Computer-Aided Design 58 (2015) 117–122

Contents lists available at ScienceDirect

Computer-Aided Design

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

Lightweight wrinkle synthesis for 3D facial modeling and animation☆

Jun Li[a], Weiwei Xu[b,*], Zhiquan Cheng[c], Kai Xu[d,*], Reinhard Klein[a]

[a] Bonn University, Germany
[b] Hangzhou Normal University, China
[c] Avatar Science Company, China
[d] National University of Defense Technology, China

HIGHLIGHTS

- We present a lightweight non-parametric approach to generate wrinkles for 3D facial modeling and animation.
- Our method represents a convenient approach for generating plausible facial wrinkles with low-cost.
- Our method enables the reconstruction of captured expressions with wrinkles in real-time.

ARTICLE INFO

Keywords:
3D facial animation
Texture synthesis
Lightweight wrinkle generation

ABSTRACT

We present a lightweight non-parametric method to generate wrinkles for 3D facial modeling and animation. The key *lightweight* feature of the method is that it can generate plausible wrinkles using a single low-cost Kinect camera and one high quality 3D face model with details as the example. Our method works in two stages: (1) offline personalized wrinkled blendshape construction. User-specific expressions are recorded using the RGB-Depth camera, and the wrinkles are generated through example-based synthesis of geometric details. (2) Online 3D facial performance capturing. These reconstructed expressions are used as blendshapes to capture facial animations in real-time. Experiments on a variety of facial performance videos show that our method can produce plausible results, approximating the wrinkles in an accurate way. Furthermore, our technique is low-cost and convenient for common users.

1. Introduction

3D facial modeling is an important research topic in computer graphics, mainly due to its wide applications in game and film industries. Recent progress in 3D facial modeling research advocates the importance of the modeling of fine-scale wrinkles, since it can reinforce the perception of the complex emotions in the facial performance [1,2]. For example, the frown lines are indispensable clues of unpleasant or disapproval emotions.

Laser scanning, multi-view stereo reconstruction and sophisticated deformation methods have been applied to the modeling of highly-detailed 3D facial models with fine-scale wrinkles [3–5]. However, these systems usually run in controlled laboratory environments by professional users with high-cost devices. They are generally hard to be accessed by common users in their living environments. A recent contribution by Garrido et al. [2] allows users to reconstruct highly-detailed 3D facial models under uncontrolled lighting. While it has significantly simplified the requirements regarding on the capturing conditions, the demand of binocular stereo reconstruction for template face acquisition might still hinder its widespread applications by common users.

In this paper, we propose a non-parametric method to synthesize detailed wrinkle geometries and create personalized blendshape models *with a single low-cost Microsoft RGBD Kinect camera*, which are subsequently used to track RGBD facial performance videos to create 3D facial animations with detailed wrinkles (see Fig. 1). We utilize the texture synthesis approach to synthesize wrinkles on the 3D face expression model for various people. The distinctive feature of this method is *lightweight*, since we only use one high-quality 3D facial model with calibrated texture as the source in the texture synthesis and a single RGB-Depth camera. The key observation is that, although the facial wrinkles look different from one person to another, the variation of their local geometry is much less. This implies that it is possible

☆ This paper has been recommended for acceptance by Dr. Vadim Shapiro.
* Corresponding authors.
E-mail addresses: jun.johnson.li@gmail.com (J. Li), weiwei.xu.g@gmail.com (W. Xu), kevin.kai.xu@gmail.com (K. Xu).

http://dx.doi.org/10.1016/j.cad.2014.08.016
0010-4485/© 2014 Elsevier Ltd. All rights reserved.

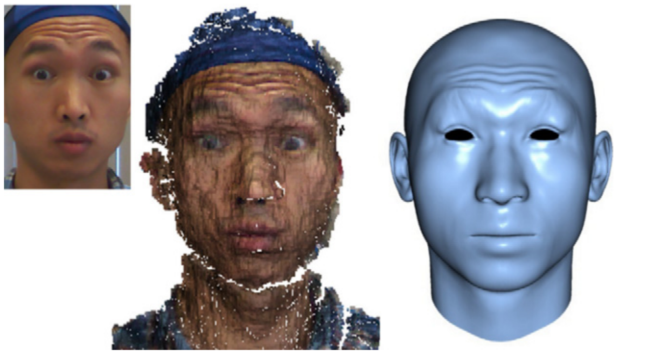


Fig. 1. Top-left: Captured RGB image. Middle: The corresponding Kinect raw depth data using the RGB image as its texture. Right: 3D mesh recovered with wrinkle details using our method.

to copy the local geometric patches of one source to synthesize different wrinkles of multiple subjects. Therefore, one can expect to synthesize different kinds of wrinkles in a lightweight fashion.

