REAL-TIME 3D FACE MODELING WITH A COMMODITY DEPTH CAMERA

Gregory P. Meyer and Minh N. Do

University of Illinois at Urbana-Champaign
{gmeyer3, minhdo}@illinois.edu

ABSTRACT

We propose a system that enables a person to quickly and easily generate a high-quality 3D model of their face using a low-cost depth camera such as the Kinect. The Kinect sensor provides depth images at video-rate, however, the depth images are often noisy and contain holes. Our system builds high-quality 3D face models by registering and integrating multiple depth images of a user's head. This method allows a user to generate a model of their face by simply moving their head in front of a fixed depth camera. Our method is built upon the KinectFusion system with modifications to enable segmented depth images to be robustly registered and integrated. Furthermore, we present a fast technique to segment a user's head from a depth image. Our system is able to run at a real-time rate on commodity hardware.

Index Terms— 3D Face Modeling, Kinect, Real-time

1. INTRODUCTION

The ability to generate a 3D model of a person's face has several uses in computer graphics and computer vision. For computer graphics, 3D face models can be used to construct a personalized avatar for multimedia applications. For computer vision, 3D face models can be used for facial recognition and face analysis.

There are two primary approaches to construct 3D models of human faces. One approach uses active sensors to capture the 3D geometry of a face, and the other uses passive sensors to reconstruct the 3D face model from one or more 2D images [4]. In the past, using an active sensor, such as a range scanner or depth camera, was restrictive due to their cost. With the success of Microsoft's Kinect, commodity depth cameras are now widely available.

The Kinect sensor provides depth images at real-time rates, however, the data is often noisy and contains holes. To improve quality, a sequence of depth images can be registered and integrated into a 3D model. The KinectFusion system [5] demonstrated that a high-quality 3D model of a room-size environment could be generated in real-time using a Kinect camera.

This work was supported in part by the Illinois-Intel Parallelism Center and by the National Science Foundation under Grant CCF-1218682.

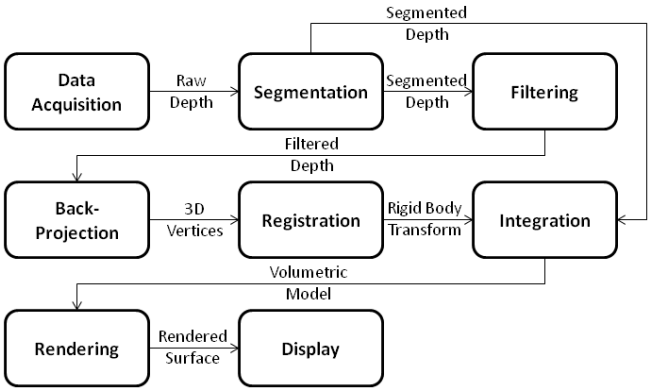


Figure 1. An overview of our system.

In this paper, we present an extension of the KinectFusion system that enables anyone to quickly and easily generate a high-quality 3D model of their face by simply moving their head in front of a fixed depth camera. Our system makes several technical improvements to the KinectFusion system to facilitate the construction of 3D face models. An overview of our system is shown in Figure 1.

In the following sections, we review work related to our proposed method, describe our system in detail, and present experimental results.

2. RELATED WORK

3D face modeling has been an active research topic for the past three decades. A large variety of methods utilizing both active and passive sensors have been developed. For an extensive survey of these systems refer to [4]. Recently, there has been a few proposed methods that use a Kinect camera to generate 3D face models [3, 9].

[9] uses the Kinect camera's color sensor to detect facial features such as the eyes and nose. They use these features to align a generic face model to a depth image. Afterwards, they deform the generic model to fit the depth image.

M. Hernandez *et al.* [3] proposed a system that generates a 3D face model from a sequence of depth images. They capture depth images from the Kinect camera, and they use a face detection algorithm to segment the user's head from the depth image. The segmented depth images are registered and integrated into a 3D model. Their method accumulates the registered depth images in an unwrapped

cylindrical 2D image, which allows them to use 2D spatial filters to remove noise.

Our method does not use a generic model to generate a 3D face model. Also, our method does not perform spatial filtering on the model to reduce noise.

