Automatic Reconstruction of Personalized Avatars from 3D Face Scans

Michael Zollhöfer, Michael Martinek, Günther Greiner, Marc Stamminger, Jochen Süßmuth
Computer Graphics Group, University Erlangen-Nuremberg, Germany

Abstract

We present a simple algorithm for computing a high quality personalized avatar from a single color image and the corresponding depth map which have been captured by Microsoft's Kinect sensor. Due to the low market price of our hardware setup, 3D face scanning becomes feasible for home use. The proposed algorithm combines the advantages of robust non-rigid registration and fitting of a morphable face model. We obtain a high quality reconstruction of the facial geometry and texture along with one-to-one correspondences with our generic face model. This representation allows for a wide range of further applications such as facial animation or manipulation. Our algorithm has proven to be very robust. Since it does not require any user interaction, even non-expert users can easily create their own personalized avatars.

Keywords: personalized avatars, face scanning, face reconstruction, face model fitting, Microsoft Kinect sensor

1 Introduction

During the past few years, massive multiplayer online games (*MMOGs*) such as World of Warcraft, Aion or Second Life have gained tremendous attention. In these games, millions of users, each represented by a virtual character – known as *avatar* – meet in a three-dimensional virtual world. Since the users should be distinguishable from each other, each user must have a unique avatar. Most MMOGs offer a character editor in which the user can select his avatar from a set of existing models. These avatars can be further customized using simple user interfaces which allow the user to morph between different shapes or to include customized textures or models.

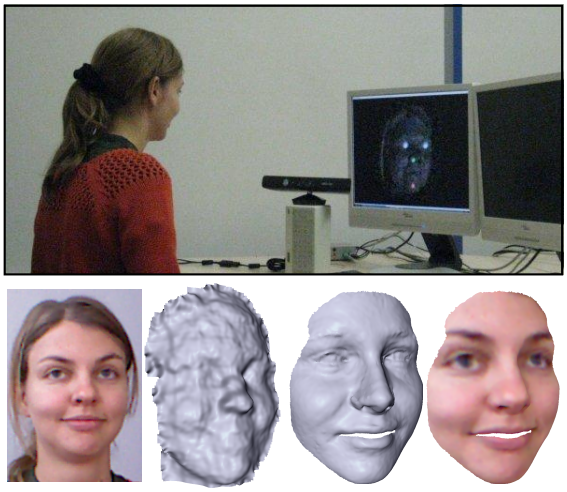

Figure 1: Overview of the proposed avatar reconstruction. The upper image shows our acquisition setup in action. The lower row shows the recorded color image and depth scan and the fitted mesh without and with textures.

As the graphics in computer games become ever more realistic and the focus of the games changes from fantasy scenarios towards real life settings, the demand for photorealistic avatars, which resemble the users themselves, is constantly increasing. Unfortunately, current in-game avatar editors offer only a limited expressibility, making it almost impossible for a user to create a virtual clone for the online world.

Besides their usage in online games and chats, personalized 3D avatars are also important in gadget applications such as aging simulations or digital 3D beauty salons.

In this paper, we present a fully automatic method for the generation of realistic three-dimensional face models with textures. As in-

put for our algorithm we use a depth scan and a color image that were recorded using the Microsoft Kinect sensor. This device was initially released as a video game controller for the Microsoft Xbox. Meanwhile, official open source drivers have been released which enable the general purpose usage of the device on a regular personal computer. Due to the low market price and its widespread use, the Microsoft Kinect sensor is ideally suited for recording personalized 3D avatars. Using our system, everyone can automatically digitalize his face and create a personalized avatar within seconds.

1.1 Related Work

The reconstruction of personalized 3D face models has become very popular during the last years. Two different approaches exist for this purpose: algorithms which try to reconstruct the 3D model solely from color images and algorithms which reconstruct the 3D model by fitting a template mesh to a depth scan of the face.

