Automated Capture of Real-time 3D Facial Geometry and Motion

Jing Chi[1,2] and Caiming Zhang[1]

[1]Shandong Economic University, peace_world_cj@hotmail.com
[2]Shandong Provincial Key Laboratory of Digital Media Technology, czhang@sdu.edu.cn

ABSTRACT

We present a novel automated method for capturing real-time 3D facial geometry and motion. Our method takes as input real-time face point clouds, and uses a deformable mesh model to fit each point cloud to capture the varying expressions. We propose a new normal-keeping constraint in the metric for fitting the deformable mesh to the point cloud, which not only enforces the consistency of vertex normal and intra-frame vertex motion to achieve automated capture, but also works effectively when the deformable mesh is very different from the point clouds in geometry. We also propose a new angle constraint to avoid generating poor triangles on the deformable mesh. Compared with existing techniques, our method 1) avoids manual selection of feature points, so it achieves automated capture. 2) avoids unstable optical flow estimation in traditional automated techniques, so it has high robustness. 3) always maintains good mesh structure in capture. Experiment results show the efficiency of our method.

Keywords: automated motion capture, real-time point clouds, mesh fitting.
DOI: 10.3722/cadaps.2011.859-871

1 INTRODUCTION

Capturing the shape and dynamic performance of a person's face is an attractive and challenging problem in computer graphics, not only due to its wide application in synthetic character animation, expression recognition, face modeling, etc., but also due to its inherent problems in capturing the accurate facial likeness and behaviors. The real-time point clouds that record the facial geometry and motion can be acquired with 3D scanner running at video rate, e.g., 24 frames-per-second, or recovered from the synchronized 2D image sequences with spatio-temporal stereo matching technique [4]. If we directly reconstruct a mesh on each point cloud to reflect the expression at a certain time, these meshes may have different connectivity, and not establish the intra-mesh corresponding points that map to the same point on the person's face. These limitations make it difficult to accurately reflect the subtle expressional variations and reanimate the captured expressions. Therefore, we need to build a sequence of meshes with the same connectivity and integrally represent the mesh sequence with a single deformable mesh model. The mesh sequence will reflect the time-varying facial expression and the single deformable mesh model can support further processing such as expression editing, surface deformation analysis, and so on.

It is difficult to capture dynamics of expression easily and accurately sine the complex anatomical structure of human face allows for large numbers of subtle expressional variations while humans are

especially sensitive to any unnatural expressions. Although the amazing realism of the synthetic facial animation in recent films makes a strong case that these difficulties can be overcome with the aid of highly skilled artists, tremendous amount of artistry, skill, and time will be spent in manual aid. Almost all of current capturing techniques need manual intervention to generate adequate details and realism. Additionally, in many cases, the specific model designed for capturing expressions of a certain human face may fail in another due to the individual difference among facial structures. Therefore, study of automated and self-adaptive capturing techniques that could decrease manual intervention and improve personality adaptability is in urgent need.

[8],[10],[13],[18] take as input the face image data obtained from 2D video sequences, and focus on accurately capturing facial feature points located around the eyes, nose, mouth, etc. Theses techniques have relatively low realism because many distinct characteristics of a person's expression lie in the subtle details such as wrinkles and furrows, not in the feature points. [3] and [14] compute the deformable model directly from the regular video streams. However, these techniques produce low-resolution results, compare to, e.g., structured light stereo.

[1],[7],[15-17] introduce high-resolution deformable models to capture the subtle expressions. [17] utilizes a high-resolution deformable parametric model to fit through the face shape sequence from a single view. [7] chooses different spherical subsets of the model volume and determines different affine transformations within these regions, then blends linearly the affine transformations between the sphere centers to get a smooth deformation field. The blended deformation is not optimal as the affine transformations are determined independently per sphere. [16] uses a multi-resolution model to fit the face point clouds. It performs global rigid deformations on the coarse level of the face model while local non-rigid deformations on the fine level. The non-rigid deformations integrate an implicit shape representation and B-spline based Free Form Deformation (FFD). The implicit representation increases computational complexity while FFD is less intuitive to control deformation. [1] extends the iterative closest point (ICP) framework to non-rigid registration by incorporating additional constraints in the closest point search. This technique accurately recovers the facial expressions, but works poorly when the deformable mesh and the point clouds have very different facial shapes. The techniques in [1],[7],[15-17] all take the feature correspondence as an important factor to constraint the deformation of the deformable mesh model. So they cannot automatically capture expressions since they require users to select facial feature correspondences manually for fitting each point cloud.

[2] uses a multi-scale face model to animate the spatial and temporal behavior of a person's facial wrinkles. The multi-scale model first computes a large-scale linear deformation, on top of which medium-scale wrinkles are synthesized. The technique does not need manual intervention, but the wrinkles on actor's face should be marked with different diffuse colors in order to establish the feature correspondences automatically. This way of motion data acquisition is time-consuming and makes actor uncomfortable. [9] computes an implicit function in R4 to approximate the time-space surface of the time-varying point clouds. It can get coherent meshes approximating the input data at arbitrary time instances, but the evaluation of the implicit function is computationally expensive. [12] computes a fitting shape automatically from the real-time point clouds, but the computation is complex and time-consuming. [5-6],[11] utilize optical flow as constraint to achieve automated expression capture. [11] first computes optical flow from 2D image sequences, then uses optical flow as constraint to deform the deformable model to fit the point cloud. The utilization of optical flow avoids manual selection of feature correspondences in capture, but image sequence is not always acquired. For those points having no texture information, their motions cannot be constrained. In addition, computation of optical flow is complex. Furthermore, estimation of optical flow is not robust in some cases, which may result in the unwanted deformations.

