KinectFusion for Faces: Real-Time 3D Face Tracking and Modeling Using a Kinect Camera for a Markerless AR System

Márcio C. F. Macedo*†, Antônio L. Apolinário Jr.*, Antonio C. S. Souza†
*Department of Computer Science
Federal University of Bahia (UFBA)
40170-970 - Salvador - BA - Brazil
Email: marciocfm@dcc.ufba.br, apolinario@dcc.ufba.br
†Laboratório de Realidade Aumentada, Jogos Digitais e Dispositivos Móveis (LABRAGAMES)
Department of Electro-Electronic Technology
Federal Institute of Bahia (IFBA)
40301-015 - Salvador - BA - Brazil
Email: antoniocarlos@ifba.edu.br

Abstract—**In this paper we present an extension to the KinectFusion algorithm that allows a robust real-time face tracking and modeling using the Microsoft's Kinect sensor. This is achieved changing two steps of the original algorithm: pre-processing and tracking. In the former, we use a real-time face detection algorithm to segment the face from the rest of the image. In the latter, we use a real-time head pose estimation to give a new initial guess to the Iterative Closest Point (ICP) algorithm when it fails and an algorithm to solve occlusion.**

Our approach is evaluated in a markerless augmented reality (MAR) system. We show that this approach can reconstruct faces and handle more face pose changes and variations than the original KinectFusion's tracking algorithm. In addition, we show that the realism of the system is enhanced as we solve the occlusion problem efficiently at shader level.

Index Terms—**Augmented Reality; Head Pose Estimation; Face Modeling.**

I. Introduction

Augmented reality (AR) is a technology in which a user's view of a real scene is augmented with additional virtual information. Accurate tracking, or camera pose estimation, is required for the proper registration of virtual objects. However, tracking is one of the main technical challenges of AR.

In some AR systems, the user turns his head in front of a camera and the head is augmented with a virtual object. In this case, is desirable an algorithm able to track the person's head with enough accuracy and in real-time. One way to achieve this goal is building a reference 3D model of the user's head and aligning it to the current head captured by the sensor. However, this process (i.e. face modeling) often has high computational costs and needs manual post-processing.

With the recent advances in the field of 3D reconstruction, nowadays it is possible to reconstruct high-detailed models in real-time exploiting the power of the graphics processing units (GPUs) [1]. The user can achieve this goal by turning an object in front of a 3D scanner or by turning a 3D scanner around an object. It can be easily extended to faces if we make the assumption that the user will turn his head in front of the scanner with a neutral expression and as rigidly as possible, as in [2].

We present an approach for robust real-time face tracking and modeling using the Microsoft's Kinect sensor for a markerless augmented reality (MAR) system. First, we apply the Viola-Jones face detector [3] to locate and segment the face in the whole image. Afterward, a reference 3D model is built with a real-time 3D reconstruction system. Next, the Kinect raw data is aligned to the reference 3D model, predicting the current camera pose. Finally, to improve the robustness of the system, is used a head pose estimator to give an initial guess to the tracking algorithm when it fails. An overview of this method can be seen in Figure 1. In addition, we enhance the realism of the system solving the occlusion problem between the virtual and real objects.

The method is inspired by three notable works: An algorithm that allows the dense mapping of extended scale environments in real-time using only Kinect raw data called KinectFusion [1]; an algorithm for estimating the location and orientation of a person's head from low quality depth data [4] and an algorithm for detecting faces in real-time [3]. Our approach adapts the KinectFusion to the face modeling and extends its tracking using the head pose estimation. We show that this approach can reconstruct faces and handle more face pose changes and variations than the original KinectFusion's tracking algorithm. This approach is evaluated in a MAR system.