The first class of algorithms reconstructs the facial geometry directly from one or more 2D images. Tang and Huang [1] automatically extract salient facial features from a front and profile face image. The detected features are then used to adapt a coarse template mesh. Since this method requires a one-to-one relationship between facial features and the vertices of the template mesh, it can only produce very coarse reconstructions. Blanz and Vetter [2] employ an iterative optimization to adapt the parameters of a three-dimensional morphable model by projecting the current morphable model onto the image plane and then minimizing the difference between the pixel colors. To improve the stability of their method, the optimization is performed in a coarse to fine manner. Breuer et al. [3] use Support Vector Machines to detect the face in a 2D image and then extract facial features using a regression- and classification-based approach. After that, they apply a flip-flop optimization to determine the best-fitting morphable model for the detected features, which is, in turn, used to improve the feature detection and classification. Instead of reconstructing the three-dimensional facial model from features in 2D images, Lee et al. [4] fit a morphable face model in such a way that the shading of the model is as close to the shading of the input image as possible. Commercially available software tools such as AvMaker, FaceGen or FaceShop also compute a 3D avatar from features in 2D images. These features can either be automatically detected or hand-picked by the user. Common to all methods which reconstruct the 3D face model solely from photographs is that while they can accurately reconstruct the face in feature-rich regions, the fitting in feature-less regions like the cheek or the forehead is rather poor.

Methods of the second category reconstruct the 3D face by fitting a template mesh to depth scans. Weissenfeld et al. [5] construct a multi-resolution detail pyramid of the input face scan by successive Laplace smoothing. Using a set of manually selected features, they fit a generic face model to the multi-resolution detail pyramid in a coarse-to-fine manner. Blanz et al. [6] describe an iterative algorithm for fitting a morphable model to a textured depth scan. In each iteration, the current morphable model is projected onto a two-dimensional cylindrical image. An energy term which considers both, the color and the depth value after the projection, is then minimized to get a better-fitting morphable model. Basso and Verri [7] approximate the input depth scan by an implicit function and then solve for the parameters of a morphable face model such that the distance between the face and the depth scan is minimized. To improve the convergence of their method, they split the template mesh into four sub-meshes which are fitted independently. The sub-meshes are then smoothly blended to obtain the final reconstruction. Li et al. [8] employ a sophisticated non-linear optimization process to fit a source mesh to a target depth scan. By enforcing local rigidity and global smoothness of the deformation, they obtain high quality registrations given a good initial alignment. Most similar to the proposed algorithm is the work recently published by Kim et al. [9] since they also use non-rigid registration to fit a common template mesh to a depth scan. However, our algorithm is more robust in practice since we do not rely on robust 2D facial features during the non-rigid registration and since our regularization term tries to maintain surface features while their regularization term only minimizes surface stretch.

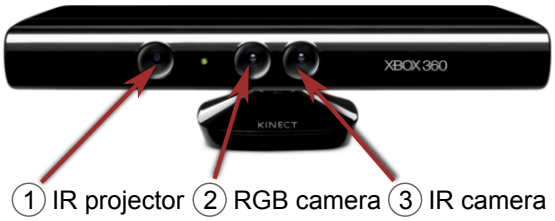

Figure 2: Microsoft Xbox Kinect sensor

1.2 Overview

The proposed algorithm automatically reconstructs a high quality 3D face model with texture from an RGB image and a depth map by fitting a morphable face model to the input data. We use the Microsoft Kinect sensor to capture an RGB image and the corresponding depth map from which a metric point cloud is computed. Using four automatically detected feature points, we estimate an initial rigid alignment of a common template mesh with the recorded point cloud. Since this alignment is in general not good enough to be used as input for the morphable model fitting, we non-rigidly register the average face of the morphable model with the input scan. Finally, we use the morphable model to compute the face which approximates the deformed template face best. The result of the proposed algorithm is a high-quality 3D model of the scanned face which has one-to-one correspondences to the faces in the morphable model. Thus, it can be easily analyzed, animated or modified. To add more realism, we compute a corresponding texture for the faces using the captured RGB data.

2 Data Acquisition

In this section, we briefly describe the way to obtain a raw depth image of the face of a person sitting in front of the Kinect and the augmentation of this image with valuable feature points as well as color information.

2.1 Input Device

The Kinect is a device which combines a regular RGB camera and a 3D scanner, consisting of an infrared (IR) projector and an IR camera as shown in Figure 2. The projector sends several thousands of structured IR rays into the scene which are reflected by objects and recaptured by the IR camera. The distortion between the emitted and the received pattern is used to reconstruct the depth values for each reflected ray. The driver interpolates the depth values between the rays and outputs a 640x480 depth grid with a precision of 11 bits @ 30 Hz. Microsoft officially specifies a depth range of 1.2m - 3.5m but in our experiments we found that a minimum distance of 0.5m is sufficient to receive good 3D images.

