Compact and Accurate 3-D Face Modeling Using an RGB-D Camera: Let's Open the Door to 3-D Video Conference

Pavan Kumar Anasosalu [†,‡]
[†]University of Southern California
California, USA
anasosal@usc.edu

Diego Thomas[‡] and Akihiro Sugimoto[‡]
[‡]National Institute of Informatics
Chiyoda, Tokyo, Japan
{diego_thomas, sugimoto}@nii.ac.jp

Abstract

We present a method for producing an accurate and compact 3-D face model in real time using a low cost RGB-D sensor like the Kinect camera. We extend and use Bump Images for highly accurate and low memory consumption 3-D reconstruction of the human face. Bump Images are generated by representing the Cartesian coordinates of points on the face in the spherical coordinate system whose origin is the center of the head. After initialization, the Bump Images are updated in real time with every RGB-D frame with respect to the current viewing direction and head pose that are estimated using the frame-to-global-model registration strategy. While high accuracy of the representation allows to recover fine details, low memory use opens new possible applications of consumer depth cameras such as 3-D video conferencing. We validate our approach by quantitatively comparing our result with the result obtained by a commercial high resolution laser scanner. We also discuss the potential of our proposed method for a 3-D video conferencing application with existing internet speeds.

1. Introduction

Automatic reconstruction of 3-D models of the human face has a long history ([7], [9], [16]) yet is still a challenging topic in computer vision. In particular, most of the research dwells in generating precise models but few address the need for compactness of the generated 3-D models. Recently, a significant effort has been made to develop inexpensive consumer depth cameras that allow to acquire depth images at a video rate, *e.g.* Microsoft Kinect camera or Asus Xtion Pro camera. The video-rate depth cameras are now becoming a commodity tool for depth measurements with reasonable accuracy. Such sensors allow live 3-D face modeling, which opens new possibilities for interactive applications such as 3-D video conference. To achieve real-time remote interaction between users, a bottleneck is how to generate a compact live 3-D model with maintaining as much quality as possible. In comparison to some of the state-of-the-art algorithms that employ point clouds [4], Surfels [12] or volumes [6], the Bump Image [5, 15] representation consumes the least amount of memory and at the same time maintains accuracy and color information. Thanks to this compactness of Bump Images and also the availability of color information by RGB-D cameras, applications like 3-D video conferencing become tangible, where the local geometry of the head and the color information can be streamed live.

For automatic 3-D face modeling, some methods (*e.g.* [16]) use a predefined template mesh and perform a non-rigid alignment of the template mesh to the depth image obtained by a depth sensor. Usually such systems are time consuming and cannot be used for real-time applications. Moreover, using a predefined template brings some bias towards the template. Other popular methods like Kinect Fusion [6] require a large amount of memory and the output often needs to be post-processed before being used by other applications.

Our goal is to develop a method that enables live high-quality 3-D face model reconstruction from a consumer RGB-D camera while maintaining low memory use. Our proposed method opens new possibilities such as live streaming of the 3-D model during the reconstruction process, which cannot be avoid in 3-D video conferencing for example.

In this paper, we use spherical maps to represent the 3-D face model; they can also be termed as Bump Images. As shown in [15], some advantages of this representation are compactness and easiness of meshing and texturing. Since the head is roughly spherical in shape, we can represent points of the face more efficiently in spherical coordinates rather than Cartesian coordinates. Thus the whole head can be represented in a 6-channel 360x180 map where the first 3-channels represent the spherical coordinates and the last 3-channels represent the color information. This also simplifies the task of searching for adjacent points, thus mesh-

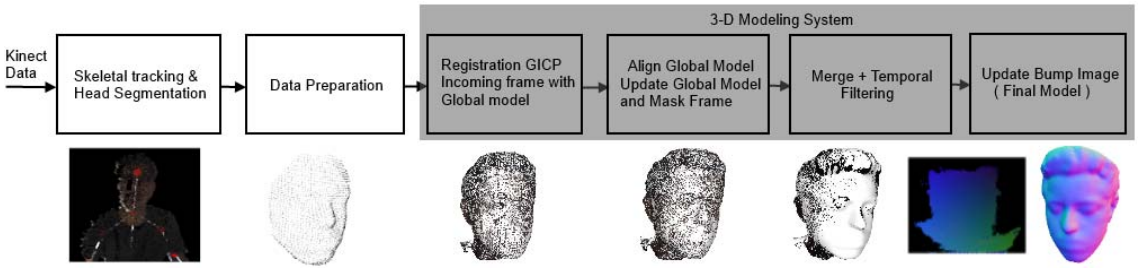

Figure 1: Overview of processing pipeline

ing the point cloud becomes easy when the model is represented by Bump Images. Since meshing is an integral part of our algorithm, this representation boosts the performance of our system. The formation of Bump Images is explained in Section 3.1.

