} is a vector with corresponding x,y,z coordinates of currently known 3D model vertices; $\Delta\Theta_{3D}$ is a vector of 3D parameter updates. $\frac{\partial S_{3D}}{\partial \Theta_{3D}}$ is a 3D model Jacobian formulated as:

$$\frac{\partial S_{3Di}}{\partial \Theta_{3D}} = \left[\frac{\partial S_{3D_i}}{\partial \theta_x} \;\; \frac{\partial S_{3D_i}}{\partial \theta_y} \;\; \frac{\partial S_{3Di}}{\partial \theta_z} \;\; \frac{\partial S_{3D_i}}{\partial T_x} \;\; \frac{\partial S_{3D_i}}{\partial T_y} \;\; \frac{\partial S_{3D_i}}{\partial T_z} \;\; \frac{\partial S_{3D_i}}{\partial AU_1} \;\; \cdots \;\; \frac{\partial S_{3D_i}}{\partial AU_M} \right]$$
$$\frac{\partial S_{3D_i}}{\partial \theta_x} = s\frac{\partial R}{\partial \theta_x}\bar{s}_{3D_i};\frac{\partial S_{3D_i}}{\partial \theta_y} = s\frac{\partial R}{\partial \theta_y}\bar{s}_{3D_i};\frac{\partial S_{3Di}}{\partial \theta_z} = s\frac{\partial R}{\partial \theta_z}\bar{s}_{3D_i}$$
$$\frac{\partial S_{3D_i}}{\partial T_x} = \begin{bmatrix}1\\0\\0\end{bmatrix};\frac{\partial S_{3D_i}}{\partial T_y} = \begin{bmatrix}0\\1\\0\end{bmatrix};\frac{\partial S_{3D_i}}{\partial T_z} = \begin{bmatrix}0\\0\\1\end{bmatrix}$$
$$\frac{\partial S_{3D_i}}{\partial AU_j} = sR\bar{a}_{i,j} \tag{15}$$

where $\frac{\partial R}{\partial \theta_x},\frac{\partial R}{\partial \theta_y},\frac{\partial R}{\partial \theta_z}$ are the rotation matrix derivatives with regard to Euler rotation angles; $\bar{s}_{3D_i}$ is the i^{th} vertex of the 3D model in the model space defined by (3); $\bar{a}_{i,j}$ is the i^{th} deformation vector of a j^{th} AU basis vector that is applied to S_{3D_i}.

Once we know all Jacobians and residuals for expanded energy function (5), we set its derivative $\frac{dE}{d\Theta}$ as zero and find a set of normal equations. The equation for parameter update $\Delta\Theta$ then becomes:

$$\Delta\Theta = -H^{-1}\left(w_{2D}\mathrm{J}_{2D}{}^T r_{2D} + w_{2D3D}\mathrm{J}_{2D3D}{}^T r_{2D3D} + w_{temp}\mathrm{J}_{temp}{}^T r_{temp} + w_{fseg}\mathrm{J}_{fseg}{}^T r_{fseg} + w_{depth}\mathrm{J}_{depth}{}^T r_{depth}\right) \tag{16}$$

where J_{term} and r_{term} are the Jacobian matrix and residual vector for a corresponding term. H is a combined Hessian defined as a sum of Hessians for each energy term:

$$H = w_{2D}\mathrm{J}_{2D}{}^T\mathrm{J}_{2D} + w_{2D3D}\mathrm{J}_{2D3D}{}^T\mathrm{J}_{2D3D} + w_{temp}\mathrm{J}_{temp}{}^T\mathrm{J}_{temp} + w_{fseg}\mathrm{J}_{fseg}{}^T\mathrm{J}_{fseg} + w_{depth}\mathrm{J}_{depth}{}^T\mathrm{J}_{depth} + w_{reg}\mathrm{I} \tag{17}$$

The energy function is non-linear, so we iterate this process until convergence, updating Θ with each iteration. The 2D parameter updates $\Delta p,\Delta q$ defined in the mean shape coordinate frame are used to update p, q defined in the image frame by the warp composition as described in Matthews and Baker [5].

2.7. 3D face model initialization

Our face tracking algorithm fits the 2D AAM (1, 2) and the linear 3D face model (3, 4) to compute 2D facial landmarks, 3D head pose, and 3D animation coefficients $\{AU_i\}$. We assume the head scale in formula (4) and the face shape parameters $\{SU_i\}$ in formula (3) to be known and constant for a given user during tracking time. In this section, we describe how to compute these shape parameters and initialize our 3D model.