The RBG image is provided in the same resolution and framerate as the depth data, however, the two signals do not naturally match due to different extrinsic and intrinsic camera parameters. The exact parameters may even vary among different Kinect devices which makes an individual calibration unavoidable.

2.2 Data Preparation

To assist the calibration of the IR and the RGB camera, we use OpenCV's calibration routines to specify the required DOFs. After the calibration, the data is available as 3D point cloud with natural metric units and a corresponding RGB value is assigned to each point. Yet, we still maintain the topology information in form of the 640x480 grid from the raw data to simplify the feature detection and face segmentation, which requires neighborhood information for each point.

The depth data we recieve from the Kinect in a single frame is quite noisy and can contain holes at arbitrary positions. To reduce these effects, we average the depth values of eight successive frames, which leads to a much smoother result. Also, missing points in one frame may be compensated by other frames in which the device was able to capture these points. In addition, we apply a Gauss filter and simple hole-filling to the temporally smoothed data, which further improves the input data. Figure 3 shows a face scan at different stages of the preparation pipeline. To avoid motion blurring artifacts, it is required that the user remains motionless for at least eight frames (0.27s) while capturing the image.

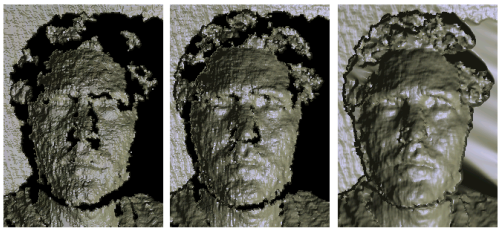

Figure 3: Left: raw data from a single frame. Middle: temporally smoothed data. Right: additionally Gauss-filtered data

2.3 Face and Feature Detection

Assuming that exactly one person is sitting in front of the Kinect, we first detect the face along with a number of significant feature points to assist the alignment of the scanned data at a later stage. We particularly aim to detect the middle point of each eye, the nose tip and the chin. Our idea is to locate regions of significant face parts in the RGB image, then map these regions onto the geometry data and to proceed the detection of feature points in the geometry domain.

For the first part, we use OpenCV to find the bounding rectangles of the face, the eyes and the nose in the RGB image (see Figure 4). After we have detected the face on the entire range of the input RGB image, we reduce the further searching domain in the image to the face's bounding box and detect the eyes and the nose. This reduction does not only provide better performance, but also increases robustness since the algorithm might falsely identify background parts as an eye or a nose.

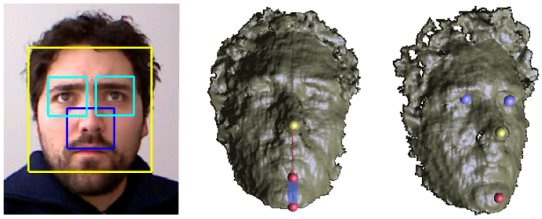

Figure 4: Left: face-, nose- and eye-detection on the RGB image. Middle: temporary chin features (red) and search domain for the final chin feature (blue area). Right: final feature points.

Due to the fact that the RGB data is aligned with the point cloud and that the topology is the same in both spaces, we can apply the rectangular regions directly in the geometry domain.

In order to find the nose tip, we loop through all 3D points inside the detected nose rectangle and pick the one with the smallest depth value as nose feature.

For the chin, no feature detectors are available in OpenCV which allow for a pre-selection of the chin's region in the RGB image. As a solution, we use the following heuristic: We first perform a line search from the detected nose tip down the y-axis until we find a significant ascend of the depth values. The resulting point is a temporary feature point which we call the chin edge. From this point, we sample the same line back to the nose until we detect a local maximum of the depth values. This second temporary feature point is named the chin groove. We now define the final chin feature to be the point between the chin groove and the chin edge which is closest to the camera. In order to compensate for a small roll of the subject's head (if the main face axis is not perfectly aligned with the y-axis), we extend the search region by a small offset of ± 5 pixels in x-direction.

An eye feature would ideally be the point on the eyeball closest to the camera. However, the resolution and the quality of the input data is not sufficient to robustly find these points on the geometry. As an approximation, we take the center point of the detected bounding box which turned out to be a robust estimate for the initial alignment at a later stage. Figure 4 (right) shows the final feature points on a scanned face.

Since the feature detection runs in real-time, a person sitting in front of the camera gets real-time feedback of the resulting face scan together with the detected feature points (see Figure 1). This allows the user to optimize his position before he eventually captures the current data. To further stabilize the features, we perform a smoothing operation over various frames as we did on the raw depth data.