Our method works in two stages: offline personalized wrinkled blendshape construction and online 3D facial performance capturing. In the offline stage, the user-specific expressions are recorded as blendshapes, and the wrinkles on them are generated through example-based geometric detail synthesis. During the online stage, given an RGB-D video captured by a Kinect camera, the 3D facial animation with detailed wrinkles is reconstructed for each frame as the linear combination of the wrinkled blendshape models.

It should be noted that the synthesized wrinkles can only approximate the real wrinkle geometry. However, experiments show that our lightweight wrinkle synthesis method effectively guarantees the visual plausibility of the reconstructed detailed 3D facial models and animations.

2. Related work

3D facial modeling: The highly detailed 3D facial model is usually created via state-of-the-art 3D reconstruction techniques, such as laser scanning, structured light, multi-view video based systems.

XYZRGB [6] is a typical laser-based 3D scanning system. It takes over 10 s per scan which makes the system more suitable for capturing static expressions. Structured light systems are capable of capturing 3D dynamic faces with salient wrinkles in less than 0.1 s per scan. They are adopted in [7,8] to scan facial mesh animation sequences with detailed wrinkles.

In contrast to the mentioned active scanning systems, multi-view video-based facial capture systems are passive. They capture in fast speed, which is important to capture the fast occurring subtleties in the facial expressions. Markers are often used to assist the feature tracking in the facial performance video. Bickel et al. [1] explicitly tracked the wrinkles in the performance video and imposed them on the reconstructed facial model via linear shell deformation to create bulging effects.

Marker-less approaches can ease the burden of putting markers on the human face. Such approaches usually rely on multi-view stereo reconstruction with dense correspondences to recover the detailed wrinkles [3–5]. A lightweight solution, binocular stereo reconstruction, for marker-less facial animation is proposed in [2,9]. This technique is robust to illumination change because of the albedo estimation for the captured face.

Real-time facial animation: Facial performance video is a major type of input for real-time facial animation algorithms, since it is easy and fast to capture. The combination of marker tracking and fast deformation algorithms has already been used to produce facial animation in real-time [10]. Robust face tracking algorithms combined with linear blendshape models are also widely used in single camera based real-time facial animation [11,12,9,13]. To separately parameterize the different attributes of facial animation, a multi-linear model for face animation transfer is developed in [14]. However, the detailed wrinkles are usually missing in their reconstructed facial animations.

In [15], Huang et al. proposed to scan wrinkled 3D face models to create blendshape models, and successfully reconstruct the detailed wrinkles in the real-time facial animation. Bickel et al. proposed to learn a mapping from the sparse measurement facial skin strain to the wrinkle formulation to synthesize wrinkle geometries [16]. The blendshape models and training examples in these two methods are obtained through laser scanning and structured light, respectively. In contrast, our work allows users to create their own wrinkled blendshape models with a single RGBD camera, which is low cost and easy to access.

Example-based texture synthesis: It aims to synthesize a new texture appearing to be generated by the same process of the given exemplar. The 2D texture synthesis algorithms can be categorized into pixel-level algorithms [17,18], i.e. synthesize textures at each pixel, and patch-level algorithms, which can synthesize textures at a connected group of pixels simultaneously [19,20]. They are generally faster than pixel-based approaches.

We adapt the texture optimization algorithm in [21] to synthesize height values used to displace the face mesh vertices to generate wrinkles. While the individual components of our method may resemble previous approaches, but we are not aware of the existing work to synthesize wrinkles with non-parametric texture synthesis.

3. Wrinkle synthesis

To be robust to skin color and luminance change, our wrinkle synthesis runs in the gradient domain. Basically, there are three types of images used for both the input and the example face: the RGB image I, the gradient image G and the height image H, where the gradient image G stores $\partial I/\partial x$ and $\partial I/\partial y$ in its two channels and the height values are stored in H. Given an input image I_f, we first convert it into a gradient image G_f. Afterwards, we synthesize a gradient image G_s using the patches in the example gradient image G_e and compute a height image H_s using the correspondence between the example gradient image G_e and height image H_e. H_s is then used to displace the vertices on the face model to form wrinkles. The algorithm pipeline is shown in Fig. 2. In the following, we first describe how to generate the exemplars and then detail the wrinkle synthesis algorithm.

3.1. Exemplar extraction

Our exemplars for texture synthesis are extracted from one high quality face model M_e in the public facial performance sequence ETH man from [4], as shown in (a) of Fig. 3. The face model is accompanied with a high quality RGB texture T_e.