We start tracking a new face with the 3D model (3) initialized to the mean face (scale is set to 1.0 and $\{SU_i\}$ are set to 0). The system then collects tracking results and RGBD input data for successfully tracked frames. We ask the user to only exhibit neutral emotionless face during this time. We do this to reduce the influence of expressions on the face shape computation. For example, expressions like big smiles may influence the geometry of the mouth region. We also ask the user to look straight at the camera, to rotate their head left and right by 30° and to look up and down by 15°. This process is needed to gather more diverse data and eliminate occlusions.

Once our system accumulates enough frames, we run a batch-based algorithm to compute 3D face model shapes. We found that 16 frames are enough to provide good photo-realistic quality with an Xbox One Kinect camera. We only collect frames in which the fitting error is less

than a predefined threshold. Our shape modelling algorithm is similar to the model initialization procedure described by Cai et al. [12]. The main difference is in 2D feature points—we use 2D face shapes aligned by our 2D + 3D AAM and Cai et al. [12] uses points computed by a different feature detector (Active Shape Model). We also use depth frames in addition to 2D facial points. When the face shape modelling finishes, we update the 3D model used in formula (5) with the newly computed parameters to further improve face tracking accuracy.

We formulate the 3D shape computation as an energy minimization problem. We run the optimization over *a batch of Q frames* to ensure a good variety of poses and to filter out temporal camera noise present in the input depth data. The optimization computes face shape parameters that explain the user's face geometry on all collected frames. The energy function is defined as:

$$E = w_{2D3D}\left\|S_{2Dbatch} - P(S_{3Dbatch}(\Theta_{batch}))\right\|_2^2 + w_{depth}\left\|D_{batch} - S_{3Dbatch}(\Theta_{batch})\right\|_2^2 + w_{reg}\sum_{i=1}^{L}\frac{SU_i^2}{\sigma_{SU_i}^2} \quad (18)$$

where Θ_{batch} is the combined set of parameters defined as a vector:

$$\Theta_{batch} = \left[{\Theta_{3D}}^1, \ldots, {\Theta_{3D}}^Q, \Theta_{shape}\right]^T \quad (19)$$

${\Theta_{3D}}^i$ is a vector with 3D pose and facial expression parameters $\{AU_i\}$ for i^{th} frame:

$${\Theta_{3D}}^i = \left[{\theta_x}^i, {\theta_y}^i, {\theta_z}^i, {T_x}^i, {T_y}^i, {T_z}^i, AU_1^i, \ldots, AU_M^i\right]^T \quad (20)$$

${\Theta_{3D}}^i$ is initialized to values computed by our face tracking algorithm. Θ_{shape} is a vector with the head scale and shape parameters $\{SU_i\}$. These parameters are common to all frames:

$$\Theta_{shape} = [s, SU_1, \ldots, SU_L]^T \quad (21)$$

see (3) for SU and AU parameter definition. $S_{2Dbatch}$ is a vector with 2D coordinates of facial landmarks found by our face tracking algorithm for all frames in the batch:

$$S_{2Dbatch} = \left\{{S_{2D}}^i\right\}_{i=1}^{Q} \quad (22)$$

${S_{2D}}^i$ points are constant during 3D shape computation. D_{batch} is a vector with x,y,z coordinates of 3D points nearest to corresponding 3D model vertices:

$$D_{batch} = \left\{D^i\right\}_{i=1}^{Q} \quad (23)$$

D^i is a vector of the i^{th} frame 3D depth points selected such that they are nearest to the corresponding 3D model vertices (with the same indexes). We store the full point cloud for the face region and recompute D_{batch} during the shape computation iterations because the old values become invalid as the 3D model changes in shape and position. The nearest depth-based 3D points are found the same way as in the face tracking algorithm (see Section 2.5). We introduce $S_{3Dbatch}$ as a vector of 3D model vertices for all frames:

$$S_{3Dbatch} = \left\{{S_{3D}}^i\left({\Theta_{3D}}^i, \Theta_{shape}\right)\right\}_{i=1}^{Q} \quad (24)$$

Here, ${S_{3D}}^i\left({\Theta_{3D}}^i, \Theta_{shape}\right)$ is a vector of 3D model vertices for the i^{th} frame. It is defined as in formulas (3, 4) and depends on the i^{th} frame pose and animation parameters ${\Theta_{3D}}^i$ and common 3D shape parameters Θ_{shape}. $P(S_{3Dbatch}(\Theta_{batch}))$ is a vector containing 3D shapes from $S_{3Dbatch}$ projected to the RGB image frame (below, we use P() to indicate a projection function). The term's weights w_{2D3D},w_{depth} are set to be the variances of their term residuals divided by the 2D term variance (in this case w_{2D3D} is equal to 1.0).