2.4 Face Segmentation

The recorded depth data still contains unnecessary background data at this point. In order to reduce the costs in the next steps, we separate the face from the rest of the input data as seen in the scanned images from Figure 4. From the feature detection, we already know some points which definitly belong to the face. The rest of

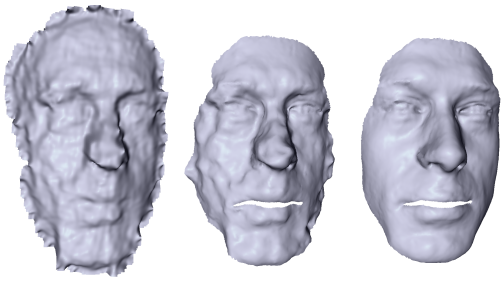

Figure 5: Generic face fitting: (from left to right) input scan, registered average face and best fitting morphable model.

the points is found with a floodfill-like algorithm using the detected face feature points as seed. In each recursion, we check the four-neighborhood of a current face point whether the depth values change by more than 5mm. If this is not the case, we add the corresponding point to the list of face points and recursively call the routine with this point. This method has turned out to robustly separate the face from the background and the body of the user.

3 Fitting a Generic Face Model

The point cloud that was generated in the previous step represents the geometry of the scanned face. However, except for the few detected feature points, the geometry does not contain any semantic information which would be necessary to animate or further process the face. To attribute the scanned face with a semantic meaning, we fit a morphable face model [2] to the scan. Therefore, we compute a rough initial alignment of the scanned point cloud $\mathcal{P}$ and the average face $\mathcal{T}$ of our morphable model. Given the previously computed features points and the corresponding points on the generic face model, we can compute the shape-preserving transformation which best aligns the two data sets by performing a generalized Procrustes analysis [10].

Theoretically, one could now solve for the parameters of the morphable model in such a way that it approximates the input scan as good as possible. Unfortunately, morphable models are not invariant under shape-preserving transformations which means that the input shape must be perfectly scaled and aligned with the morphable model to generate satisfactory results. We found that it is very difficult to produce such a rigid alignment without any user assistance or very accurate marker positions since a registration using the ICP algorithm [11] tends to converge into a local minimum if the deformation between the source and the target shape is too large. To obtain a more robust alignment for the final fitting of the morphable model, we compute a non-rigid registration of the template model $\mathcal{T}$ and the input scan.

Given a template mesh $\mathcal{T}$ and a coarsly aligned input scan $\mathcal{P}$, the goal of the non-rigid registration is to find a plausible space deformation Φ such that the distance between the deformed template mesh $\Phi(\mathcal{T})$ and the input scan $\mathcal{P}$ is minimized.

Following Suessmuth et al. [12], we formulate the non-rigid registration as a variational problem. Therefore, we define a registration energy E_{reg}, which is composed of a fitting term E_{fit}, that attracts the template mesh towards the input scan geometry, and an internal energy term E_{def}, that serves as regularizer and prevents unnatural deformations of the template mesh. The deformation Φ which best aligns the template mesh with the input scan is then found by minimizing the registration energy.

3.1 Fitting Energy Term

As can be seen in the left image of Figure 5, the pre-processed scanner data is still noisy, contains striping artifacts and may contain holes. To alleviate these artifacts, we do not register the template mesh directly with the scanner data but with an implicit function f which has been fitted to it. Since the implicit function f is computed in such a way that its zero-set approximates the input data in a least squares sense, it reduces the noise and closes the remaining holes. Given the implicit function $f : \mathbb{R}^3 \mapsto \mathbb{R}$, the distance d of a point $\mathbf{x}$ to the zero-set of f can be approximated by

$$d(\mathbf{x}, f) = \frac{|f(\mathbf{x})|}{\|\nabla f(\mathbf{x})\|_2}.$$

The distance between the deformed template mesh $\Phi(\mathcal{T})$ and the zero-set of f, which provides us with the fitting energy term, can then be defined as the sum over the squared distances

at the vertices $\hat{\mathbf{v}}_i$ of $\Phi(\mathcal{T})$:

$$E_{\text{fit}} = \sum_{\hat{\mathbf{v}}_i \in \Phi(\mathcal{T})} \left(\frac{|f(\mathbf{v}_i)|}{\|\nabla f(\mathbf{v}_i)\|_2} \right)^2 \quad (1)$$