While the example gradient image G_e can be computed from T_e easily using the Sobel operator, the computation of height map H_e to encode the wrinkle geometry is not straightforward. To this end, we deform a smooth neutral face, which is also provided in this facial sequence, to match the selected expression model M_e and extract the height values to be stored in the H_e. The algorithm to compute H_e can be separated into two steps:

- 1. The non-rigid shape registration algorithm proposed in [22] is used to register a smooth neutral face to the selected expression. To avoid overfitting the detailed features in the deformation, we cluster the vertices according to the vertex normal variation on the neutral mesh. The deformation

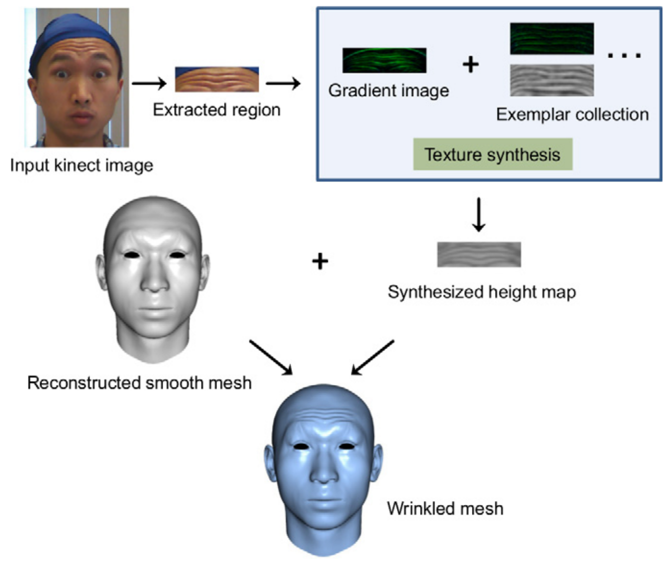


Fig. 2. The pipeline of our offline wrinkle synthesis method.

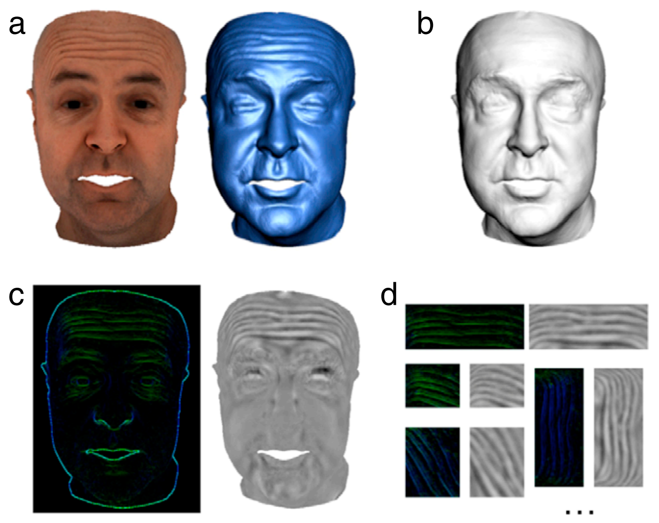


Fig. 3. (a) shows the texture T_e and 3D model M_e of the selected expression, (b) is the registered smooth mesh. (c) shows the corresponding gradient image G_e and the height map H_e. (d) shows the extracted exemplars.

variables are then reduced to rigid transformations for each cluster, which is similar to Huang et al. [22]. Since the neutral face and M_e share the same mesh connectivity, we can set up per-vertex correspondence and obtain a smooth registered mesh which successfully captures the overall facial expression but lacks the fine-scale wrinkles, as shown in Fig. 3(b).

- 2. After the registration, the fine-scale wrinkle information of M_e is extracted as its position difference between M_e and the registered smooth face by closest point searching. We compute the height values by projecting the difference vector into the normal direction at the searched closest point on the registered face. With the associated texture T_e, we can store height values into a height image H_e. Note that the correspondence between G_e and H_e can be well established using texture coordinates of M_e.

Since there are many areas in T_e without wrinkle information, we manually crop the wrinkle region on the forehead as the final G_e and H_e. The reduced search area in the exemplar is also beneficial to the performance of the texture synthesis algorithm. However, the wrinkles on the forehead are mostly horizontal. We thus compute new exemplars by rotation to significantly enlarge the wrinkle geometry variations. For example, we could rotate the face by 90° and obtain vertical wrinkle exemplars. In our experiments, we rotate the forehead wrinkles by 30°, 60°, 90°, 120° and 150°. Combined with the original forehead wrinkles, we collect six exemplars in total to be used in the wrinkle synthesis algorithm.

3.2. Salient wrinkle region detection

Before we perform the texture synthesis algorithm, we detect wrinkle edges from the input image. The texture synthesis is thus applied in a significantly smaller region to boost the performance. It also avoids adding unnecessary height displacement on non-wrinkle regions.