The last term in the energy function is a regularization term. It provides numerical stability and pushes the fitting process to produce more realistic face shapes. The values $\left\{\sigma_{SU_i}^2\right\}$ are set to be the Eigenvalues of the PCA covariance matrix, which we compute during 3D face model training. Each $\sigma_{SU_i}^2$ corresponds to its shape deformation SU_i produced by PCA. We do 3D model training over a set of high precision 3D ground truth face masks produced by our stereo rig. We first do a non-parametric fit of our 3D model to each ground truth mask, then we align all fitted masks, and finally perform a PCA over the resulting set. We also tried a simpler L2-regularization term $\|\Delta\Theta_{shape}\|_2^2$, but found that the computed face shapes looked generic and similar to the mean face.

We minimize (18) by first doing the first order Taylor expansion. It gives:

$$E = w_{2D3D}\left\|S_{2Dbatch} - P(S_{3Dbatch}(\Theta_{batch})) - J_{2D3D}\Delta\Theta_{batch}\right\|_2^2 + w_{depth}\left\|D_{batch} - S_{3Dbatch}(\Theta_{batch}) - J_{depth}\Delta\Theta_{batch}\right\|_2^2 + w_{reg}\sum_{i=1}^{L}\frac{\left(SU_i^{old} + \Delta SU_i\right)^2}{\sigma_{SU_i}^2} \quad (25)$$

where Jacobian matrices are defined as:

$$J_{2D3D} = \begin{bmatrix} \frac{\partial P\left({S_{3D}}^1\right)}{\partial {\Theta_{3D}}^1} & 0 & \cdots & 0 & \frac{\partial P\left({S_{3D}}^1\right)}{\partial \Theta_{shape}} \\ 0 & \frac{\partial P\left({S_{3D}}^2\right)}{\partial {\Theta_{3D}}^2} & \cdots & 0 & \frac{\partial P\left({S_{3D}}^2\right)}{\partial \Theta_{shape}} \\ 0 & 0 & \ddots & 0 & \vdots \\ \vdots & \vdots & 0 & \frac{\partial P\left({S_{3D}}^Q\right)}{\partial {\Theta_{3D}}^Q} & \frac{\partial P\left({S_{3D}}^Q\right)}{\partial {\Theta_{shape}}^Q} \\ 0 & 0 & 0 & & \end{bmatrix} \quad (26)$$

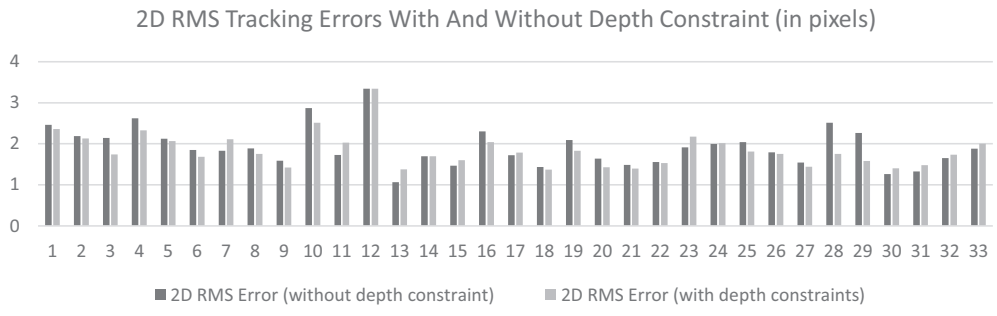

Fig. 3. Comparison of the 2D alignment RMS errors for our face tracking system with disabled depth constraint (left bars) and with enabled depth constraint (right bars) for 33 videos.

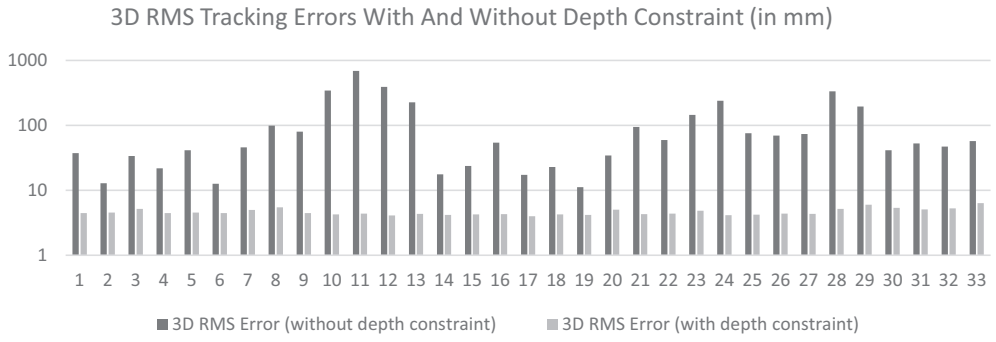

Fig. 4. Comparison of the 3D RMS errors for our face tracking system with disabled depth constraint (left bars) and with enabled depth constraint (right bars) for 33 videos.