To overcome the limitations discussed above, we present a novel automated method for capturing 3D facial geometry and motion. Our new method takes as input the real-time face point clouds obtained at video rate, and uses a deformable mesh model to fit each point cloud to capture the varying expressions. We introduce a new proposed normal-keeping constraint into the new fitting metric to enforce the consistency of vertex normal and intra-frame vertex motion. The normal-keeping constraint not only ensures the deformable mesh automatically approximate the point cloud without manual selection of feature correspondences, but also gets good result when the deformable mesh is very different from the point clouds in facial shape. We also introduce a new proposed tangle constraint into the new fitting metric to maintain the original good mesh structure of the deformable

mesh in capture. Our method can achieve high-quality automated expression capture without manual aid and improve adaptability to face point clouds of different humans. Moreover, our method does not need 2D image sequence, and has low complexity and high stability, so the efficiency and robustness of our new method are higher than those automated techniques based on optical flow.

2 CONTRIBUTIONS

Our new method adopts the mesh fitting idea used in many existing techniques, but we have advantages: 1) Our method avoids manual selection of feature points on deformable mesh and each point cloud, which is usually needed in existing techniques, so it can achieve automated capture. 2) Without feature point constraint, the traditional ICP framework in existing techniques may generate wrong fitting results in many cases where there are rigid and non-rigid deformations between two adjacent point clouds, our method can always get correct results by introducing the normal-keeping constraint, so it can achieve accurate automated capture. 3) Our method avoids unstable optical flow estimation in traditional automated capture techniques, so it does not need synchronized 2D image sequence and has high robustness. 4) The triangle shapes of the deformable mesh may become poor, e.g., slivers, in capturing process in existing techniques, but our method can maintain good triangle shapes of the deformable mesh by introducing the angle constraint.

3 OVERVIEW: NEW METHOD

3.1 Overview

We present a new automated method for capturing 3D facial geometry and motion. The new method takes as input the real-time face point clouds acquired with 3D scanner at video rate or recovered from 2D image sequence. Each point cloud can be triangulated as a mesh that reflects the facial expression at a certain time. We call the triangulated mesh the target mesh. All the target meshes compose a target mesh sequence. We use a deformable parametric mesh, which we call the template mesh, to fit through the target mesh sequence. We propose a new metric to compute the optimal deformation of the template mesh to approximate each target mesh closely. The new meshes generated by deforming the template mesh have the same connectivity and accurately recover the expressional variation with all the subtleties. The time-varying deformable template mesh that integrally represents these new meshes can support further processing.

Our new method directly uses the first frame of the target mesh sequence as the template mesh to fit through the target mesh sequence with new proposed metric and without manual intervention. Our method also allows for using user-defined mesh as the template mesh. In this case, it only needs a small amount of user guidance to fit the template mesh to the first target mesh, and then automatically fit the initial deformed template mesh through the rest meshes in the sequence with new proposed metric and without manual intervention.

3.2 Representation

Let $\Upsilon = \{T_m \mid m = 1,2,...,M\}$, be a target mesh sequence reconstructed from the real-time point clouds with M frames. Each frame in the target mesh sequence can be represented as $T_m = (V_m, E_m, F_m)$, where V_m is vertex set, E_m is edge set, and F_m is face set. Obviously, geometry and connectivity are both different between different target meshes. We can use the first frame T_1 or user-defined mesh as the template mesh. For notational convenience, Let $S = (V, E, F)$ be the template mesh, with vertex set $V = \{v_i\}$, edge set $E = \{(i_1, i_2)\}$ and face set $F = \{(i_1, i_2, i_3)\}$. Our goal is to deform the template mesh onto each target mesh to capture the varying expressions, and integrally represent the target mesh sequence as the deformations of the single template mesh with the certain connectivity.

To fit to the target mesh, we assign an affine transformation defined by 3×3 matrix Q and displacement vector d, which, for notational convenience, we write as $Q + d$, to per vertex on the template mesh S. These unknown affine transformations supply enough degrees of freedom to

capture the subtle deformations of the target mesh that accurately reflect the expressional variations. We use a new metric to solve for these affine transformations so that the mesh S' with vertex set $\{Q_i v_i + d_i\}$, obtained by deforming S with these affine transformations, optimally approximates the target mesh. The new metric consists of a closest-point term, a smoothness term, a normal-keeping term, and an angle term. The first two terms are usually used in many existing capture techniques, but the last two terms are new proposed and adopted in our method. The normal-keeping term constraints the intra-frame motion of the template vertices to achieve automated fitting, and gets the suggested results even when the template mesh is very different from the target mesh in shape. The angle term constraints the shapes of the template triangles to get good mesh structure.

4 NEW METRIC

4.1 The Closest-point Term

The closest-point term measures the distance between the template mesh after being deformed and the target mesh. Naturally, the distance should be small. This term is expressed as

$$E_\alpha(\{Q_i + d_i\}) = \sum_{v_i \in V} w_i \left\| Q_i v_i + d_i - q_i \right\|^2 \tag{4.1}$$

where $Q_i v_i + d_i$ is the new location of v_i after being transformed, i.e., the vertex on the deformed template mesh S', and its closest compatible point on the target mesh T is denoted by q_i. $\|Q_i v_i + d_i - q_i\|$ is the Euclidean distance between $Q_i v_i + d_i$ and q_i. w_i is a weight factor to control the influence of the target mesh on the template mesh.