The detection is based on the observation that the gradient values have different signs on different sides of a wrinkle boundary. Specifically, in the implementation, we detect edges along both x and y directions. In x direction, for a pixel $p(x, y)$, if the gradient values in $q_1(x-1, y)$ and $q_2(x+1, y)$ have different signs and their difference is larger than a user-specified threshold ϵ, we mark this point as an edge point. We perform this operation along the y direction as well. From our experiments, we found that this strategy is more robust for wrinkle detection since it captures the characteristics of the buckling geometry of wrinkles.

After the edge point detection, we group them into lines through tracing them based on adjacency. We remove very short lines and the lines around mouth, lips, eye brows and nose using the feature points detection in [23], since these lines are usually formed by color or lighting changes on smooth regions. The remaining lines are further clustered to line sets according to spatial adjacency. The 2D bounding box of each line set represents a wrinkle region. For some cases, automatic detection may not work well, for instance, the small wrinkle regions around eye corners. We thus allow user interaction to extract the wrinkle region more accurately. The user interaction is only necessary for small regions around the eyes. We only ask the user to draw several rectangles to indicate the small regions, and it takes no more than 30 s.

3.3. Texture synthesis

For texture synthesis, we require the synthesized gradient image G_s to be locally similar to the example gradient image G_e and globally almost identical to the input gradient image G_f. That is, we hope to use the local patches from G_e to form G_s which are similar to G_f. The synthesized height image H_s is calculated by copying height values from example height image H_e to H_s with the closest neighborhood information between G_e and G_s. However, if we only consider the similarity of the gradient images, the resulting height map might not be smooth, since the similarity in gradient values does not directly imply the smoothness of the synthesized height values. We thus adapt the objective function in the texture optimization technique in [21] according to:

$$\mathbf{E} = \sum_{p \in G_s} \|\mathbf{g}_p - \mathbf{g}'_p\|^2 + \lambda \|\mathbf{h}_p - \mathbf{h}_q\|^2 \quad (1)$$

where the first term measures the similarity between G_s and G_e. $\mathbf{g}_p$ is a vector of stacked gradient values from a small size window centered at p in G_s, and $\mathbf{g}'_p$ denotes a stacked vector of gradient values at the same pixel p from G_e. The second term measures the similarity between the formed height images H_s and H_e locally, where $\mathbf{h}_p$ is the vector of stacked height values at pixel p from H_s, and $\mathbf{h}_q$ represents the vector of stacked height values at pixel q from H_e. q indicates the closest pixel in G_e to the pixel p found in the texture optimization procedure using their neighborhood information.

Texture optimization can be achieved iteratively with an algorithm similar to Expectation–Maximization (EM) [21]. The key difference of our optimization algorithm is to set the initial gradient image directly as G_f. In the E step, our algorithm calculates the average gradient and the height value at each pixel p according to the neighborhood information of its closest pixel q in the example gradient G_e and height image H_e.

For the M step, where the closest neighborhood is searched in the example, we fix the gradient values at each input pixel same to the G_f. This prevents the algorithm from altering the synthesized gradient image iteratively, which will lead to a severe drift from the input gradient image in the final result. Specifically, we search the nearest neighborhood in G_e and H_e using the combined gradient and height distance as defined in Eq. (1), but using the input gradient values and the synthesized height values. As the scale of the height values in H_e (we use mm as unit) is different from the gradient values, we weight the channel H by λ (usually we set λ to 10) to keep a balance between spatial coherence of the height map and visual appearance similarity with G_f.

Wrinkle geometry generation: We then displace the vertices on the 3D face expression model along their normals to form wrinkle details according to the synthesized height image H_e. Note that we only keep the height values in the neighborhood around the detected wrinkle regions (see Section 3.2) and set others to be zero. In our experiments, the neighborhood size is set to 5 pixels. After displacing the vertex, we apply Laplacian smooth operations to further improve the mesh quality. It also constructs smooth transitions between the wrinkled and non-wrinkled regions.

4. Application to real-time facial animation

Our wrinkle synthesis method allows users to use a low-cost Microsoft Kinect to generate personalized blendshape models, which is also a linear model to reinforce the captured facial animation with detailed wrinkles.

In our implementation, we usually capture twenty expressions for each user, such as *neutral pose, smile, brow lower, brow raiser, jaw left, jaw right, jaw forward, mouth left, mouth right, mouth open, cheek blowing, lip funnel, lip pucker, grin and chin raiser*. More complex expressions are required to be captured for the user if higher quality is demanded. While the number of captured blendshape models is less than the number required in FACS system [24], we found the captured user-specific expressions in the blendshape model are sufficient to reconstruct the facial animation with detailed wrinkles.