$$J_{depth} = \begin{bmatrix} \frac{\partial S_{3D}{}^{1}}{\partial \Theta_{3D}{}^{1}} & 0 & \cdots & 0 & \frac{\partial S_{3D}{}^{1}}{\partial \Theta_{shape}} \\ 0 & \frac{\partial S_{3D}{}^{2}}{\partial \Theta_{3D}{}^{2}} & 0 \cdots & 0 & \frac{\partial S_{3D}{}^{2}}{\partial \Theta_{shape}} \\ 0 & 0 & \ddots & 0 & \vdots \\ \vdots & \vdots & 0 & \frac{\partial S_{3D}{}^{Q}}{\partial \Theta_{3D}{}^{Q}} & \frac{\partial S_{3D}{}^{Q}}{\partial \Theta_{shape}} \\ 0 & 0 & 0 & & \end{bmatrix} \quad (27)$$

$$\frac{\partial P\left(S_{3D}{}^{i}\right)}{\partial \Theta_{shape}} = \left[\frac{\partial P\left(S_{3D1}{}^{i}\right)}{\partial S_{3D1}{}^{i}} \frac{\partial S_{3D1}{}^{i}}{\partial \Theta_{shape}} \quad \cdots \quad \frac{\partial P\left(S_{3DN}{}^{i}\right)}{\partial S_{3DN}{}^{i}} \frac{\partial S_{3DN}{}^{i}}{\partial \Theta_{shape}} \right]^{T}$$
$$\frac{\partial P\left(S_{3D}{}^{i}\right)}{\partial \Theta_{shape}} = \left[\frac{\partial P\left(S_{3D1}{}^{i}\right)}{\partial S_{3D1}{}^{i}} \frac{\partial S_{3D1}{}^{i}}{\partial \Theta_{shape}} \quad \cdots \quad \frac{\partial P\left(S_{3DN}{}^{i}\right)}{\partial S_{3DN}{}^{i}} \frac{\partial S_{3DN}{}^{i}}{\partial \Theta_{shape}} \right]^{T} \quad (28)$$

where N is the number of 3D model vertices. $\frac{\partial P\left(S_{3D\,j}{}^{i}\right)}{\partial S_{3D\,j}{}^{i}}$ is the derivative of the projection function with regard to the j^{th} 3D model vertex for the i^{th} frame. $\frac{\partial S_{3D\,j}{}^{i}}{\partial \Theta_{3D}{}^{i}}$ is the derivative of the j^{th} 3D model vertex for the i^{th} frame with regard to 3D pose and animation parameters defined as in formula (15). $\frac{\partial S_{3D\,j}{}^{i}}{\partial \Theta_{shape}}$ is the derivative of the j^{th} 3D model vertex for the i^{th} frame with regard to 3D shape parameters defined as:

$$\frac{\partial S_{3D\,j}{}^{i}}{\partial \Theta_{shape}} = \left[R\bar{s}_{3D\,j}{}^{i} \quad sR\bar{s}_{1,j} \quad \cdots \quad sR\bar{s}_{l,j} \right] \quad (29)$$

$\bar{s}_{3D\,j}{}^{i}$ is the j^{th} vertex of the 3D shape for the i^{th} frame in the model space. $\bar{s}_{k,j}$ is the j^{th} deformation vector of a k^{th} 3D shape basis vector that is applied to $S_{3D\,j}{}^{i}$.

We set the derivative of $\frac{dE}{d\Theta_{batch}}$ to zero and derive the normal equations for the parameter update:

$$\Delta\Theta_{batch} = H^{-1}\left(w_{2D3D}\mathrm{J}_{2D3D}{}^{T} r_{2D3D} + w_{depth}\mathrm{J}_{depth}{}^{T} r_{depth} - w_{reg} Z \Theta_{batch}{}^{old}\right) \quad (30)$$

where combined Hessian and residuals are defined as:

$$H = w_{2D3D}\mathrm{J}_{2D3D}{}^{T}\mathrm{J}_{2D3D} + w_{depth}\mathrm{J}_{depth}{}^{T}\mathrm{J}_{depth} + w_{reg} Z \quad (31)$$

$$r_{2D3D} = S_{2Dbatch} - P(S_{3Dbatch}(\Theta_{batch})); r_{depth} = D_{batch} - S_{3Dbatch}(\Theta_{batch}) \quad (32)$$

Matrix Z is a square matrix with row and column count equal to the row count of Θ_{batch} parameter vector. All elements of Z are set to 0, except for diagonal elements that correspond to shape parameters $\{SU_i\}$; those elements are set to $\frac{1}{\sigma^2_{SU_i}}$.