The rest of the paper is arranged as follows. Section 2 provides a brief review on the related work of surface reconstruction, real-time face modeling, markerless AR and real-time head pose estimation. Section 3 presents the proposed algorithm. Section 4 discusses the experimental results. The paper concludes in Section 5, with a summary and discussion

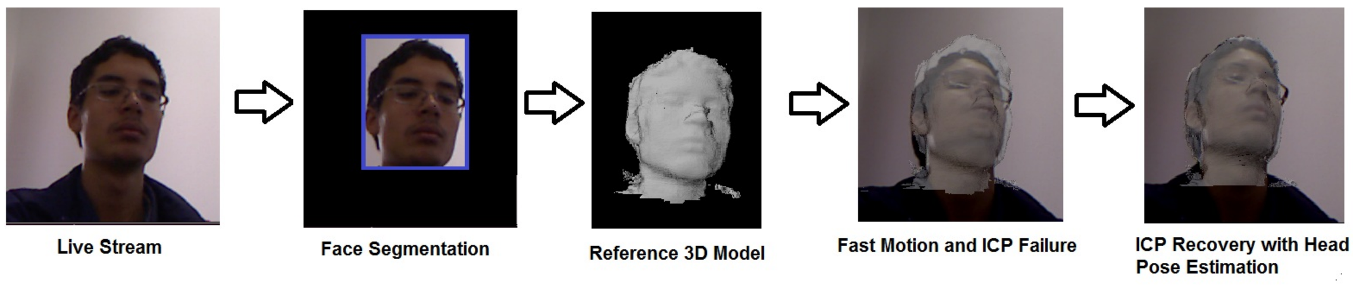
Live Stream
Face Segmentation
Reference 3D Model
Fast Motion and ICP Failure
ICP Recovery with Head Pose Estimation

Fig. 1. Overview of the online processing pipeline. A) RGB-D live stream. B) A face detector is used to locate and segment the face from the rest of the scene. C) Reference 3D model is reconstructed with KinectFusion. D) In a MAR system, the user rotated his face fast and the ICP failed. E) The head pose estimation is used to give an initial guess to the ICP and the fast motion is compensated.

of future work.

II. Related Work

Surface reconstruction, face modeling, markerless AR and head pose estimation have been driven by different approaches, as we can see in the next subsections (a full review of face modeling is beyond the scope of this paper and we refer to [5] for a more detailed discussion).

Surface reconstruction: In 1996, Curless and Levoy [6] described a method for volumetric integration of complex models from range images (VRIP). The volumetric integration basically consists of a cumulative weighted signed distance function (SDF). This method is able to integrate high-detail models, in the order of a million triangles. However, the execution time can be in the order of hours and it is not suitable for AR applications. The range images used in this work were captured by laser scanners. Laser scanners provide range images with high accuracy, but the drawback of them is the high cost of the hardware.

In 2002, Rusinkiewicz et al. [7] described a method for real-time 3D model acquisition. Using a real-time low-quality structured-light 3D scanner, they aligned the range images from different viewpoints to produce complete 3D rigid objects. Different from the method proposed by Curless and Levoy, it operated at $\approx$ 10 Hz with lower cost hardware but did not reconstruct high-quality models. It was the first system to reconstruct and display the 3D models in real-time and it increased the possibility to do markerless AR with surface reconstruction.

In 2010, Cui et al. [8] described a method for 3D object scanning using a time-of-flight (ToF) camera. In this work, Cui et al. showed a superresolution method that improves significantly the quality of the depth maps acquired from a ToF camera. One drawback of this method is that it does not run in real-time. Compared to the other scanners presented, time-of-flight cameras have the lowest cost and provide range images with the lowest accuracy.

Markerless AR: In 1999, in the field of AR, Kato and Billinghurst [9] presented a video-based AR system with marker tracking which mixed virtual images on the real world. They used fast and accurate computer vision techniques to track the fiducial markers through the video. The system presented is also called ARToolKit and it is one of the most used systems in this field.

In 2000, Simon et al [10] described one of the first methods using markerless tracking for an AR system: a tracker of planar structures. Despite being a special case of tracking (i.e. when there is a planar structure visible in the scene), the method does not need fiducial markers and robustly tracks the planar structures through the video.