As the depth data from Kinect camera is noisy, we adopt a method similar to [25] to create high-quality expression models. The user needs to rotate her/his head slightly while keeping the expression unchanged in front of the camera. The captured multiple depth images are registered into the first scan by fast ICP, and finally merged into one smooth point cloud [25]. The first scan is required to be of front view, and we also captured its corresponding RGB image as the input of our wrinkle synthesis algorithm. We then apply the morphable face model [26] to the point cloud data captured for the neutral pose, resulting in a smooth 3D neutral face mesh. This neutral mesh is used as template and warped to each captured expressions using the non-rigid registration approach of [22]. These reconstructed expression meshes share the same topology.

4.1. Online facial performance capture

The reconstructed blendshape models can be directly used in real-time performance-driven facial animation. The overall procedure consists of two steps: rigid motion tracking to compute the global rigid transformation of the head and non-rigid motion tracking to reconstruct the local deformations at each expression.

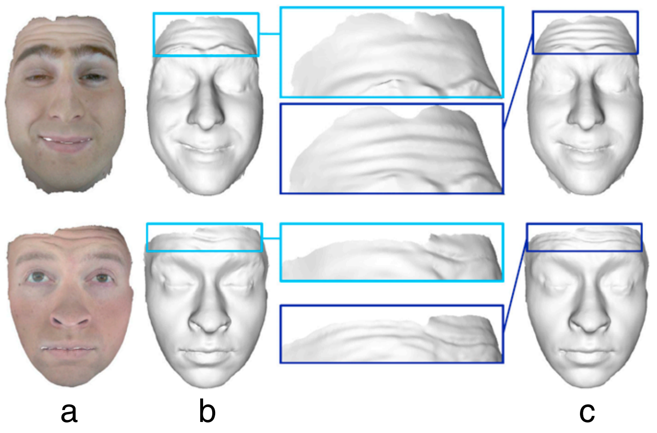


Fig. 4. Comparison with Bradley et al. [5]. (a) The original captured images. (b) The results from [5]. (c) Wrinkle details generated by our method.

For rigid motion tracking, we utilize the fast projective variant of ICP [27] based on point-to-plane constraints. However, due to the noisy Kinect data, the resulting rigid tracking is not temporally smooth. To filter out high frequency flickering, we first smooth the raw data from Kinect temporally, i.e. a weighted average on the depth map for k (default 5) consecutive frames. We also filter the computed rotations (represented by quaternion) and translations to further improve the tracking quality [25].

The purpose of non-rigid tracking is to determine the weights of the blendshape models and reconstruct the user expressions in each frame, which can be formulated into a non-negative least squares problem:

$$\arg\min_{\alpha_i} \|F(\alpha_1 \ldots \alpha_n)^t - f_i\|^2 \tag{2}$$

where $F(\alpha_1 \ldots \alpha_n)^t$ denotes the features on the shape determined by current weights $\{\alpha_i\}$, and f_i are the corresponding captured features. These features could be 3D sample points or 2D sparse features detected from RGB images. Finally, the tracked wrinkled mesh could be recovered by:

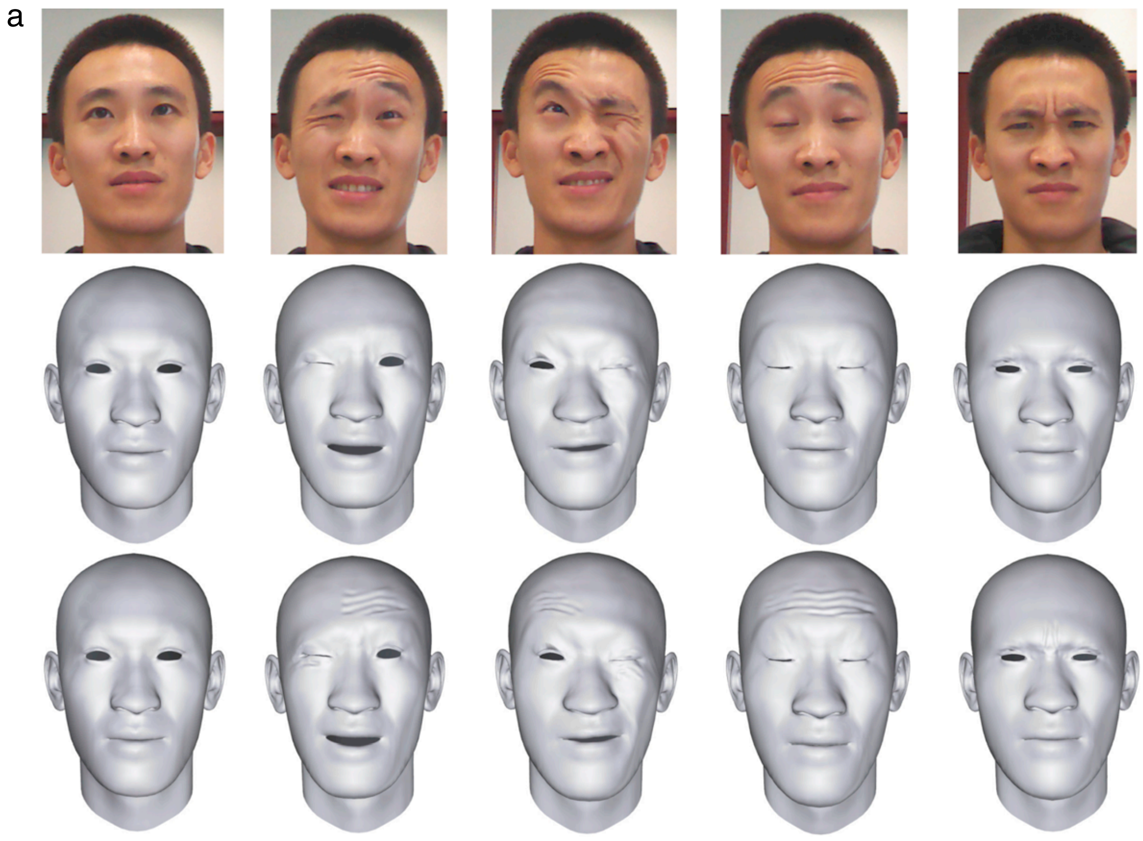
$$\mathbf{E} = \mathbf{N} + \sum_i \alpha_i B_i \tag{3}$$