The energy function is non-linear, so we iterate this process until convergence, updating Θ_{batch} with each step. The optimization finds head pose and animation $\left\{\Theta_{3D}{}^{i}\right\}$ parameters (see formula (20)) for each frame and Θ_{shape} shape parameters (see formula (21)) that are common across all frames. After convergence, we update the 3D face model in the face tracking runtime with computed Θ_{shape} parameters. This improves face tracking accuracy since our 3D model matches the observed face more accurately.

We use the Levenberg–Marquardt algorithm in our production system. It is more robust than Gauss-Newton for the face modelling task. We also exclude depth points that may be located in the occluded face areas (for example by hair). Occluded regions may lead to wrongly computed shapes. We use a skin detector to find skin pixels in an input RGB frame. The face shape computation uses only depth points that correspond to skin pixels.

3. Experimental results

We tested our face tracking algorithm by using annotated videos of people exhibiting facial movements in front of Kinect camera at a distance of 1 to 2.5 m. The Kinect system has a pair of cameras: a color video camera with 1920 × 1080 resolution and an infrared/depth camera with 512 × 424 resolution. These cameras are calibrated to provide registration between video and depth frames. We recorded people of different genders, ages, and ethnicities. Each video sequence

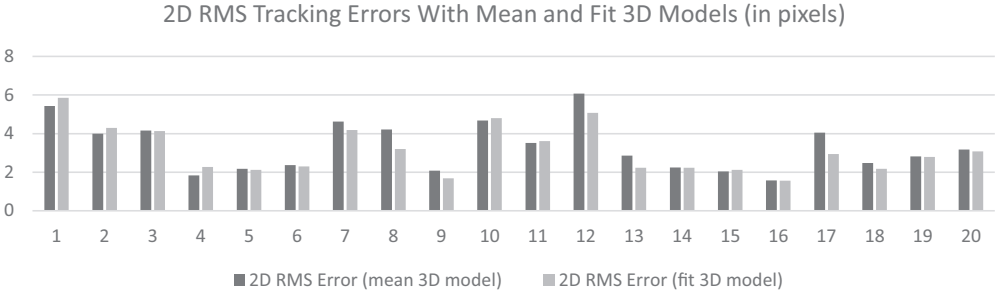

Fig. 5. Comparison of the 2D alignment RMS errors for our face tracking system with mean 3D model (left bars) and with fit 3D model (right bars) used in tracking process for 20 videos.

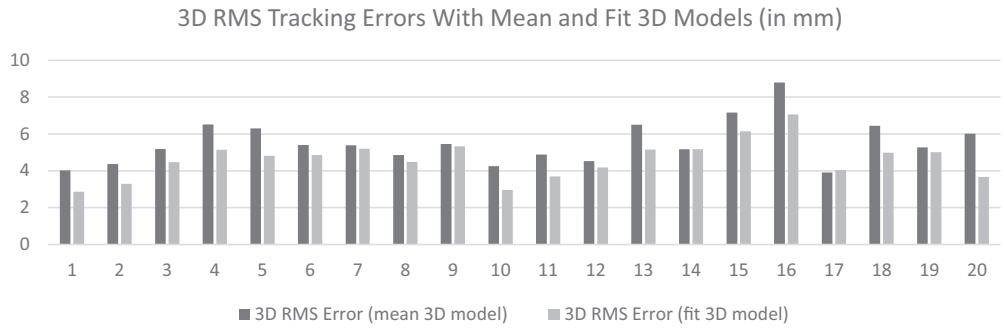

Fig. 6. Comparison of the 3D RMS errors for our face tracking system with mean 3D model (left bars) and with fit 3D model (right bars) used in tracking process for 20 videos.

starts with the user showing no emotions while they move their heads left/right and up/down, so our face shape fitting algorithm can compute their personalized 3D face model. Then subjects show a wide variety of facial expressions and movements like smiles, frowning, eye brow movements, jaw movements, pucks, and kisses. The videos were annotated by marking 2D facial landmarks on RGB frames by hand.

In addition to Kinect recordings, we used 12 high-definition DSLR color cameras to capture 3D ground truth for the test subjects exhibiting neutral faces. The DSLR cameras were positioned in a hemisphere formation in front of the user's face. We used commercially available software to stereoscopically build highly detailed 3D ground truth models from the captured photos. The resulting models were manually annotated with facial landmarks in 3D. Some ground truth data (about 500 faces) was used to compute our 3D face model with the help of PCA. Other ground truth models were used for testing our face shape capture algorithm (along with corresponding Kinect videos).