3.2 Regularization Energy Term

We use a variant of the embedded graph based mesh deformation algorithm by Sumner et al. [13] to model the deformation of the template mesh. Thereby, a global space deformation is obtained by blending neighboring affine transformations $A_i(\mathbf{x}) = \mathbf{M}_i(\mathbf{x} - \mathbf{p}_i) + \mathbf{p}_i + \mathbf{t}_i$ with local support, which are organized in a sparse graph. Here, $\mathbf{M}_i$ is a 3x3 matrix, $\mathbf{p}_i$ is the node's position and $\mathbf{t}_i$ is a translation vector. The local transformations define how the surrounding space is deformed. To obtain a deformation, which maintains local surface features, the local transformations should be close to rigid. An energy term E_{rot}, which punishes the deviation of a local transformation $\mathbf{M}_i$ from being rigid, can be defined as:

$$E_{\text{rot}}(\mathbf{M}_i) = \|\mathbf{M}_i^T \mathbf{M}_i - \mathbf{I}\|_F^2 \quad (2)$$

Since neighboring transformations have overlapping influence, they affect a common region in space. It is therefore important that they are consistent w.r.t. one another. This is enforced by a consistency energy term E_{con}, which measures the distance between the position to where a graph node is transformed by its own transformation and the position where it is mapped to by the transformation of a neighboring graph node:

$$\begin{aligned} E_{\text{con}}(e_{ij}) = &\|A_i(\mathbf{p}_j) - A_j(\mathbf{p}_j)\|_2^2 \quad (3) \\ &+ \|A_j(\mathbf{p}_i) - A_i(\mathbf{p}_i)\|_2^2 , \end{aligned}$$

where e_{ij} is the edge connecting the two nodes i and j and $\mathbf{p}_i$ and $\mathbf{p}_j$ are the respective node positions.

3.3 Optimization

A combination of the fitting energy term and the two regularization energy terms leads to an energy function E_{reg}, which is a measure for the quality of a registration introduced by a given deformation graph:

$$E_{\text{reg}} = \alpha E_{\text{fit}} + \beta \sum_i E_{\text{rot}}(\mathbf{M}_i) + \gamma \sum_{e_{ij}} E_{\text{con}}(e_{ij}). \quad (4)$$

Here, the first sum runs over all nodes in the deformation graph and the second sum runs over all edges. An optimal deformation Φ, which registers the template mesh $\mathcal{T}$ with the input scan can now be computed by minimizing Equation (4) for the unknown node transformations using a modified Gauss-Newton algorithm. As proposed by Li et al. [8], we increase the influence of the fitting energy in each iteration by doubling α. Initially, we set $\alpha = 4$, $\beta = \gamma = 1$.

3.4 Fitting the morphable face model

The result of the non-rigid registration step is a deformed template mesh $\Phi(\mathcal{T})$ which tightly fits the input scan. However, as can be seen in Figure 5 (middle), this mesh still contains the bumps and dents that were originally present in the scanner data. To obtain the final 3D reconstruction of the face, we fit a morphable face model [2] to the deformed template obtained by the non-rigid registration. This projects the solution into the space of reasonable faces and thereby removes the mentioned artifacts.

Since the deformed template mesh provides one-to-one correspondences with the average shape of our morphable model, we can now robustly align $\Phi(\mathcal{T})$ with the morphable model using a Procrustes analysis again. Let $\hat{\mathcal{T}}$ be the deformed template mesh that was aligned with the average face $\mathcal{T}$ of the morphable model and $\mathbf{E}$ the (reduced) eigenbasis of the morphable model. We can thus construct a new face $\mathcal{F}$ from a given coefficient vector $\mathbf{c}$ by transforming the coefficient vector back into face space and adding the average face: $\mathcal{F} = \mathcal{T} + \mathbf{E} \cdot \mathbf{c}$. We find the coefficients $\mathbf{c}$ of the morphable model which approximate $\hat{\mathcal{T}}$ best by minimizing the least squares distance to the computed morph $\mathcal{F}$:

$$\min_{\mathbf{c}} \|(\mathcal{T} + \mathbf{E} \cdot \mathbf{c}) - \hat{\mathcal{T}}\|_2^2.$$

To obtain $\mathbf{c}$, we solve the resulting normal equations:

$$\mathbf{E}^T \mathbf{E} \cdot \mathbf{c} = \mathbf{E}^T \cdot (\hat{\mathcal{T}} - \mathcal{T}) \quad (5)$$

Since the pseudo-inverse of the system matrix $\mathbf{E}^T \mathbf{E}$ can be precomputed for the given morphable model, the unknown coefficients can be computed by a simple matrix-vector multiplication.