where $\mathbf{N}$ represents the 3D neutral model and B_i the i-th blendshape model.

5. Results

In this section, we validate our method in the generation of 3D facial models with detailed wrinkles and real-time facial animations of various people using the Microsoft Kinect camera. The current facial performance capture system runs in real-time on a machine with Intel Core 2.66 GHz Quad CPU and GeForce GTX 580 graphics card. The average computational time of the wrinkle synthesis for each captured expression is around 10 s, and the cost of online facial performance with wrinkles is about 20 ms for one frame in average.

Comparison with multi-view stereo: Fig. 4 illustrates a comparison between our results and the passive facial capture method [5]. Although the reconstructed wrinkle details from [5] are smoother than our synthesized wrinkle geometry, our synthesis results are visually very similar and provide high-quality clues of the input user expressions.



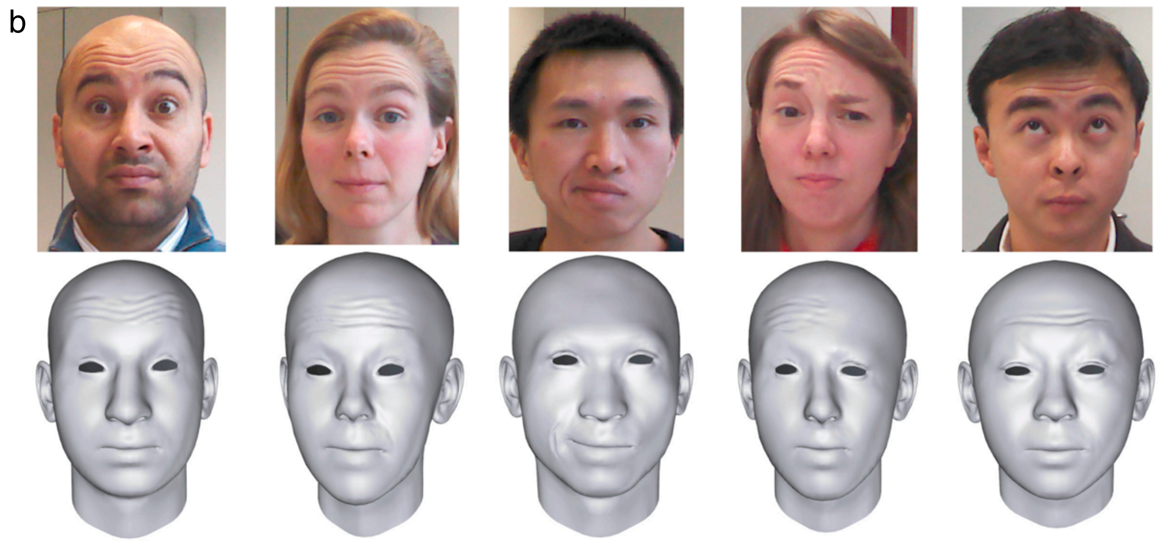


Fig. 5. Wrinkle synthesis results using our method.

Testing on Kinect data: We test our method on various captured RGB-D images and videos from different people to demonstrate the ability of our system to produce plausible 3D facial performances with detailed wrinkles.