We compute 2D face tracking errors as Euclidian distances between projections of key points on aligned 3D face models and annotated 2D landmarks. The 3D face tracking errors are computed as Euclidian distances between these 3D key points and nearest depth frame points. The face shape capture errors are computed as Euclidian distances between 3D landmarks on ground truth models and corresponding 3D vertices on computed face models. We compute root-mean-square (RMS) errors for each video and for each face shape capture. We use 20–33 subjects to test our face tracking and 100 subjects to test our face shape capture.

The two charts in Figs. 3 and 4 compare 2D/3D accuracy of our face tracking system with the new depth constraint enabled and with this constraint disabled. All other constraints and conditions are the same in both tests and defined as in formula (5). When depth data is available, the face tracker pre-computes a face scale from five facial points found during initialization (see Section 2.3). We run tests against 33 videos

Fig. 7. Facial expression samples (left) and the corresponding deformed 3D model instances (right) computed by our face tracking engine. The model's 3D pose is mirrored.

Table 1
Face tracking performance results for different CPUs. This table lists processing times and number of iterations required to compute face alignment for one frame.

CPU	Processing time in milliseconds	Number of iterations
Intel i5	11–13	3–5
Intel i7	5–7	3–5

(one test subject per video). Both cases are similar in terms of 2D alignment errors, but the experiment with the enabled depth constraint shows significantly better 3D results. Big 3D errors in the disabled depth constraint case (video only 2D + 3D AAM) happen when tracking children—our mean face 3D model is closer to an adult head and so video-only projective-based fitting provides very poor results. Our experiments show that depth data corrects these errors well. The 3D accuracy improvements are in 7.02–686.46 mm range. If we exclude test cases where the 3D model shape is far from the test subject faces (when tracking children), then the average gain in 3D accuracy is 23.95 mm. We can conclude that using depth data in our face alignment algorithm leads to significantly improved 3D accuracy.

The charts in Figs. 5 and 6 compare two cases where we use the mean face versus a fitted personalized model as a 3D model in our face tracking system (it has the depth constraint enabled in both cases). We first compute personalized face models for all 20 test subjects and then run tracking tests. 2D performance is almost the same. 3D accuracy is slightly better with the personalized 3D model. The 3D accuracy improvements are in 0.01–2.35 mm range. Faces that are further away from the mean face show the greatest improvement.

Fig. 7 shows samples of some facial expressions 3D alignments produced by our face tracking system. The 3D model is rendered as mirrored.

Table 1 shows face tracking performance results. It lists the processing times and number of iterations required to compute face alignment for one frame. In this experiment, we used: a 2D AAM with 100 2D shape points and a 50x50-pixel template; a 3D model with 1347 vertices, 17 animation deformation parameters, and 110 shape deformation parameters. Processing time is listed with the measured range. Actual times depend on various factors, such as camera distance (which produces more RGBD data to process when the user is closer), head rotation, and speed of head motion.

Fig. 8 shows the maximum and RMS errors for face models computed by our face shape modelling algorithm for 100 people. The RMS errors are in 1.95–5.60 mm range and the max errors are in 4.98–15.87 mm range.

Fig. 9 shows examples of the computed face models compared to the ground truth. Each face is shown in frontal, ¾, and profile view.

4. Conclusions

In this paper, we proposed how to extend a 2D + 3D AAM based face tracker to use depth data from a RGBD camera like Kinect. This extension significantly improves 3D accuracy of AAM based tracker. The resulting real-time system tracks faces with 3–6 mm accuracy.

In Section 2.1 we showed why video data based 2D + 3D AAM is not accurate enough for 3D face tracking. It computes face alignments by minimizing distances between projected 3D model vertices and aligned 2D AAM points. This method is error prone, because tracking an object of unknown shape with a camera lacking depth data is an ill-posed problem. In Section 2.3 we introduced a new depth-based term into the face tracker's energy function. The term is formulated similarly to ICP energy function. It minimizes distances between 3D model vertices and nearest input depth points. Test results in Section 4 show significant improvements in 3D tracking accuracy when we use our new constraint in AAM fitting (with improvements in the range of 7.02–686.46 mm). The biggest improvements in 3D accuracy occur when we track children—our mean face model is closer to an adult head and therefore video-only 2D + 3D AAM produces significant 3D errors when tracking children. We also show how to initialize our 3D face model by computing its shape from a batch of tracking data and corresponding input depth frames. We incorporate the updated 3D model back into the face tracker to improve its accuracy further. The accuracy improvements (in 0.01–2.35 mm range) are not as great as in the case where we introduce depth data into 2D + 3D AAM.