Our system is based on the KinectFusion system [5]. The KinectFusion system was designed to generate a 3D model of room-size environment. To construct a model, a person moves a Kinect camera around a room, and the system registers and integrates the captured depth images into a 3D model. The purpose of our system is to enable a person to generate a 3D model of their face by fixing a depth camera and moving their head. For this approach to work, significant improvements to the KinectFusion system are necessary. We developed a fast technique to segment a person's head from a depth image. Unfortunately, the reduced set of depth measurements in the segmented depth images can affect our ability to accurately construct a 3D model. To allow robust registration of segmented depth images, we perform additional filtering to eliminate unreliable depth measurements, and pre-align the images based on their 3D center of mass before registration. In addition, our system is tuned to model faces instead of room-size environments.

3. METHOD

3.1. Data acquisition

The Kinect sensor is capable of capturing depth information of a scene at video-rate. At time τ our system acquires a raw depth image D_τ from the Kinect. D_τ consists of a set of pixels (u, v) with each pixel containing a depth measurement $D_\tau(u, v)$ in millimeters.

3.2. Segmentation

The depth image D_τ contains depth measurements for the entire scene within view of the sensor. For our system, we only want to consider depth pixels on the user's face. We developed a method to quickly segment the user's head from the depth image. To simplify the task, we make assumptions on how the user is positioned in front of a fixed camera. We assume the user is sitting upright near the camera with their head and shoulders clearly visible.

Our method segments the depth image into foreground and background regions, where the foreground includes the entirety of the user, and the background contains the rest of the environment. Connected component analysis is used to determine the foreground region. Two neighboring depth pixels are considered connected when their difference is below a threshold. We assume the foreground region is the largest component within a short distance from the camera.

To determine the location of the user's head within the foreground region, our method looks for a horizontal scan line that separates the foreground into head and torso regions. In order to accomplish this task, we generate a row histogram h,

$$h(v) = \sum_u M_\tau(u, v) \tag{1}$$

where M_τ is the foreground mask, and each bin $h(v)$ contains the width of the foreground region in row v. The bins in the row histogram can be split into sets $\mathcal{H}$ and $\mathcal{T}$,

$$\mathcal{H} = \{v \mid v \le s \text{ and } h(v) \neq 0\} \tag{2}$$

$$\mathcal{T} = \{v \mid v > s \text{ and } h(v) \neq 0\} \tag{3}$$

where s is the horizontal scan line that separates the bins. We want to determine s, such that, all rows that contain the head region are in set $\mathcal{H}$, and all rows that contain the torso region are in set $\mathcal{T}$. Typically, the width of a user's head is significantly different than the width of their torso, so we can find s by maximizing the variance between the histogram bins in $\mathcal{H}$ and $\mathcal{T}$,

$$s = \underset{s}{\operatorname{argmax}}\, W_\mathcal{H} W_\mathcal{T} (\mu_\mathcal{H} - \mu_\mathcal{T})^2 \tag{4}$$

where,

$$W_\mathcal{H} = \frac{|\mathcal{H}|}{|\mathcal{H} \cup \mathcal{T}|}, W_\mathcal{T} = \frac{|\mathcal{T}|}{|\mathcal{H} \cup \mathcal{T}|} \tag{5}$$

$$\mu_\mathcal{H} = \frac{\sum_\mathcal{H} h(v)}{|\mathcal{H}|}, \mu_\mathcal{T} = \frac{\sum_\mathcal{T} h(v)}{|\mathcal{T}|} \tag{6}$$

and $|\cdot|$ is the size of the set. A segmented depth image D'_τ,

$$D'_\tau(u, v) = \begin{cases} D_\tau(u, v) & \text{iff } M_\tau(u, v) = 1 \text{ and } v \le s \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

is generated by removing all depth measurements not in the head region.

3.3. Filtering

The depth measurements captured by the Kinect sensor are often noisy. This is especially true in regions that contain hair. The noise can affect our ability to correctly register depth images. To improve registration, we eliminate depth measurements with a high local variance $\sigma(u, v)^2$,

$$D''_\tau(u, v) = \begin{cases} D'_\tau(u, v) & \text{iff } \sigma(u, v)^2 < \sigma^2_{max} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where σ^2_{max} is the threshold on the local variance of a pixel. In addition, we use a bilateral filter to reduce noise while preserving depth discontinuity [8].

3.4. Back-projection

The Kinect sensor provides depth measurements in the form of a depth image. To register depth images together, we first need to create a point cloud by back-projecting the depth pixels to 3D vertices and estimate each vertices' surface normal. We can generate a vertex map V_τ by back-projecting each depth pixel into the camera's reference frame [5],

$$V_\tau(u,v) = D''_\tau(u,v)K^{-1}\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{9}$$

where K is the Kinect sensor's intrinsic calibration matrix.