Real-Time Face Modeling + Markerless AR: Izadi et al. [1] described a system that enables real-time detailed 3D reconstruction of a scene using the depth stream from a Kinect. The system was called KinectFusion. Using a GPU, it was the first system to reconstruct high-detail models at $\approx$ 30Hz. Izadi et al. [1] also presented some markerless AR applications, showing the level of the user interaction in their system. This method was originally developed to reconstruct large scale scenes but it can be applied for face modeling.

In the same year, Weise et al. [11] presented a system that enables active control of facial expressions of a digital avatar in real-time. The system is called FaceShift [12]. It was the first system to enable high-quality reconstruction and control of facial expressions using blendshape representation in real-time. FaceShift represents a great advance in the field of MAR and non-rigid surface reconstruction.

An adaptation of the KinectFusion for face modeling was done in [13]. It reconstructs high-quality face models by representing the face in cylindrical coordinates and by applying temporal and spatial smoothing on the 3D face shape. We present an alternative and simpler method to adapt the original KinectFusion to reconstruct faces properly and in real-time.

Head Pose Estimation: Recently, automatic real-time 3D head pose estimation have become popular due to the increasing availability of the 3D scanners.

Breitenstein et al. [14] developed a real-time algorithm to estimate 3D head pose using GPUs. Using high-quality depth data, the algorithm computes a set of candidate nose positions and compares the input depth data to precomputed pose images of an average face model.

Fanelli et al. [4] developed a real-time algorithm to estimate 3D head pose using only the CPU. Using low-quality depth data (e.g. captured from a Kinect sensor), the algorithm trains random forests to estimate head pose.

We choose this last algorithm for head pose estimation because it operates directly on low-quality depth data.

III. KinectFusion for Faces

In this section we describe the proposed improvements we made to the KinectFusion to track and reconstruct faces. Before, we describe the original KinectFusion, the head pose estimation and the face detector used.

A. Reconstructing 3D Models with KinectFusion

KinectFusion [1] is a system that integrates raw depth data from a Kinect camera into a voxel grid to produce a high-quality 3D reconstruction of a scene.

The system first applies a bilateral filter [15] to the depth map to reduce the noise preserving discontinuities of the raw data. The filtered depth map is then converted into a vertex map and a normal map in the camera's coordinate space. It is done by the product between the filtered depth map and the Kinect infrared camera's intrinsic calibration matrix.

To compute the transformation that defines the camera pose is used a real-time variant of the well known ICP (Iterative Closest Point) algorithm [16]. The ICP estimates the transformation that aligns the current depth frame with the accumulated model. It consists of six stages:

- **Selection of points**: All the points visible (depth greater than 0) are selected;
- **Matching of points**: It is used the projective data association [17] that is described in more details in the Subsection III-D;
- **Weighting of pairs**: It is assigned constant weight to each association;
- **Rejecting pairs**: Pairs are rejected if the Euclidean distance or angle between the points are greater than some user-defined threshold;
- **Error metric**: The error is defined by the sum of squared distances from each point in the current frame to the tangent plane at its corresponding point in the accumulated model (point-to-plane error metric) [18];
- **Error minimization**: The error is minimized using a Cholesky decomposition on a linear system.

Once with the current transformation, the raw depth data can be integrated into the voxel grid. The grid stores at each voxel the distance to the closest surface (SDF) [19] and a weight that indicates uncertainty of the surface measurement. The SDF values are positive in-front of the surface, negative behind and zero-crossing where the sign changes. In the KinectFusion, the SDF is only stored at a truncated region around the surface. This distance is also referred as a truncated signed distance function (TSDF). These volumetric representation and integration are based on the well known VRIP algorithm [6].

Surface extraction is achieved by detecting zero-crossings through a raycaster. All these operations are made using the GPU. An overview of this method can be seen in Figure 2.