Yet, in spite of these improvements, our tracker isn't perfect. Currently it can tolerate some occlusions, but fails when more than a quarter of a face is occluded. The 2D AAM fitting is not stable enough in the presence of larger occlusions. Consequently, thick glasses, beards or big mustaches can cause alignment errors. This problem can be reduced by proactively removing outliers in 2D AAM fitting. Our tracking runtime uses three different AAM models to cover more head rotations (frontal, left, right). Switching between these models may yield wrong face alignments for a few frames. This manifests itself as incorrectly computed facial animations even when a steady face rotates left to right. We believe that adding statistical regularization for our animation parameters $\{AU_i\}$ based on possible facial expressions can reduce this problem. The tracking system should produce only the most probable facial expressions in this case.

Our face shape computation algorithm relies on depth data and 2D face points. Therefore if some areas of a face are occluded by hair or something else and it is visible in depth, then our shape computation produces deformed shapes for those areas. We can mitigate this by using color-based skin segmentation and then removing areas from the depth image that are not skin from the shape computation.

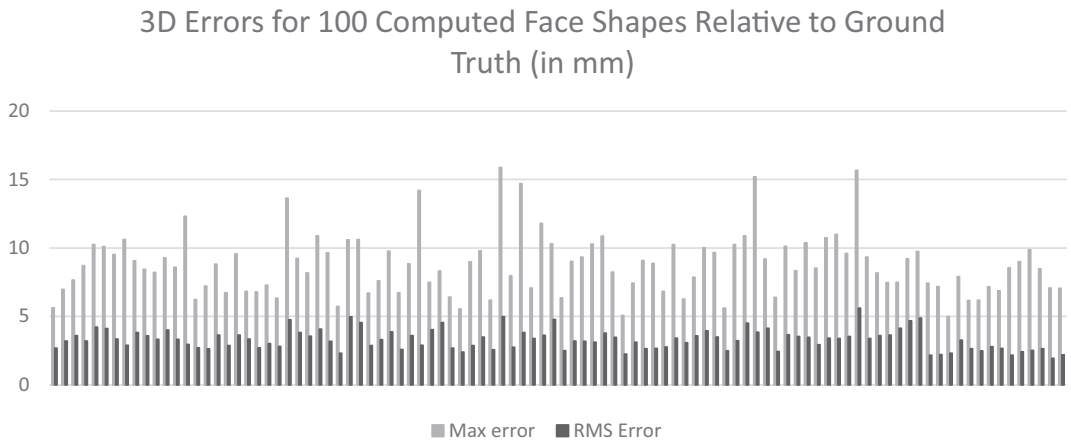

Fig. 8. Absolute maximum and RMS 3D errors for 100 computed face models relative to high-precision ground truth masks computed by a stereo rig.