In our method, a vertex v_i' on S' and a point q' on T is considered to be compatible if the difference in orientation of vertex normal at v_i' and triangle normal at q' is less than 90° and the distance between v_i' and q' is within a threshold. For each vertex v_i' on S', we first search its closest compatible vertex on the target mesh, then compute the triangle set Φ sharing the closest vertex, finally project v_i' into each triangle of Φ to find the closest compatible point. Note that, the computed closest compatible point may be a vertex or a point in one triangle of the target mesh. If no compatible point could be found, the weight w_i is set to zero. In closest-point search, the triangle set Φ sharing each vertex depends only on the target mesh, so we compute and store them only once. Furthermore, we judge the normal compatibility before computing Euclidean distance to remove incompatible vertices or triangles on the target mesh. All these accelerate the closest-point computation.

4.2 The Smoothness Term

Generally, simply deforming the template mesh only with the closest-point constraint will not result in a very attractive mesh, because the neighboring vertices on the template mesh may displace to disparate parts of the target mesh. For example, if the target mesh does not completely cover the template mesh, the template mesh may deform without penalty where there is no data. Furthermore, determining one affine transformation per template vertex supplies numerous degrees of freedom, and there may be many affine transformations that have the same effect on a single vertex. Therefore, we introduce a smoothness term to enforce the neighboring template vertices to undergo similar affine transformations. Specifically,

$$E_\beta(\{Q_i + d_i\}) = \sum_{(i_1, i_2) \in E} \left\| (Q_{i_1} + d_{i_1}) - (Q_{i_2} + d_{i_2}) \right\|_F^2 \tag{4.2}$$

where $\|\cdot\|_F$ is the Frobenius norm.

4.3 The Normal-keeping Term

In many cases, the computed deformation of the template mesh only with the closest-point and the smoothness constraints is still unwanted. For example, when the template mesh does not completely cover the target mesh, the template mesh after being deformed may bunch together in some regions while still approximating the target mesh closely and undergoing the similar affine transformations between neighboring template vertices. To address this problem, most of the existing methods impose the correspondence constraint on certain feature points located around the eyes, nose, mouth, etc. This constraint measures the distance between the corresponding feature points on the template mesh and the target mesh. Naturally, the distance should be small. The correspondence constraint works effectively when the template mesh and the target mesh are different in shapes, but it requires users to select the corresponding feature points. Obviously, the manual selection of feature points in fitting the template mesh to each target mesh makes these methods unable to achieve automated capture.

In our method, the target mesh sequence is acquired at video rate, so all frames in a sequence are similar in shapes, and adjacent frames are very close spatially and temporally, that is, the time interval between adjacent frames is very short, the distance between adjacent frames is very close, and the deformation between adjacent frames is very small. Therefore, in our method, we discard the correspondence constraint in fitting the template mesh through the target mesh sequence. However, it may result in wrong fitting results only with the closest-point and the smoothness constraints in many cases where there are both rigid and non-rigid deformations between two adjacent frames. As shown in Fig.1, T_k and T_{k+1} are two adjacent frames in a target mesh sequence. There are both small rigid and non-rigid deformations between T_k and T_{k+1}. Assuming that the template mesh after deformation has fitted to T_k closely, then, the deformed template mesh and T_{k+1} satisfy the properties of adjacent frames. So the problem of fitting the deformed template mesh to T_{k+1} can be transformed to the problem of fitting T_k to T_{k+1}. Without loss of generality, we regard T_k as the template mesh and T_{k+1} as the target mesh to discuss the fitting between them. It can be seen from Fig.1(c), only with the closest-point and the smoothness constraints, the vertices on the lower lip of the template mesh after deformation move towards the upper lip wrongly. To overcome the problem, we introduce the normal-keeping constraint in our method.

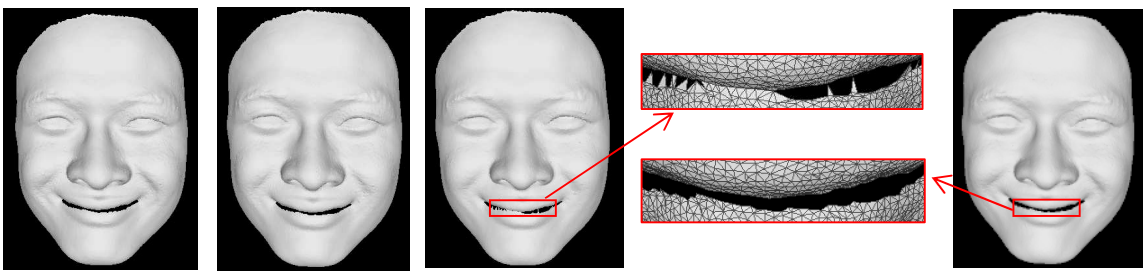


Fig. 1: Fitting two adjacent frames in a target mesh sequence. (a) The prior frame T_k. (b) The next frame T_{k+1}. (c) The result after fitting T_k to T_{k+1} only with closest-point and smoothness constraints. (d) The result after fitting T_k to T_{k+1} with closest-point, smoothness, and normal-keeping constraints. Rectangles are respectively close-up view of the mouth of meshes in (c) and (d).

In ideal case, if the template mesh is deformed to optimally approximate the target mesh, then each template vertex after being displaced should map onto a point on the target mesh, and has the same normal as the target point. Here, the target point may be a vertex on the target mesh, or a point in one triangle on the target mesh. Naturally, the deformed template mesh well reflects the shape of the target mesh since its vertices are located on the target mesh. This ideal case means that, to achieve optimal fitting result, not only the distance between the deformed template mesh and the target mesh should be small, but also the directional difference between normals on the displaced template vertex

and the corresponding target point should be small. Therefore, we use the consistency of normal directions as an important factor to constraint the deformation of the template mesh. Specifically,

$$E_{\gamma}(\{Q_i + d_i\}) = \sum_{v_i \in V} u_i Agl^2(N_{Q_i v_i + d_i},\ N_{q_i}) \tag{4.3}$$

where $Q_i v_i + d_i$ and q_i are the corresponding points as defined as in Eqn. (4.1). $N_{Q_i v_i + d_i}$ is the vertex normal on $Q_i v_i + d_i$, N_{q_i} is the surface normal on q_i, and $Agl(N_{Q_i v_i + d_i}, N_{q_i})$ is the angle between $N_{Q_i v_i + d_i}$ and N_{q_i}. u_i is a weight factor which is set to zero where no corresponding point q_i could be found for $Q_i v_i + d_i$. We call the Eqn. (4.3) the normal-keeping constraint.