In Fig. 5(a) are the wrinkle synthesis results on a few expressions captured the Kinect sensor. The first row is the RGB images, the second row lists the meshes before wrinkle details were added and the last row shows the corresponding wrinkled mesh. Note the neutral pose is shown in the left column. In Fig. 5(b) are more examples of face models with synthesized wrinkles using our method. Fig. 6 shows some sample frames from online facial performance with well reconstructed wrinkle details. For the left part, from top to bottom are RGB images, Kinect point clouds and the reconstructed facial models with detailed wrinkles. The close-up view in the right column shows the quality of synthesized wrinkles from our system. As our wrinkle generation depends on the coefficients of blendshapes, it is robust to light changing, hair occlusion and head rotation during online tracking.

Limitation: Our wrinkle synthesis algorithm now is limited to front view facial images, and it might be influenced by shadows cast by hairs on the face. We plan to explore facial image normalization and shadow removal algorithms to further improve the range of facial images that can be handled by our wrinkle synthesis algorithm.

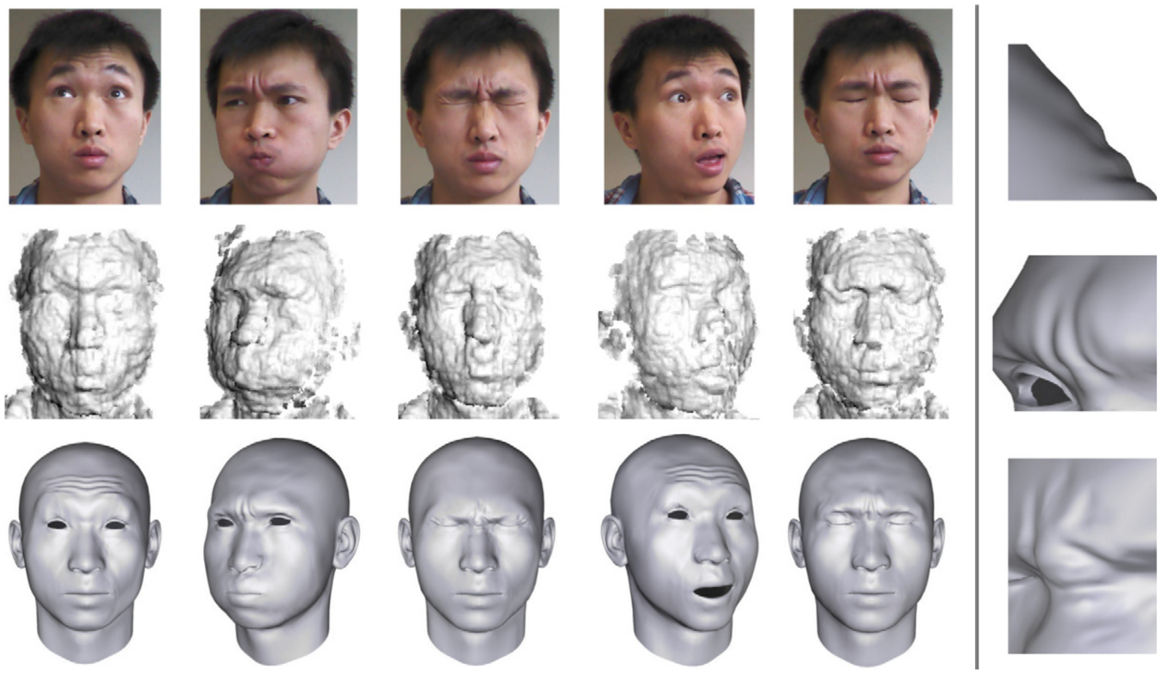


Fig. 6. Online facial performance with recovered wrinkle details.

6. Conclusions

We have presented a lightweight wrinkle synthesis method to enhance the reconstructed 3D face models with detailed wrinkles. Our method is lightweight, because we only use a single Microsoft Kinect camera plus one selected, high quality wrinkled 3D face model to generate all the results in this paper.

In future, we plan to enlarge the database so as to incorporate different styles of wrinkle geometry images from people of different ages and races. In wrinkle synthesis, images similar to the capture data will be first retrieved for the wrinkle synthesis. This would lead to higher quality of synthesized results with more enriched expressions captured by our system. We would also like to explore the fast texture synthesis method to reduce the wrinkle generation time.

Acknowledgments

We thank Thomas Vetter for kindly providing the Morphable model and Hao Li for the generic blendshape model. The research leading to these results was supported in part by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 323567 (Harvest4D) 2013-2016, NSFC (61322204, 61272392, 61103084, 61272334, 61202333), CPSF (2012M520392) and grants from China Scholarship Council.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.cad.2014.08.016.