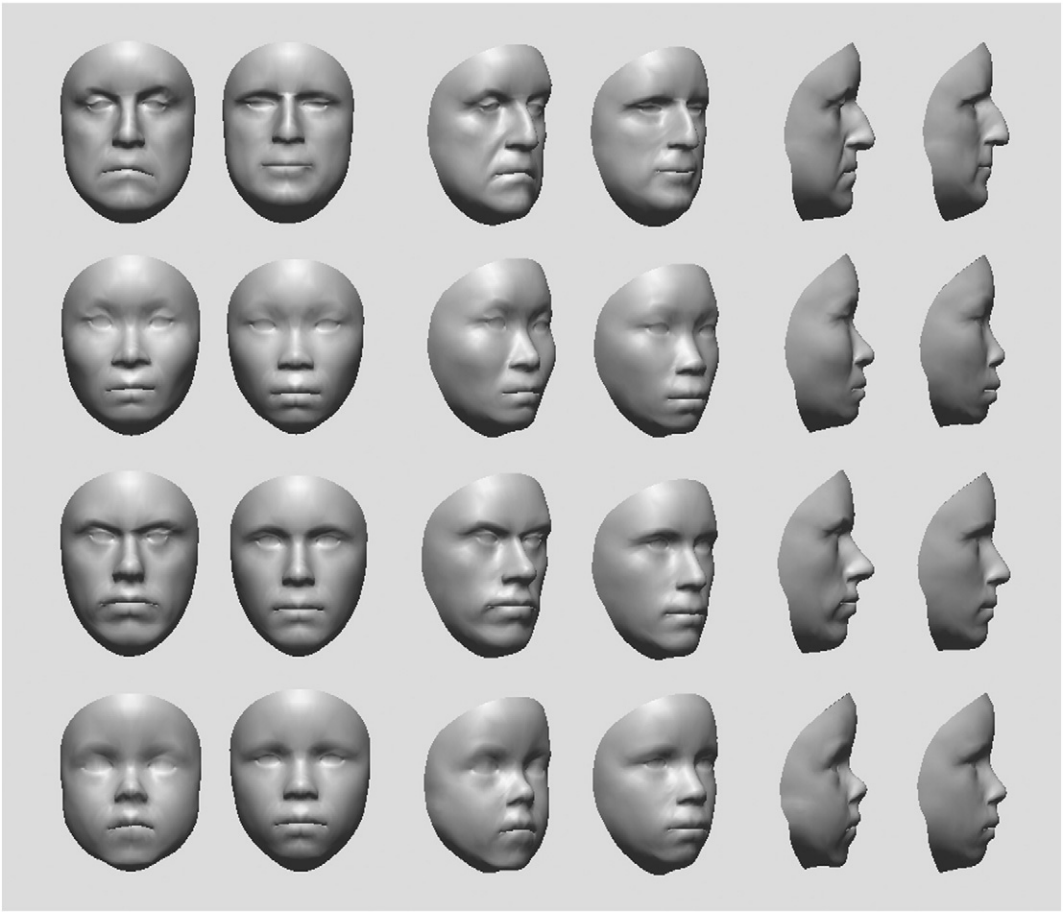

Fig. 9. Ground truth masks (left columns in each pose) versus computed face models (right columns in each pose). The RMS errors for these reconstructions are in 1.95–4.50 mm range. The second row reconstruction has the smallest error. The bottom row reconstruction has the largest error.

RGBD cameras have temporal and systematic noise. Higher levels of noise can contribute to inaccurate results. For example, Kinect's temporal depth noise increases quadratically with distance. We found that we need to lower the weight of our depth constraint beyond 2-meter distances to reduce noise influence, thus our tracker transitions into a mostly video-based tracker at larger distances. Time-of-flight depth cameras have their own systematic bias when computing distances to human faces. We plan to build error models for different depth cameras to estimate probabilities of measurement errors for input pixels. We can use such probabilities to reduce weights of noisy input data.

References

[1] T.F. Cootes, G.J. Edwards, C.J. Taylor, Active appearance models, Eur. Conf. Comput. Vis. 2 (1998) 484–498.
[2] T.F. Cootes, C.J. Taylor, Constrained active appearance models, Int. Conf. Comput. Vis. (2001) 748–754.
[3] I. Scott, T. Cootes, C. Taylor, Improving appearance model matching using local image structure, Conference on Information Processing in Medical Imaging 2003, Springer-Verlag, 2003, pp. 258–269.
[4] V. Blanz, T. Vetter, A morphable model for the synthesis of 3D faces, SIGGRAPH, 1999. 187–194.
[5] I. Matthews, S. Baker, Active appearance models revisited, Int. J. Comput. Vis. 60 (2) (2004) 135–164.
[6] S. Baker, R. Gross, I. Matthews, Lucas-kanade 20 years on: a unifying framework: part 4, Technical Report CMU-RI-TR-04-14, Robotics Institute, Carnegie Mellon University, 2004.
[7] J. Xiao, S. Baker, I. Matthews, T. Kanade, Real-time combined 2D + 3D active appearance models, Comput. Vis. Pattern Recognit. (2004) 535–542.
[8] I. Matthews, T. Ishikawa, S. Baker, The template update problem, IEEE Trans. Pattern Anal. Mach. Intell. 26 (6) (June, 2004) 810–815.
[9] R. Lengagne, P. Fua, O. Monga, 3D face modelling from stereo and differential constraints, Int. Conf. Pattern Recognit. (1998) 148–153.
[10] M. Zhou, L. Liang, J. Sun, Y. Wang, AAM based face tracking with temporal matching and face segmentation, Comput. Vis. Pattern Recognit. (2010) 701–708.
[11] S. Rusinkiewicz, M. Levoy, Efficient variants of the ICP algorithm, Proceedings of 3rd International Conference on 3D Digital Imaging and Modeling, 2001, pp. 145–152.
[12] Q. Cai, D. Gallup, C. Zhang, Z. Zhang, 3D deformable face tracking with a commodity depth camera, Eur. Conf. Comput. Vis. (2010) 229–242.
[13] T. Weise, S. Bouaziz, H. Li, M. Pauly, Realtime performance-based facial animation, ACM Transactions on Graphics, SIGGRAPH, 2011.
[14] S. Bouaziz, Y. Wang, M. Pauly, Online modelling for realtime facial animation, ACM Transactions on Graphics, SIGGRAPH, 2013.
[15] J. Saragih, S. Lucey, J. Cohn, Face alignment through subspace constrained mean-shifts, Int. Conf. Comput. Vis. (2009) 1034–1041.
[16] W. Zhang, Q. Wang, X. Tang, Real time feature based 3-d deformable face tracking, Eur. Conf. Comput. Vis. (2008) 720–732.
[17] T.F. Cootes, G.V. Wheeler, K.N. Walker, C.J. Taylor, View-based active appearance models, Image Vis. Comput. 20 (9–10) (2002) 657–664.
[18] R.I. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, Cambridge, UK, 2004.