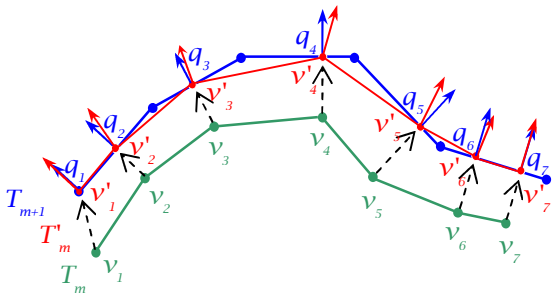


Fig. 2: Illustration of automated fitting between adjacent frames without the normal-keeping constraint. T_m (green) and T_{m+1} (blue) are two adjacent frames. T'_m (red) is the resulted mesh after fitting T_m to T_{m+1}. For each vertex v'_i on T'_m, its closest point on T_{m+1} is denoted as q_i. The distance between T'_m and T_{m+1} achieves minima, but for most v'_i on T'_m, their normals (red arrow) have very different directions from that of their closest point q_i (blue arrow) on T_{m+1}.

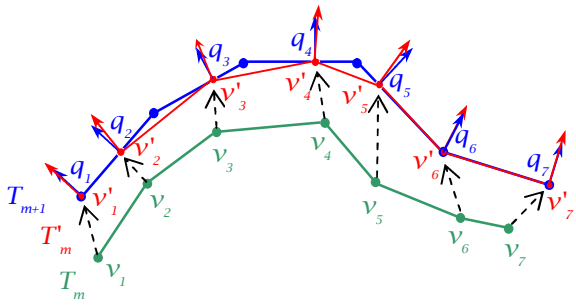


Fig. 3: Illustration of automated fitting between adjacent frames with the normal-keeping constraint. T_m (green) and T_{m+1} (blue) are two adjacent frames as same as in Fig.2. T'_m (red) is the resulted mesh after fitting T_m to T_{m+1}. The distance between T'_m and T_{m+1} achieves minima, and the normals on v'_i (red arrow) and its closest point q_i (blue arrow) have approximately the same directions.

The normal-keeping constraint can measure the consistency of vertex normal and intra-frame vertex motion. Combined with the closest-point and the smoothness constraints, it can deform the template mesh to approximate the target mesh optimally. These constraints do not refer to feature correspondence, so it can achieve automated fitting. As shown in Fig.2, T_m (green) and T_{m+1} (blue) are two adjacent frames. There are small rigid and non-rigid deformations between T_m and T_{m+1}. Only with the closest-point constraint and the smoothness constraint, the mesh T_m with vertices $\{v_i\}$ may be deformed to T'_m with vertices $\{v'_i\}$ (red). Here, the distance between T'_m and T_{m+1} achieves minima, but many vertices on T'_m have very different normal directions from their corresponding closest points on T_{m+1}, e.g., v'_2 and q_2, v'_3 and q_3, v'_4 and q_4, v'_5 and q_5, v'_6 and q_6. It means that the computed affine transformations are not optimal in this case. It is necessary to adjust the affine transformations considering the normal consistency as well as the distance minimization. As shown in Fig. 3, T_m (green) and T_{m+1} (blue) are the same two adjacent frames as in Fig. 2. With combination of the closest-point constraint, the smoothness constraint, and the normal-keeping constraint, mesh T_m with vertices $\{v_i\}$ is deformed to T'_m with vertices $\{v'_i\}$ (red). Now, the distance between T'_m and T_{m+1} achieve minima,

and the normals on v_i' and q_i have approximately the same directions. Obviously, the mesh T_m' in Fig.3 is the suggested result because its shape is more similar to T_{m+1} than T_m' in Fig.2.

As shown in Fig.1(c), only with the closest-point and the smoothness constraints, some vertices on the lower lip of mesh T_k move towards wrong positions. The new positions of these vertices are very close to mesh T_{k+1}, but the normal directions on the new positions are very different from that on their corresponding points on T_{k+1}. Fig.1(d) is the result that fits T_k to T_{k+1} by introducing the normal-keeping constraint. It can be seen that the vertices on the lower lip of mesh T_k move towards correct positions. These vertices after being displaced are close to T_{k+1}, and have the approximately same normal directions as their corresponding points on T_{k+1}. Here, utilization of normal-keeping constraint can effectively avoid the unwanted results. Furthermore, these constraints do not involve the feature correspondence term. Therefore, we can fit arbitrary adjacent frames in the target mesh sequence without manual intervention.