Each vertices' surface normal can be estimated using the cross product of neighboring vertices [5],

$$\begin{aligned} N_\tau(u,v) = &\big(V_\tau(u+1,v) - V_\tau(u,v)\big) \\ &\times \big(V_\tau(u,v+1) - V_\tau(u,v)\big) \end{aligned} \tag{10}$$

3.5. Registration

To create a 3D model we need to align several vertex maps. For each vertex map V_τ we must determine the rigid body transformation $T_{g,\tau}$,

$$T_{g,\tau} = \begin{bmatrix} R_{g,\tau} & t_{g,\tau} \\ 0 & 1 \end{bmatrix} \tag{11}$$

that transforms the vertices and surface normals into a global coordinate space.

Iterative closest point (ICP) is a common method of computing a rigid body transformation that aligns 3D point clouds. [7] suggests a high speed variant of ICP that uses projective data association and the point-to-place error metric. Projective data association allows point correspondences to be computed directly by finding corresponding points along camera rays [1]. Also, the point-to-plane error metric has been found to improve convergence rates over other error metrics [7].

To implement ICP, we require an initial guess for the global transformation $T^0_{g,\tau}$. Since each vertex map V_τ only has points on the user's head, we can estimate the between frame transformation $T^0_{\tau-1,\tau}$ as the translation that aligns the 3D centers of V_τ and $V_{\tau-1}$,

$$T^0_{\tau-1,\tau} = \begin{bmatrix} I & (\mu_{\tau-1} - \mu_\tau) \\ 0 & 1 \end{bmatrix} \tag{12}$$

where μ_τ and $\mu_{\tau-1}$ are the centroids of V_τ and $V_{\tau-1}$, respectively. Using the between frame transformation $T^0_{\tau-1,\tau}$ and the previous frame's global transformation, we can approximate the current frame's global transformation $T^0_{g,\tau}$,

$$T^0_{g,\tau} = T_{g,\tau-1}T^0_{\tau-1,\tau}. \tag{13}$$

We iteratively refine our guess for the global transformation $T^k_{g,\tau}$, where each iteration k we find the set of corresponding points using projective data association, and solve for an incremental transformation by minimizing the point-to-plane error metric. This process is continued for a set number of iterations or until the error stops rapidly decreasing. After the last iteration $T_{g,\tau}$ is set to $T^k_{g,\tau}$.

3.6. Integration

Once the rigid body transformation $T_{g,\tau}$ is known, the depth image can be integrated into the 3D model. A volumetric representation is used to model the user's face. For our system the volume is roughly the size of a person's head. The volume contains 512^3 voxels, where each voxel p contains a weight $w(p)$ and a signed distance $f(p)$ to the nearest surface along camera rays [2]. The signed distances are only accurate near a surface, so they are truncated to prevent surfaces from interfering with each other [2]. We assume the actual surface is within $\pm\mu$ of a surface measurement, therefore, non-truncated signed distances only need to exist in this region of uncertainty near a surface [5]. Each segmented depth image D''_τ is incrementally fused into the volumetric model,

$$f(p) \leftarrow \frac{w(p)f(p) + w_\tau(p)f_\tau(p)}{w(p) + w_\tau(p)} \tag{14}$$

$$w(p) \leftarrow \min(w(p) + w_\tau(p), w_{max}) \tag{15}$$

where $w_\tau(p)$ and $f_\tau(p)$ are the incremental weight and signed distance, respectively. To determine $w_\tau(p)$ and $f_\tau(p)$, voxel p is projected into the depth image, and the signed distance is calculated by subtracting the depth of the voxel from the depth measurement,

$$S_p = D''_\tau(u,v) - \frac{\left\|T^{-1}_{g,\tau}p\right\|_2}{\|K^{-1}[u,v,1]^T\|_2} \tag{16}$$

and truncating the resulting value,

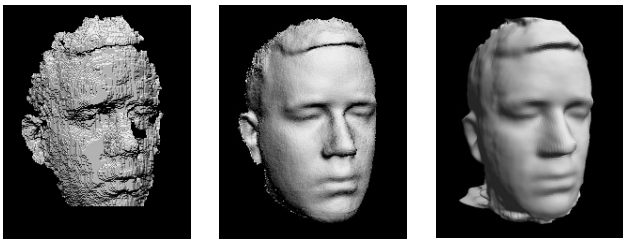
$$f_\tau(p) = \text{sign}(S_p)\min\left(1, \frac{|S_p|}{\mu}\right) \tag{17}$$