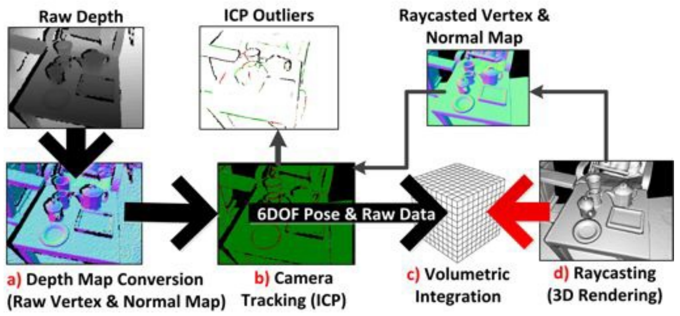
Raw Depth
ICP Outliers
Raycasted Vertex & Normal Map
6DOF Pose & Raw Data
a) Depth Map Conversion (Raw Vertex & Normal Map)
b) Camera Tracking (ICP)
c) Volumetric Integration
d) Raycasting (3D Rendering)

Fig. 2. Overview of KinectFusion's pipeline [1].

B. Real-Time Head Pose Estimation from Consumer Depth Cameras using Random Regression Forests

Random Regression Forests are trees trained randomly that generalize a problem better than decision trees taken separately [20]. Fanelli et al. [4] trained random forests to estimate head pose from low-quality depth images. To train the trees, each depth map was annotated with labels indicating head center and Euler rotation angles. These labels were estimated automatically using ICP after a 3D facial reconstruction. After the labeling and training, the head pose can be estimated letting every image region to vote it. The vote consists of a classification whether the image region contains a head and a retrieval of a Gaussian distribution computed during the training and stored at the leaf. This probabilistic approach achieves high accuracy and runs in real-time using only CPU.

C. Robust Real-Time Face Detection based on Haar-like Features

Viola and Jones [3] described a method for robust real-time face detection in color images. They used a representation called integral image to compute Haar-like features quickly (i.e. each pixel contains the sum of the pixels above and to the left of the original position) and a combination of simple classifiers built using the Adaboost learning algorithm [21] to detect the face regions.

D. Our Approach

Our approach consists of two main stages: head reconstruction and markerless AR face tracking. The first stage consists in the application of KinectFusion to reconstruct the user's head (Figure 3) and the second stage consists in tracking of the user's face augmented with a virtual object. As mentioned before, we extended the original KinectFusion algorithm in two steps: preprocessing and tracking. We also use our tracking solution in both stages of the system.

For each new depth frame D, we segment the region of interest (i.e. user's head) by applying a Z-axis threshold of $1.3m$. $1.3m$ was chosen because it is the maximum acceptable distance from the user's head to the camera center in the original Fanelli's head pose estimation [4].

Next, we apply the Viola-Jones face detector [3] to locate and segment the face in the color image. Face detection is only performed during head reconstruction. Once the face is

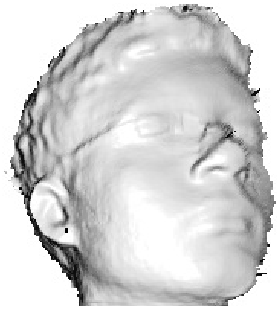

Fig. 3. An example of user's head reconstructed with the KinectFusion and rendered with Phong shading [22].

Algorithm 1 Use of the head pose estimation

```
1: estimate head pose of $D_{prev}$.
2: $R_{prev} \leftarrow$ extract rotation matrix estimated from $D_{prev}$.
3: $Hc_{prev} \leftarrow$ extract global head center from $D_{prev}$.
4: estimate head pose of the $D_{curr}$.
5: $R_{curr} \leftarrow$ extract rotation matrix estimated from $D_{curr}$.
6: $Hc_{curr} \leftarrow$ extract global head center from $D_{curr}$.
7: $R_{inc} \leftarrow R_{curr} * R_{prev}^{-1}$.
8: $\Delta t \leftarrow Hc_{prev} - Hc_{curr}$.
9: $t \leftarrow t + \Delta t$.
10: raycast the implicit surface to generate a new view.
11: rotate the raycasted view around $Hc_{curr}$ with $R_{inc}$.
```

detected, we fix the window that contains the face region. Then, the user is constrained to translate and rotate his face in that window (an example can be seen in Figure 1-B). With this simple method, we can reconstruct the user's head in real-time.