Figure 6: Transfering an animation onto a captured avatar.

4 Results

We have tested the proposed method on 20 individuals. In all cases, the proposed method was able to automatically produce accurate results. The results for four test persons are shown in Figure 8. As texture, we project the RGB image back onto the reconstructed face. The morphable model that we used for face fitting has been constructed from 53 faces [14], which were registered using the proposed algorithm for non-rigid registration. For all examples shown in this paper, we used the 25 most significant principal components of the face space to span the eigenbasis $\mathbf{E}$. In the non-rigid registration, we performed six Gauss-Newton iterations to warp the template mesh towards the input scan. The reconstruction of the fully textured facial avatars took on average 18 seconds on an Intel Core i7 processor at 2.93GHz.

To assess the quality of the reconstructed 3D face geometries, we compare a face model that was reconstructed using our algorithm to ground truth data in Figure 7. The ground truth face model was generated by scanning the test person with a high quality structured light scanner. The maximum deviation of our reconstruction from the ground truth model is 4.7mm, while the average deviation is roughly 2mm.

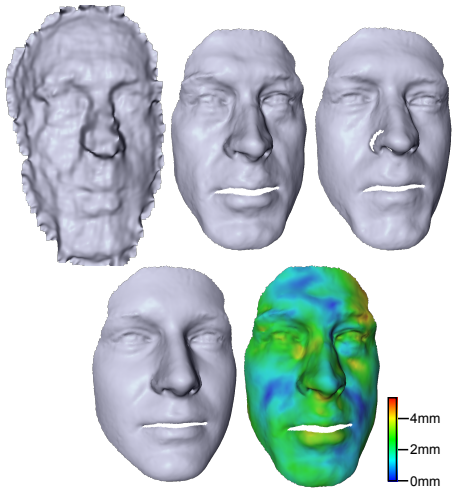

Figure 7: Comparison with ground truth. From top left to bottom right: depth scan, fitted result, ground truth, average shape and deviation from ground truth.

Since the morphed face model generated by our algorithm has one-to-one correspondences with the average face, we can directly transfer semantic informations that are annotated to the average face onto our reconstruction. For example, facial animations (cf. Figure 6) can be easily cloned onto the new geometry.

5 Conclusion

We have presented a novel system for the automatic generation of high-quality personalized avatars using the Microsoft Kinect sensor. The proposed system allows a huge audience to generate high-quality facial models within seconds. Using non-rigid registration to compute correspondences for the subsequent morphable model fitting makes our approach very robust and allows to generate convincing results even for bad input data.

The rather small morphable model we are using is currently one of the limiting factors. We plan to extend our data base and to handle models of the whole head. We further plan to extract geometry and texture information from multiple views and to capture whole head scans in super-resolution by registering the obtained input data. By using a segmented head model, we could further improve the expressability of the morphable model. In addition, we plan to animate the computed results and provide the user the possibility to customize his avatar.

Acknowledgements

This work was partly funded by the Deutsche Forschungsgemeinschaft (DFG).