If a user-defined mesh is used as the template mesh in our method, there may be large difference between the template mesh and the target mesh. In this case, the normal-keeping constraint also performs effectively. We only need to select a few feature points on the template mesh and the target mesh to estimate the initial values of optimization (estimation details is described in section 5), then, combined with the closest-point and the smoothness constraints, the normal-keeping constraint can gradually tune the deformation of the template mesh in iterative optimization to approximate the target mesh closely. Because we just use the feature points to estimate initial values and do not involve the correspondence constraint in optimization, we need fewer feature points than those traditional techniques using feature correspondence constraint. Moreover, the feature correspondences do not have to be very accurate. As shown in Fig. 4, (a) is the user-defined template mesh, (b) is the target mesh. (a) and (b) are very different in shapes. Fig. 4(c) is the resulted mesh after fitting the template mesh to the target mesh with the closest-point, the smoothness and the feature correspondence constraints. We identify 33 pairs of corresponding feature points on the template mesh and the target mesh (some shown as orange dots). The mesh (c) has large errors in many positions such as the regions around the eye and on the mouth. Fig. 4(d) is the resulted mesh after fitting the template mesh to the target mesh with the closest-point, the smoothness, and the normal-keeping constraints. We only select 24 pairs of corresponding feature points, which is subset of the feature correspondences in Fig.4(c), to estimate the initial values of optimization. It can be seen that the mesh (d) approximates the target mesh closely and captures all the subtleties of expression.

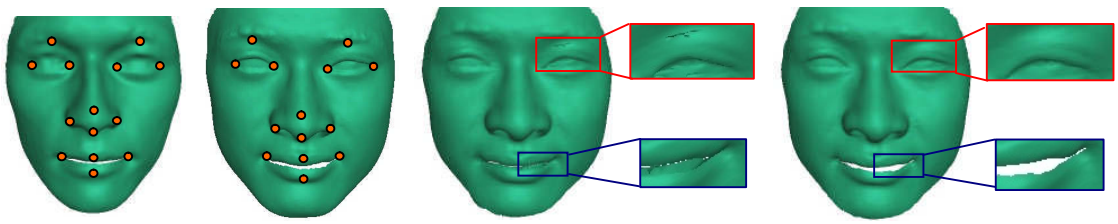


Fig. 4: Comparison of the fitting results with different constraints. (a) The template mesh. (b) The target mesh. (c) The result computed with closest-point, smoothness, and feature correspondence constraints. (d) The result computed with closest-point, smoothness, and normal-keeping constraints. Rectangles are close-up view of the eye and the mouth. Orange dots mark feature correspondences.

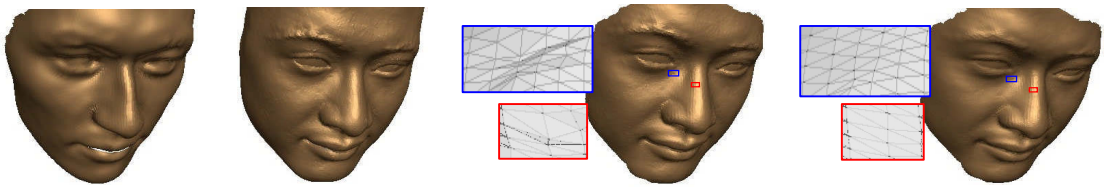
4.4 The Angle Term

In the mesh fitting discussed as above, we focus on automatically displacing the vertices of the template mesh to approximate the target mesh closely, and do not consider the mesh structure of the template mesh after deformation. It may result in poor triangle shapes, e.g., slivers, in some cases. As shown in Fig.5, the triangle shapes of the template mesh are good, but after fitting to the target mesh, some triangle shapes of the template mesh become poor due to the displacements of the vertices.

Therefore, we propose the angle constraint to enforce the template triangles to maintain good shapes as soon as possible. Specifically,

$$E_{\delta}(\{Q_i + d_i\}) = \sum_{f_j \in F} \sum_{v_i \in v(f_j)} (Agl_{Q_i v_i + d_i} - Agl_{v_i})^2 \tag{4.4}$$

Here, for each triangle f_j of the template mesh, $v(f_j)$ is the set of vertices on f_j. Agl_{v_i} is the angle on v_i before deformation, $Agl_{Q_i v_i + d_i}$ is the angle on v_i after deformation. Obviously, the smaller the difference of Agl_{v_i} and $Agl_{Q_i v_i + d_i}$ is, the better the shape of triangle f_j maintains.



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

Fig. 5: The fitting results with or without the angle constraint. (a) The template mesh. (b) The target mesh. (c) The result computed without the angle constraint. (d) The result computed with the angle constraint. Rectangles are close-up view around the eye and the nose. Although meshes (c) and (d) both approximate the target mesh closely, some triangle shapes in (c), e.g., those in the rectangles, become poor. However, these same triangles preserve good shapes in (d), as shown in rectangles in (d).

4.5 Combining the Terms

As discussed above, the new metric for fitting the template mesh through the target mesh sequence in our method, i.e., the full objective function, is a weighted sum of Eqns. (4.1)-(4.4), specifically

$$E(\{Q_i + d_i\}) = \alpha E_{\alpha} + \beta E_{\beta} + \gamma E_{\gamma} + \delta E_{\delta} \tag{4.5}$$

where the weights α, β, γ and δ are tuned to guide the optimization. By iteratively minimizing Eqn. (4.5), we can get the affine transformations that optimally deform the template mesh onto the target mesh. We solve the minimization using L-BFGS-B algorithm [19]. In iterative optimization, we set weights $\alpha = 1, \beta = 1, \gamma = 0.1,$ and $\delta = 0.001$ in first step, then respectively increase α from 1 to 51, and β from 1 to 9 in following four steps. Generally, it will get ideal results after five iterations.

If the template mesh is obtained from a user-defined mesh that may be very different from the target mesh sequence in shape and expression, we will use the new metric to deform the template mesh to optimally approximate the first frame of the target mesh sequence, initialized with a small amount of user guidance. Let the deformed template mesh that has optimally fit the first frame be S_1 (or S_1 is directly obtained from the first frame of the sequence), we like to deform it smoothly through the rest of the sequence with the new metric. Specifically, starting from S_1, we recursively get S_{m+1} that optimally approximates the (*m+1*)-th frame by deforming S_m that has fit the *m*-th frame. Each optimal deformation in the capturing process can be computed without any manual intervention.