After (or during) that, we apply the ICP algorithm to compute the current camera pose (i.e. transformation matrix T). The ICP uses the projective data association [17] to find correspondences between the current depth frame and the accumulated model. In this association, each point is transformed into camera coordinate space and perspective projected into image coordinates. The corresponding points are that on the same image coordinates. The ICP fails (i.e. does not converge to a correct alignment) when there is not a small pose variation between sequential frames. We detect it by checking if the linear system computed is solvable (i.e. the matrix is invertible). If the ICP fails, we use the head pose estimation to give a new initial guess to the ICP to compute correctly the current transformation.

The use of the head pose estimation is shown in **Algorithm 1**. Given the previous depth frame D_{prev} and the current depth frame D_{curr}, the head pose estimation is used to set the head orientation (R_{prev} and R_{curr}) and the head center (Hc_{prev} and Hc_{curr}) of them. The head centers are converted from camera to global coordinates. The incremental rotation matrix R_{inc} and the translation Δt between the previous and the current head center are computed (lines 7 and 8). The translation Δt is added to the current global translation t (line 9). The implicit surface is then raycasted to generate a new view (i.e. new previous depth frame) (line 10). The raycasted view is rotated around Hc_{curr} with R_{inc} (line 11). Finally, we reuse the ICP to estimate the current T.

Our approach also solves the occlusion problem by using a GLSL fragment shader [23] that compares the depth value of the virtual (i.e. reconstructed head model) and real (i.e. bilateral filtered depth data) objects to check whether the virtual object is in front of the real object, and vice-versa.

IV. Results and Discussion

In this section we analyze the system's performance and describe the experimental setups we used.

We based our system on the open source C++ implementation of the KinectFusion [24] released by the PCL project [25] and on the open source C++ implementation of the head pose estimation released by Fanelli [26]. For all tests we ran our system on an Intel(R) Core(TM) i7-3770K CPU @3.50GHz 8GB RAM in real-time. When the head pose estimation was used, the main pipeline of our system needed only 80ms to process a frame. Without the head pose estimation, the main pipeline needed only 40ms.

We tested our algorithm with real data captured with a Kinect sensor using a grid with volume size of 50cmx50cmx130cm that could reconstruct high-quality heads. We can analyze the qualitative performance for three cases: fast translation and rotation of the user's face and variation of illumination condition. An example of application of our approach can be seen in Figure 4.

When the user translated his face in front of the camera and the ICP failed, the algorithm could give a correct initial guess to the ICP. If the user translates his face fast, there will not be sufficient points at the same image coordinates and the ICP will fail. By applying our approach we can solve this problem. This situation can be seen in Figure 5.

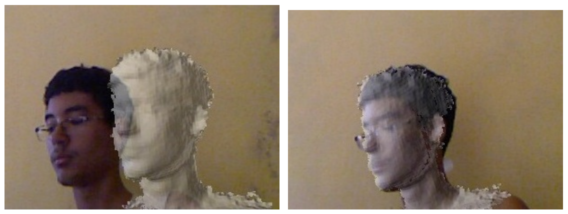

Fig. 5. A) The user translated his face fast. A small number of points were at the same image coordinates and the ICP failed. B) By applying our approach we solved this problem.

The algorithm slightly improved the tracking performance when the user rotated his face and the ICP failed. The reason is that the larger the pose variation, the larger the non-overlapping region, and there are cases that the ICP is not appropriate in the presence of non-overlapping regions (Figure 1, D and E) even if the head pose estimation provides the initial guess. In this case (Figure 6), the user needs to reposition his face to the tracking algorithm to align correctly the raw depth data. One can use non real-time up-to-date algorithms, as the Sparse ICP [27], to solve this problem.