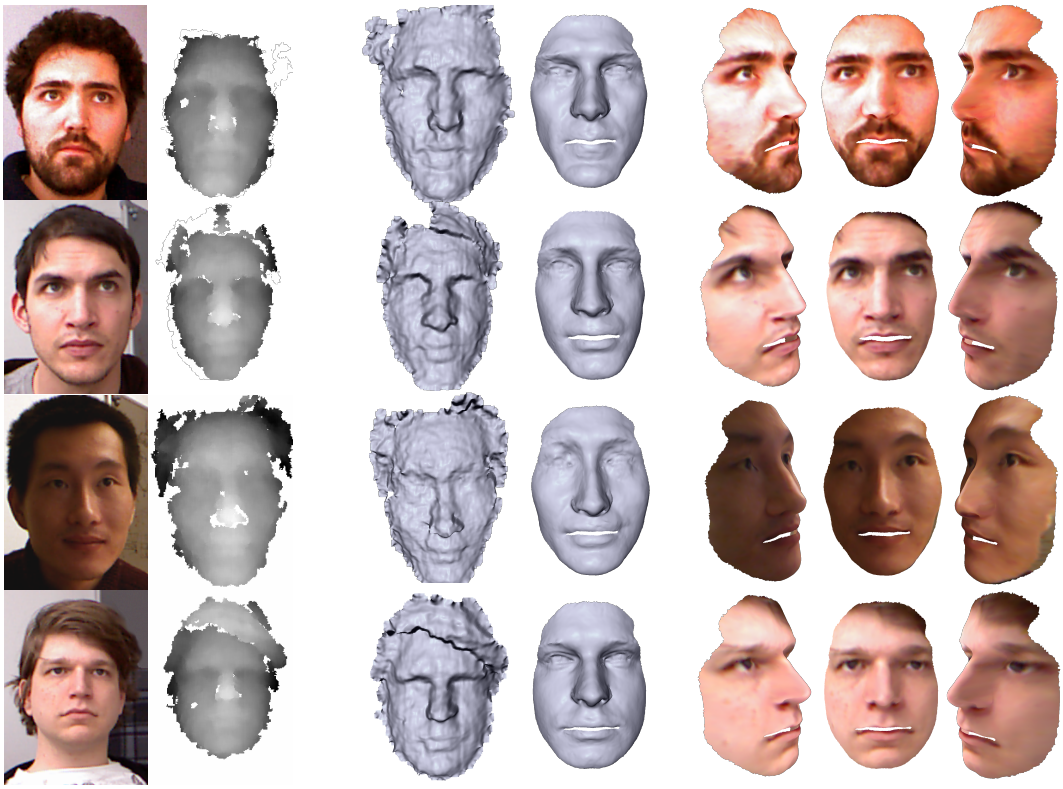

Figure 8: Face reconstruction results for four individuals. (from left to right) input RBG and depth images, processed depth scan, fitted face model and textured face from three different views.

References

[1] L.-A. Tang and T. S. Huang. Automatic Construction of 3D Human Face Models Based on 2D Images. In *Proc. Image Processing'96*, pages 467–470, 1996.

[2] V. Blanz and T. Vetter. A Morphable Model for the Synthesis of 3D Faces. In *Proc. Siggraph'99*, pages 187–194, 1999.

[3] P. Breuer, K. I. Kim, W. Kienzle, V. Blanz, and B. Schölkopf. Automatic 3D Face Reconstruction from Single Images or Video. In *Proc. FG'08*, pages 1–8, 2008.

[4] J. Lee, R. Machiraju, H. Pfister, and B. Moghaddam. Estimation of 3D Faces and Illumination from Single Photographs Using A Bilinear Illumination Model. In *Proc. EGSR'05*, pages 73–82, 2005.

[5] A. Weissenfeld, N. Stefanoski, S. Qiuqiong, and J. Ostermann. Adaptation of a Generic Face Model to a 3D Scan. In *Proc. ICOB'05*, 2005.

[6] V. Blanz, K. Scherbaum, and H.-P. Seidel. Fitting a Morphable Model to 3D Scans of Faces. In *Proc. ICCV'07*, pages 1–8, 2007.

[7] C. Basso and A. Verri. Fitting 3D Morphable Models using Implicit Representations. *JVRG*, 4(718), 2007.

[8] H. Li, R. W. Sumner, and M. Pauly. Global Correspondence Optimization for Non-Rigid Registration of Depth Scans. *Computer Graphics Forum*, 27(5):1421–1430, 2008.

[9] Y. S. Kim, H. Lim, B. Kang, O. Choi, K. Lee, J. D. K. Kim, and C.-Y. Kim. Realistic 3D Face Modeling Using Feature-Preserving Surface Registration. In *Proc. ICIP'10*, pages 1821–1824, 2010.

[10] J. C. Gower. Generalized Procrustes Analysis. *Psychometrika*, 40(1):31–51, 1975.

[11] P. J. Besl and N. D. McKay. A Method for Registration of 3-D Shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, 1992.

[12] J. Süßmuth, M. Zollhöfer, and G. Greiner. Animation Transplantation. *Computer Animation and Virtual Worlds*, 21(3-4):173–182, 2010.

[13] R. W. Sumner, J. Schmid, and M. Pauly. Embedded Deformation for Shape Manipulation. *ACM Trans. Graph.*, 26(3):Article 80, 2007.

[14] A. B. Moreno and A. Sánchez. GavabDB: A 3D Face Database, March 2004.