5 IMPLEMENTATION OF NEW METHOD

We implement our new method in the following two steps.

Step 1. If the template mesh is obtained from a user-defined mesh, we first deform the template mesh to fit the first frame of the target mesh sequence. If the template mesh is automatically obtained from the first frame, this step can be skipped. In this step, we will initialize the optimization with a small amount of manual aid. Specifically, we first select some corresponding feature points on both the template mesh and the target mesh, then solve for an over-constrained global affine transformation from the feature correspondences, and finally, iteratively optimize the new metric with the global affine transformation as the initial values of the unknowns to obtain the optimal deformation of the

template mesh. A small number of selected feature points in this step are enough and the feature correspondences do not have to be very precise since they are only used to estimate the initial values. With these initial values, the iterative optimization of the new metric will adjust the deformation of the template mesh to get an accurate final fitting result. After this step, the selected feature points are no longer used in the following capturing process.

Step 2. Let the deformed template mesh obtained in step1 be S_1 (or let the first frame of the target mesh sequence be S_1), we would automatically fit it to the rest meshes of the sequence. Specifically, we deform S_1 to fit the second frame T_2 and denote the resulted mesh as S_2; deform S_2 to fit the third frame T_3 and denote the resulted mesh as S_3; ..., deform S_{M-1} to fit the *M*-th frame T_M and denote the resulted mesh as S_M. All the meshes $S_m, m = 1,2,...,M,$ have the same connectivity and accurately reflect the varying expressions. In the capturing process, we compute the deformation in each fitting by iteratively optimizing the new metric. The new metric does not contain the feature correspondence constraint, but uses the normal-keeping constraint to enforce the consistency of vertex normal and intra-frame vertex motion. Therefore, our method can automatically fit through the target mesh sequence without manual intervention. Furthermore, compared with those automated techniques based on optical flow, our method does not have to take synchronized 2D image sequence as input, and has stronger robustness.

Additionally, our new method could automatically fill in missing data. The weights in Eqns. (4.1) and (4.3) are set to zero for the template vertices whose closest point is located on a boundary edge of the target mesh, then the affine transformations on these template vertices are only affected by the smoothness term. So the hole on the target mesh will be filled in by seamlessly transformed parts of the template mesh. As shown in Fig.6, the details that were not available in the target meshes can be drawn effectively from the template mesh.

6 EXPERIMENTAL RESULTS

We conducted our new method on the real-time face point clouds (reconstructed as the target mesh sequence) of different humans acquired at video rate. We respectively use the user-defined mesh and the first frame as the template mesh to capture these mesh sequences. We implement our system using C++ under the windows environment.

Fig. 6 shows the results of fitting through a target mesh sequence with lost details. Fig. 6 (a) is the user-defined template mesh. Fig. 6 (b) and (d) are two frames in the sequence. The meshes do not scan well, so some details are lost in the meshes. For example, the mesh in Fig. 6 (b) has holes on the right brow, and the mesh in Fig. 6 (d) has holes on the left eye and left brow. Our new method not only fits through the mesh sequence automatically, but also fills in the holes of the scanned meshes automatically. Fig. 6 (c) shows the result after fitting the template mesh to the mesh in Fig.6 (b), note that the hole has been filled in automatically. Fig. 6 (e) shows the result of fitting to the mesh in Fig. 6(d), which is obtained in the automated capturing process. Likewise, the holes have been filled in.

Fig. 7 shows the results of capturing a target mesh sequence automatically with our new method. Fig. 7 (a) is the template mesh obtained directly from the first frame of the sequence. Fig. 7 (b) and (d) are two frames in the sequence. Tab. 1 lists the geometric information about the meshes (a), (b) and (d). Fig. 7 (c) and (e) respectively show the result of fitting to mesh (b) and (d) in automated capturing process. It can be seen that the results well capture the time-varying expressions.

To measure the fitting quality in the capturing process, we use two measures for objective estimation of the fitting results. In the first measure, we average over the angle between the normals of corresponding points on the template mesh after being deformed and the target mesh. Intuitively, if the template mesh has optimally fit the target mesh, the normals of corresponding points should have approximately the same direction, that is, the average angle should be small. In the second measure, we average over the squared Euclidean distance of corresponding points on the deformed template mesh and the target mesh. The average angle and the average squared Euclidean distance obtained from the two deformed results in Fig. 7 (c) and (e) are given in Tab. 2.

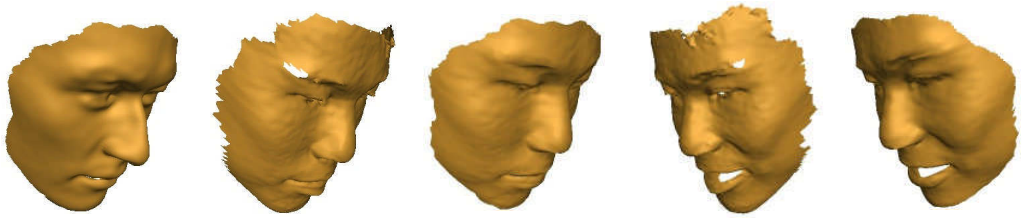


Fig. 6: The results of capturing a target mesh sequence with lost details. (a) is the template mesh. (b) and (d) are two frames in the sequence. (c) and (e) are results that respectively fit to (b) and (d).

Fig. 7: The results of capturing a mesh sequence. (a) The first frame used as the template mesh. (b) A frame in the sequence. (c) The result of deforming the template mesh to optimally fit to the frame (b), which is automatically obtained in capturing process. (d) Another frame in the sequence. (e) The result of fitting to the frame (d), which is automatically obtained in capturing process.