References

[1] Bickel B, Botsch M, Angst R, Matusik W, Otaduy M, Pfister H, et al. Multi-scale capture of facial geometry and motion. ACM Trans Graph (SIGGRAPH) 2007; 26(3).
[2] Garrido P, Valgaerts L, Wu C, Theobalt C. Reconstructing detailed dynamic face geometry from monocular video. ACM Trans Graph (SIGGRAPH ASIA) 2013; 32: 158:1–158:10.
[3] Beeler T, Bickel B, Beardsley P, Sumner B, Gross M. High-quality single-shot capture of facial geometry. ACM Trans Graph (SIGGRAPH) 2010;29(4): 40:1–40:9.
[4] Beeler T, Hahn F, Bradley D, Bickel B, Beardsley P, Gotsman C, et al. High-quality passive facial performance capture using anchor frames. ACM Trans Graph (SIGGRAPH) 2011;30(4): 75:1–75:10.
[5] Bradley D, Heidrich W, Popa T, Sheffer A. High resolution passive facial performance capture. ACM Trans Graph (SIGGRAPH) 2010;29(4): 41:1–41:10.
[6] XYZRGB. XYZ RGB system, 2012. http://www.xyzrgb.com.
[7] Ma W-C, Jones A, Chiang J-Y, Hawkins T, Frederiksen S, Peers P, et al. Facial performance synthesis using deformation-driven polynomial displacement maps. ACM Trans Graph (SIGGRAPH ASIA) 2008;27(5): 121:1–121:10.
[8] Zhang L, Snavely N, Curless B, Seitz SM. Spacetime faces: high resolution capture for modeling and animation. ACM Trans Graph (SIGGRAPH) 2004; 23(3):548–58.
[9] Valgaerts L, Wu C, Bruhn A, Seidel H-P, Theobalt C. Lightweight binocular facial performance capture under uncontrolled lighting. ACM Trans Graph (SIGGRAPH ASIA) 2012;31(6): 187:1–187:11.
[10] Vicon. Vicon homepage, 2012. http://www.vicon.com/.
[11] Cao C, Weng Y, Lin S, Zhou K. 3D shape regression for real-time facial animation. ACM Trans Graph (SIGGRAPH) 2013;32(4): 41:1–41:10.
[12] Li H, Yu J, Ye Y, Bregler C. Realtime facial animation with on-the-fly correctives. ACM Trans Graph 2013;32(4): 42:1–42:9.
[13] Weise T, Li H, Van Gool L, Pauly M. Face/off: live facial puppetry.
[14] Vlasic D, Brand M, Pfister H, Popović J. Face transfer with multilinear models. ACM Trans Graph 2005;24(3):426–33.
[15] Huang H, Chai J, Tong X, Wu H-T. Leveraging motion capture and 3D scanning for high-fidelity facial performance acquisition. ACM Trans Graph (SIGGRAPH) 2011;30(4): 74:1–74:10.
[16] Bickel B, Lang M, Botsch M, Otaduy MA, Gross M. Pose-space animation and transfer of facial details. In: SCA'08. 2008. p. 57–66.
[17] Efros AA, Leung TK. Texture synthesis by non-parametric sampling. ICCV, vol. 2. 1999. p. 1033–8.
[18] Wei L-Y, Levoy M. Fast texture synthesis using tree-structured vector quantization. In: Proceedings of the 27th annual conference on computer graphics and interactive techniques.
[19] Efros AA, Freeman WT. Image quilting for texture synthesis and transfer. In: SIGGRAPH'01. ACM Press; 2001. p. 341–6.
[20] Liang L, Liu C, Xu Y-Q, Guo B, Shum H-Y. Real-time texture synthesis by patch-based sampling. ACM Trans Graph 2001;20(3):127–50.
[21] Kwatra V, Essa I, Bobick A, Kwatra N. Texture optimization for example-based synthesis. ACM Trans Graph 2005;24(3):795–802.
[22] Huang Q, Adams B, Wicke M, Leonidas JG. Non-rigid registration under isometric deformations. In: Proceedings of the symposium on geometry processing. Eurographics Association; 2008. p. 1449–57.
[23] Saragih SJ, Cohn J. Real-time avatar animation from a single image. In: IEEE International conference on automatic face and gesture recognition. Santa Barbara (CA, USA): IEEE Computer Society; 2011. p. 117–24.
[24] Ekman P, Friesen W. Facial action coding system: a technique for the measurement of facial movement. Consulting Psychologists Press; 1978.
[25] Weise T, Bouaziz S, Li H, Pauly M. Realtime performance-based facial animation. ACM Trans Graph (SIGGRAPH) 2011;30(4): 77:1–77:10.
[26] Blanz V, Vetter T. A morphable model for the synthesis of 3D faces. In: Proceedings of SIGGRAPH'00.
[27] Rusinkiewicz S, Levoy M. Efficient variants of the ICP algorithm. In: International conference on 3D digital imaging and modeling. IEEE Computer Society; 2001. p. 145–52.