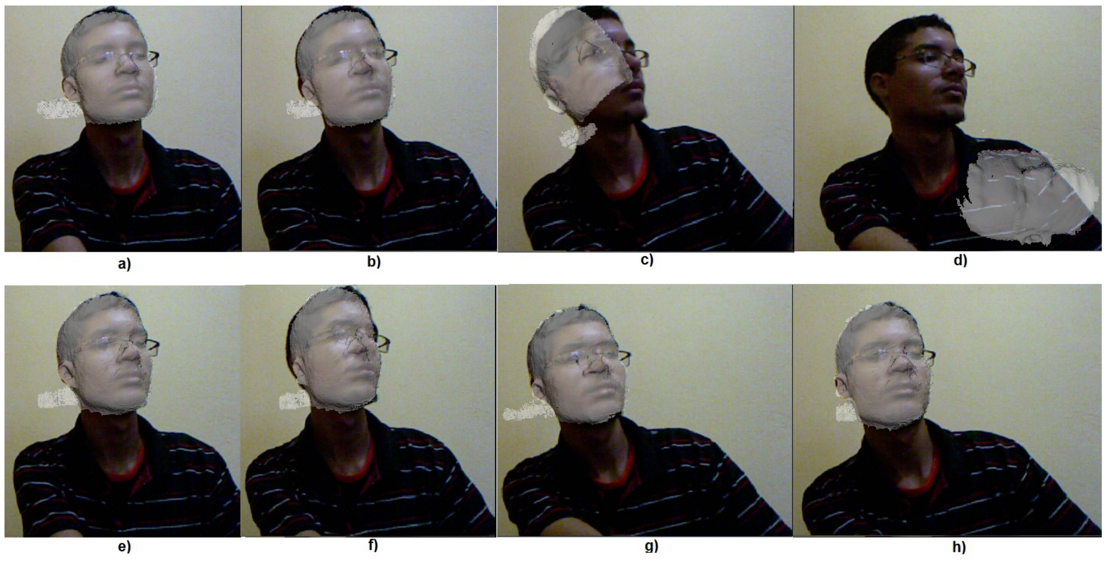

Fig. 4. Tracking with ICP: a) The tracking is started. b) The user moves his face in front of the Kinect and the motion is not totally compensated by the algorithm. c) The ICP fails d) completely.
Tracking with ICP + Head Pose Estimation: e) The tracking is started. f) The user moves his face in front of the Kinect and the motion is not totally compensated by the algorithm. g), h) The ICP fails but it recovers with the initial guess provided by the head pose estimation.

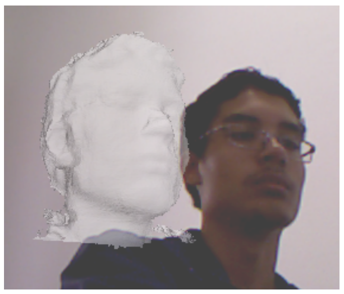

Fig. 6. An example of tracking failure. The user needs to reposition his face to the tracking algorithm align correctly the raw depth data to the reference 3D model.

The face detector was robust even in presence of low and high illumination (Figure 7), allowing the system to reconstruct faces under different illumination conditions.

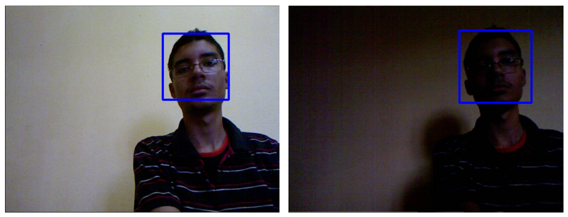

Fig. 7. A) The face was detected correctly in presence of high and B) low illumination.