Mesh	vertices	triangles
Fig.7 (a)	9962	19459
Fig.7 (b)	10053	19668
Fig.7 (d)	9953	19409

Tab.1: Numbers of vertices and triangles of meshes in Fig.7.

Meshes	Fig.7(c) and Fig.7(b)	Fig.7(e) and Fig.7(d)
Average angle	9.1E-8	2.9E-7
Average distance	2.4°	2.2°

Tab. 2: Estimation of fitting quality in Fig.7.

Fig. 8 shows the results of capturing another mesh sequence automatically with our new method. Fig. 8 (a) is the user-defined template mesh model and Fig. 8 (c) is the first frame of the target mesh sequence. Fig. 8 (b) and (d) are respectively the close-up view of the geometry and connectivity around the nose of mesh (a) and (c). The local regions corresponding to (b) and (d) are marked by red circles in (a) and (c), respectively. It can be seen that the user-defined template mesh is very different from the target mesh in facial shape. The optimal deformation of the template mesh is computed as described as step1 of our new method, and the deformed template mesh that optimally fits the first frame (c) is shown in Fig. 8 (e). In the following capturing process, as discussed as step2 of our new method, the optimal deformation that fits each target mesh is computed without manual intervention. Fig. 8 (f)-(i) shows some resulted meshes that reflect continuous expressions, and the interval of each mesh is 6 frames. Fig. 8 (j)-(m) are some other selected results after fitting the template mesh through the whole sequence. The meshes in Fig. 8 (f)-(m) well reflect the subtle expressional variations.

7 CONCLUSIONS

In this paper, we have developed a new automated method for capturing real-time 3D facial geometry and motion. The new method takes as input the time-varying face point clouds obtained at video rate. By assigning one affine transformation to each vertex of the deformable template model, the new method can supply enough degrees of freedom to capture the subtle expression details. The new method uses the normal-keeping constraint, which aims to enforce the consistency of vertex normal and intra-frame vertex motion, in the new metric to achieve automated capture of the target mesh sequence reconstructed from the point clouds. The normal-keeping constraint also works effectively when the template mesh model is very different from the target meshes in facial shapes. The new method uses the angle constraint to maintain good mesh structure in capturing process. Additionally, the new method can automatically fill in missing data in the target meshes.

Our method performs effectively on the mesh sequence acquired at video rate since the adjacent frames of the sequence are close enough, but may generate unwanted results when the time-space interval of adjacent frames is large. In the future work, we will look into improvement of the method with consideration of both the normal-keeping constraint and automated detection of facial features.

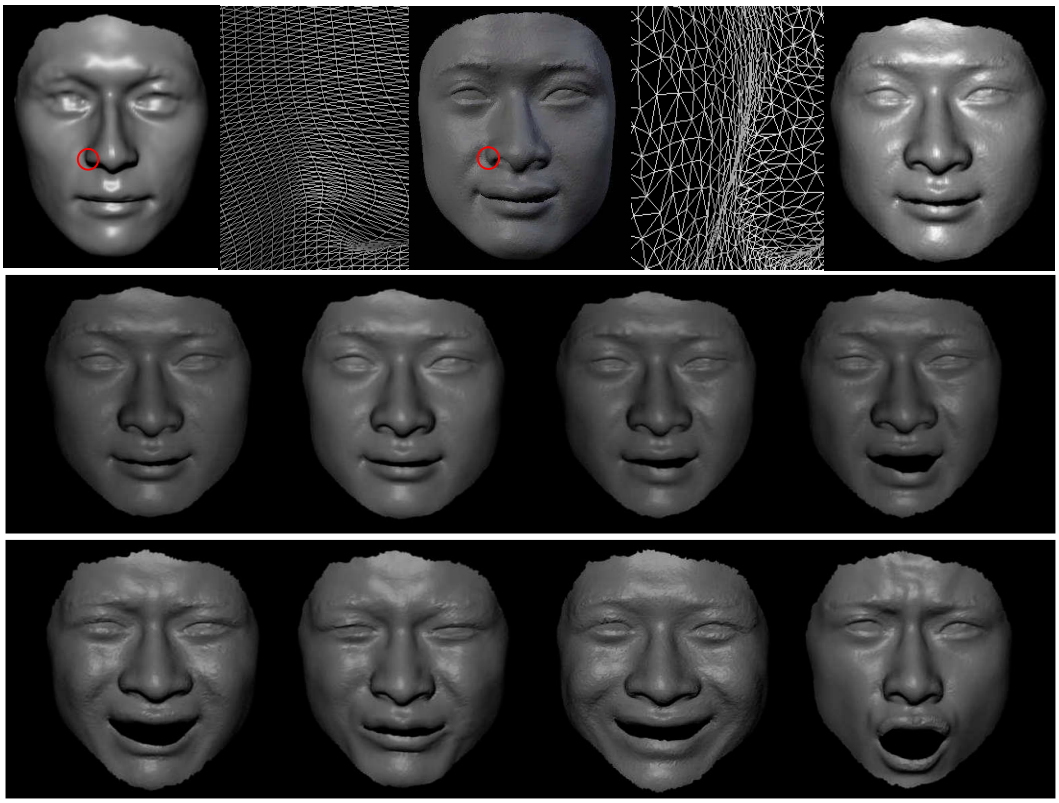


Fig. 8: The capturing results with our new method. (a) The user-defined template mesh. (b) The close-up view of the geometry and connectivity around the nose of the template mesh. (c) The first frame of the target mesh sequence. (d) The close-up view of the geometry and connectivity around the nose of the first frame. (e) The result after fitting the template mesh to the first frame. Starting from the mesh (e), we automatically fit through the whole sequence with our method. (f)-(i) are some resulted meshes with continuous expressions. (j)-(m) are some other selected results obtained in the capturing process.