[5]. The weight is selected so that only voxels that lie in front of valid surface measurements are updated,

$$w_\tau(p) = \begin{cases} 1 & S_p \geq -\mu \text{ and } D''_\tau(u,v) \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

3.7. Rendering

Rendering the volume displays the current state of the model to the user, so they can reposition their head to



(a) Raw Image (b) Our Method (c) Hernandez

Figure 2. Comparison between a raw depth image, a face model generated with our proposed method, and a face model constructed with [3].

improve different parts of the model. Also, we can use the current state of the volume to refine the vertex map V_τ and normal map N_τ. Updating the vertices and surface normals based on the model improves the robustness of the registration step [5].

Ray casting is used to render the model. For each pixel, a corresponding ray is stepped through the volume. At each step s, a ray determines its current position in the volume,

$$p_s = t_{g,\tau} + d_s \left(R_{g,\tau} K^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \right) \qquad (19)$$

where d_s is the total distance traveled by the ray during steps $1, \ldots, s$, and (u, v) is the pixel location from which the ray was casted. The signed distance $f(p_s)$ is determined by trilinear interpolation of the signed distances stored in neighboring voxels [6]. A surface is found when the ray steps from a positive signed distance f_s^+ to a negative signed distance f_{s+1}^-. The location of the surface can be determined by interpolating d_s and d_{s+1} [5].

4. RESULTS

We compared a 3D face model generated by our proposed system with a model generated by [3]. Figure 2 shows the models created by the two systems without any post-processing applied. Our method is able to capture more facial details than [3].

Our system runs in real-time on commodity hardware. Using an Intel Core i7 CPU and a NVIDIA Geforce GTX 570 GPU, our system is capable of processing more than 25 frames per second. The execution time of our method is listed in Table 1. Our method was implemented using C++ and CUDA, and the source code is available on our website.

Table 1. Runtime of our system's components.

Component	Runtime (ms)
Data acquisition	0.7
Segmentation	4.2
Filtering	1.8
Back-projection	0.8
Registration	13.1
Integration	11.9
Rendering	3.7
Total	36.2

5. CONCLUSION

We presented a system that allows a user to effortlessly generate a high-quality 3D face model using a low-cost depth camera. The user simply moves their head in front of a commodity depth camera placed on their desk, and a 3D model of their face is constructed in real-time.

We developed a simple technique to extract a person's face from a depth image. In addition, we made several improvements to the KinectFusion system to enable segmented depth images to be robustly registered and integrated.

We were able to demonstrate a visual improvement over the state-of-the-art [3]. However, further comparisons are necessary to prove the strength of our method.

6. REFERENCES

[1] G. Blais and M. D. Levine, "Registering multiview range data to create 3D computer objects," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 8, pp. 820-824, Aug. 1995.

[2] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the Conference on Computer Graphics and Interactive Techniques*, 1996, pp. 303-312.

[3] M. Hernandez *et al.*, "Laser scan quality 3-D face modeling using a low-cost depth camera," in *European Signal Processing Conference*, 2012, pp. 1995-1999.

[4] M. J. Leo and D. Manimegalai, "3D modeling of human faces - A survey," in *Proceedings of the IEEE International Conference on Trends in Information Sciences and Computing*, 2011, pp. 40-45.

[5] R. Newcombe *et al.*, "KinectFusion: Real-time dense surface mapping and tracking," in *IEEE International Symposium on Mixed and Augmented Reality*, 2011, pp. 127-136.

[6] S. Parker *et al.*, "Interactive ray tracing for isosurface rendering," in *Proceedings of Visualization*, 1998, pp. 233-238.

[7] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proceedings of the IEEE International Conference on 3-D Digital Imaging and Modeling*, 2001, pp. 145-152.

[8] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proceedings of the IEEE International Conference on Computer Vision*, 1998, pp. 839-846.

[9] M. Zollhöfer *et al.*, "Automatic reconstruction of personalized avatars from 3D face scans," *Computer Animation and Virtual Worlds*, vol. 22, no. 2, pp. 195-202, Apr. 2011.