As mentioned earlier, our approach also supports occlusion (Figure 8). However, in our case where we are comparing a reconstructed model against noisy data, the presence of holes and the partial occlusion of the real object can reduce the occlusion and tracking accuracies, as stated in [28].

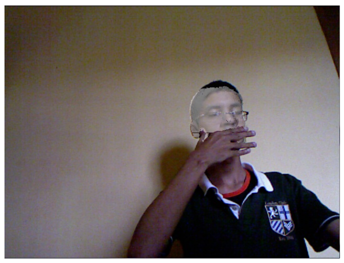

Fig. 8. Occlusion is solved by using a fragment shader that checks whether a virtual object is in front of a real object.

The accuracy of the head pose estimation is the same as the Fanelli's approach (angle error: about 8^o in each axis; head center error: $10mm$). However, as mentioned before, in the case of large pose variations, its initial guess is not sufficient for the ICP algorithm.

V. Conclusions and Future Work

We have presented the KinectFusion for Faces: an approach for real-time face tracking and modeling using a Kinect camera for a markerless AR system. We used the KinectFusion to reconstruct the user's head and we extended its tracking algorithm using the head pose estimation to give the initial guess to the ICP algorithm when it failed. Also, we have solved the occlusion problem to enhance the realist of the system. We have shown that this approach can reconstruct faces and handle more face pose changes than the original KinectFusion's tracking. In addition, we have shown that the use of the head pose estimation proposed by Fanelli et al. [4] is suitable for AR applications, as it runs in real-time.

Encouraged by the work of Meister et al. [29], for future work we plan to analyse the accuracy of the system to check if this method can be used for medical applications. Further improvements can be achieved by implementing a deformable registration algorithm to track the face, as proposed in [11], [30] and [31], or by implementing a better rendering algorithm for mixed reality, as proposed in [32].

Acknowledgments

We are grateful to Gabriele Fanelli and the PCL project for providing the open-source implementations. This research is financially supported by FAPESB and CAPES.

References

[1] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, “Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera,” in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, ser. UIST ’11, New York, NY, USA, 2011, pp. 559–568.

[2] T. Weise, H. Li, L. Van Gool, and M. Pauly, “Face/off: live facial puppetry,” in *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA ’09. New York, NY, USA: ACM, 2009, pp. 7–16.

[3] P. Viola and M. J. Jones, “Robust real-time face detection,” *Int. J. Comput. Vision*, vol. 57, no. 2, pp. 137–154, May 2004.

[4] G. Fanelli, T. Weise, J. Gall, and L. V. Gool, “Real time head pose estimation from consumer depth cameras,” in *33rd Annual Symposium of the German Association for Pattern Recognition (DAGM’11)*, September 2011.

[5] M. Leo and D. Manimegalai, “3d modeling of human faces- a survey,” in *Trendz in Information Sciences and Computing (TISC), 2011 3rd International Conference on*, Dec., pp. 40–45.

[6] B. Curless and M. Levoy, “A volumetric method for building complex models from range images,” in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH ’96. New York, NY, USA: ACM, 1996, pp. 303–312.

[7] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy, “Real-time 3d model acquisition,” in *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH ’02. New York, NY, USA: ACM, 2002, pp. 438–446.

[8] Y. Cui, S. Schuon, D. Chan, S. Thrun, and C. Theobalt, “3d shape scanning with a time-of-flight camera,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, june 2010, pp. 1173 –1180.

[9] H. Kato and M. Billinghurst, “Marker tracking and hmd calibration for a video-based augmented reality conferencing system,” in *Augmented Reality, 1999. (IWAR ’99) Proceedings. 2nd IEEE and ACM International Workshop on*, 1999, pp. 85 –94.

[10] G. Simon, A. Fitzgibbon, and A. Zisserman, “Markerless tracking using planar structures in the scene,” in *Augmented Reality, 2000. (ISAR 2000). Proceedings. IEEE and ACM International Symposium on*, 2000, pp. 120 –128.