ACKNOWLEDGEMENTS

We are grateful to Prof. Li Zhang at the University of Washington, for the help of 3D face data acquisition. This work was supported by National Nature Science Foundation under grant 60903109, and Nature Science Foundation of Shandong Province under grant ZR2010FQ031, ZR2010FQ025.

REFERENCES

[1] Amberg, B.; Romdhani, S.; Vetter, T.: Optimal step nonrigid ICP algorithms for surface registration, In Proc. IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, Minnesota, USA, 18-23 June, 2007, 1-8. DOI: 10.1109/CVPR.2007.383165

[2] Bickel, B.; Botsch, M.; Angst, R.; Matusik, W.; Otaduy, M.; Pfister, H.; Gross, M.: Multi-scale capture of facial geometry and motion, ACM Tansactions on Graphics (TOG), 26(3), 2007, Article 33. DOI: 10.1145/1276377.1276419

[3] Brand, M.: Morphable 3D models from video, In Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Kauai, HI, USA, 8-14 December, 2001, 456-463. DOI: 10.1109/CVPR.2001.990997

[4] Davis, J.; Ramamoorthi, R.; Rusinkiewicz, S.: Spacetime stereo: a unifying framework for depth from triangulation, In Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Madison, Wisconsin, USA, 16-22 June, 2003, 359-366. DOI: 10.1109/CVPR.2003.1211491

[5] Decarlo, D.; Metaxas, D.: Optical flow constraints on deformable models with applications to face tracking, International Journal of Computer Vision, 38(2), 2000, 99–127. DOI: 10.1023/A:1008122917811

[6] Decarlo, D.; Metaxas, D.: Adjusting shape parameters using model based optical flow residuals, IEEE Trans. on Pattern Analysis and Machine Intelligence, 24(6), 2002, 814-823. DOI: 10.1109/TPAMI.2002.1008387

[7] Feldmar, J.; Ayache, N.: Rigid, affine and locally affine registration of free-form surfaces, Int. Journal of Computer Vision, 18(2), 1996, 99–119. DOI:10.1007/BF00054998

[8] Lien, J.; Kanade, T.; Zlochower, A.; Cohn, J.; Li, C.: Subtly different facial expression recognition and expression intensity estimation, In Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Santa Barbara, CA, USA, 23-25 June, 1998, 853-859. DOI: 10.1109/CVPR.1998.698704

[9] Süßmuth, J.; Winter, M.; Greiner, G.: Reconstructing animated meshes from time-varying point clouds, Computer Graphics Forum, 27(5), 2008, 1469-1476. DOI: 10.1111/j.1467-8659.2008.01287.x

[10] Essa, L.; Pentland, A.: A vision system for observing and extracting facial action parameters, In Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Seattle, WA, USA, 21-23 June, 1994, 76-83. DOI: 10.1109/CVPR.1994.323813

[11] Zhang, L.; Snavely, N.; Curless, B.; Seitz, S. M.: Spacetime faces: high resolution capture for modeling and animation. ACM Transaction on Graphics, 23(3), 2004, 548-558. DOI: 10.1145/1015706.1015759

[12] Wand, M.; Adams, B.; Ovsjanikov, M.; Berner, A.; Bokeloh, M.; Jenke, P.; Guibas, L.; Seidel, H.-P.; Schilling, A.: Efficient reconstruction of non-rigid shape and motion from real-time 3D scanner data, ACM Transactions on Graphics, 28(2), 2009, Article 15. DOI: 10.1145/1516522.1516526

[13] Goldenstein, S. K.; Vogler, C.; Metaxas, D.: Statistical cue integration in dag deformable models, IEEE Transactions on Pattern Analysis and Machine Intelligence, 25(7), 2003, 801-813. DOI: 10.1109/TPAMI.2003.1206510

[14] Torresani, L.; Yang, D. B.; Alexander, E. J.; Bregler, C.: Tracking and modeling non-rigid objects with rank constraints, In Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Kauai, HI, USA, 8-14 December, 2001, 493-500. DOI: 10.1109/CVPR.2001.990515

[15] Wang, Y.; Gupta, M.; Zhang, S.; Wang, S.; Gu, X.F.; Samaras, D.; Huang, P.S.: High resolution tracking of non-rigid motion of densely sampled 3D data using harmonic maps, International Journal of Computer Vision, 76(3), 2008, 283-300. DOI:10.1007/s11263-007-0063-y

[16] X. Huang; S. Zhang; Y. Wang; D. Metaxas; D. Samaras: A hierarchical framework for high resolution facial expression tracking, In Proc. IEEE Workshop on Articulated and Nonrigid Motion (ANM'04) in conjunction with CVPR'04, Washington D.C., USA, 2 June, 2004, 22-29. DOI : 10.1109/CVPR.2004.7

[17] Wang, Y.; Huang, X.; Lee, C.-S.; Zhang, S.; Li, Z.; Samaras, D.; Metaxas, D.; Elgammal, A.; Huang, P.: High Resolution Acquisition, Learning and Transfer of Dynamic 3-D Facial Expressions, Computer Graphics Forum, 23 (3), 2004, 677-686. DOI:10.1111/j.1467-8659.2004.00800.x

[18] Yacoob, Y.; Davis, L.: Computing spatio-temporal representations of human faces, In Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Seattle, WA, USA, 21-23 June, 1994, 70-75. DOI:10.1109/CVPR.1994.323812

[19] Zhu, C.; Byrd, R. H.; Lu, P.; Nocedal, J.: Algorithm 778. L-BFGS-B: Fortran subroutines for Large-Scale bound constrained optimization, ACM Transactions on Mathematical Software, 23(4), 1997,550-560. DOI: 10.1145/279232.279236