[11] T. Weise, S. Bouaziz, H. Li, and M. Pauly, “Realtime performance-based facial animation,” in *ACM SIGGRAPH 2011 papers*, ser. SIGGRAPH ’11. New York, NY, USA: ACM, 2011, pp. 77:1–77:10.

[12] (2013, Jan.) Faceshift. [Online]. Available: http://www.faceshift.com/

[13] M. Hernandez, J. Choi, and G. Medioni, “Laser scan quality 3-d face modeling using a low-cost depth camera,” in *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, Aug., pp. 1995–1999.

[14] M. Breitenstein, D. Kuettel, T. Weise, L. Van Gool, and H. Pfister, “Real-time face pose estimation from single range images,” in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, june 2008, pp. 1 –8.

[15] C. Tomasi and R. Manduchi, “Bilateral filtering for gray and color images,” in *Computer Vision, 1998. Sixth International Conference on*, jan 1998, pp. 839 –846.

[16] P. Besl and H. McKay, “A method for registration of 3-d shapes,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 14, no. 2, pp. 239 –256, feb 1992.

[17] S. Rusinkiewicz and M. Levoy, “Efficient variants of the icp algorithm,” in *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, 2001, pp. 145 –152.

[18] Y. Chen and G. Medioni, “Object modelling by registration of multiple range images,” *Image Vision Comput.*, vol. 10, no. 3, pp. 145–155, Apr. 1992.

[19] S. J. Osher and R. P. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, 2003rd ed. Springer, Oct. 2002.

[20] L. Breiman, “Random forests,” in *Machine Learning*, 2001, pp. 5–32.

[21] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” in *Proceedings of the Second European Conference on Computational Learning Theory*, ser. EuroCOLT ’95. London, UK, UK: Springer-Verlag, 1995, pp. 23–37.

[22] B. T. Phong, “Illumination for computer generated pictures,” *Commun. ACM*, vol. 18, no. 6, pp. 311–317, Jun. 1975.

[23] R. J. Rost, *OpenGL(R) Shading Language (2nd Edition)*. Addison-Wesley Professional, 2005.

[24] (2013, Mar.) Kinfu. [Online]. Available: http://svn.pointclouds.org/pcl/trunk/gpu/kinfu/

[25] R. Rusu and S. Cousins, “3d is here: Point cloud library (pcl),” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, may 2011, pp. 1 –4.

[26] (2013, Mar.) Fanelli’s home page. [Online]. Available: http://www.vision.ee.ethz.ch/∼gfanelli/

[27] S. Bouaziz, A. Tagliasacchi, and M. Pauly, “Sparse iterative closest point,” *Computer Graphics Forum (Symposium on Geometry Processing)*, vol. 32, no. 5, pp. 1–11, 2013.

[28] M. Macedo, A. A. Jr., and A. C. Souza, “A markerless augmented reality approach based on real-time 3d reconstruction using kinect,” in *Workshop of Works in Progress (WIP) in SIBGRAPI 2013 (XXVI Conference on Graphics, Patterns and Images)*, S. M. Alejandro C. Frery, Ed., Arequipa, Peru, august 2013.

[29] S. Meister, S. Izadi, P. Kohli, M. Hämmerle, C. Rother, and D. Kondermann, “When can we use kinectfusion for ground truth acquisition?” in *Workshop on color-depth fusion in robotics, IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[30] H. Li, J. Yu, Y. Ye, and C. Bregler, “Realtime facial animation with on-the-fly correctives,” *ACM Transactions on Graphics*, vol. 32, no. 4, July 2013.

[31] S. Bouaziz, Y. Wang, and M. Pauly, “Online modeling for realtime facial animation,” *ACM Transactions on Graphics*, vol. 32, no. 4, July 2013.

[32] M. Knecht, C. Traxler, O. Mattausch, and M. Wimmer, “Reciprocal shading for mixed reality,” *Computers and Graphics*, vol. 36, no. 7, pp. 846 – 856, 2012.