The depth measurements corresponding to the head (which is segmented out from the rest of the depth image in a pre-processing step) are integrated to the global model in real time using the frame-to-global-model fusion framework. Live measurements are first aligned to the global model using the GPU implementation of the Generalized Iterative Closest Point Algorithm (GICP) [14]. Then the global model is updated using a weighted average filter and with respect to the current viewing direction. By doing so we achieve high quality 3-D face reconstruction. Our global model is kept in memory using Bump Images, which is compact and thus enables low memory consumption.

Though recent consumer depth sensors can deliver direct depth sensing and standard RGB video stream of VGA format at 30 frames per second, the depth data is noisy and the resolution is very low. Hence we use a view-centric approach based on OpenGL rendering pipeline [15] to align the global model with incoming point cloud data. The pipeline of our proposed method is illustrated in Figure 1.

2. Related Work

3-D face reconstruction methods can be classified into two categories: data-driven methods and template fitting methods. While methods that fall into the first category are well adapted to real-time applications, methods falling into the second one produce nicer results but are more time consuming and biased towards the input template.

Data-driven methods. Data-driven 3-D face modeling methods (*e.g.* [5], [9]) generally build the 3-D face model by integrating tracked live depth images into a common final 3-D model. Hernandez *et al*. [5] proposed to use the frame-to-reference-frame framework for tracking the live depth images. They also introduce the Bump Image framework, where the 3-D facial surface is parameterized by cylindrical coordinates (similar approach is used in [1]). Though high quality face models could be obtained, the camera tracking system is significantly affected by inaccurate pose estimates when the current frame has a large pose variation against the reference frame. Moreover, the proposed data integration method has its limitation as it is not view-centric and thus do not account for the directional nature of the noise in depth images. Also the cylindrical Bump Images record only the geometry; there is no provision for recording texture information. Recently, Thomas *et al*. [15] proposed a frame-to-global-model framework using Bump Images for indoor scene reconstruction. Though the proposed method can handle large pose variation in the image sequences, the fixed 3-D scene is represented as a collection of planes, which is not well adapted to reconstruct a moving 3-D face.

In [9], Newcombe *et al*. proposed Kinect Fusion where the global model is a TSDF volumetric scene representation and the camera is tracked based on the frame-to-global-model framework which is more robust against drift. For reconstructing the scene, the method makes use of volumetric integration which is expensive in memory, unlike the lightweight Bump Images. This poses limitation on dealing with large amount of data, and this is critical in, for example, teleconferencing applications or streaming applications. Though, the method proposed in [9] was developed for a fixed scene, Izadi *et al.* [6] extended it by considering dynamic actions of the foreground. However we show quantitatively that our method outperforms Kinect Fusion in terms of both accuracy and compactness of the 3D models.

Template fitting methods. Methods for 3-D face modeling that fall into the template fitting category (*e.g.* [1], [2], [16]) start from a generic model which is then deformed to fit the input face. Most of shape fitting methods employ a pre-defined morphable template mesh obtained by statistical methods like the PCA, and they are usually parameterized by shape and texture parameters. In [1], Blantz *et al*. described an iterative algorithm for fitting a morphable template mesh to a textured depth scan. The method simulates the process of image formation in 3-D space and estimates 3-D shape and texture information from single im-

ages. The concept of cylindrical Bump Images is introduced for computing dense point-to-point correspondences for defining appropriate shape and texture parameters. The energy function is minimized by both color and depth information to estimate the best fitting morphable model. The method described in [16] employed a morphable model that is fit to the depth images obtained from an RGB-D camera. The template mesh and the incoming frame are aligned using features detected in the RGB image as a coarse alignment step. The template is then aligned non-rigidly to the incoming frame and the morphable model is fit to the template. Unfortunately, this approach produces results that are biased towards the template as seen in the results of [16].

Other methods like [3] make use of complex sensor arrangements to estimate the parameters used to reconstruct the shape. The binocular photometric stereo setup described in [3] makes use of the binocular stereo to estimate the depth by triangulation and the normals from shading variations by using controlled LED lights. The shape reconstruction is formulated as a Poisson equation constrained by the depth image and the normal map previously estimated. The algorithm completely relies on the data capture equipment and the method is not benchmarked for speed. By contrast, our proposed method aims to be independent of the depth sensor used for data acquisition as we assume a noisy and incomplete (filled with holes) depth data as the input. Also for better speed and real-time implementation we approximate the surface normal rather than estimate them using shape from shading cues by varying illumination as required by methods described in [3] or [8]. This results in high quality 3-D face models obtained in real time.

3. Proposed method

Our proposed method takes a live sequence of RGB-D images streamed from a fixed consumer RGB-D sensor with unknown head pose. We assume that the relative movement of the head between two successive frames is small and that the facial expression do not change during reconstruction. We start with an initial location and radius of the head obtained using the SDK tracking system of the sensor. With this setting, our method tracks the pose of the head in successive frames using the frame-to-global-model framework and fuses aligned data in a view-centric manner. We make use of the 2-D grid organization of data in Bump Images and capacities of OpenGL for fast and robust processing.

3.1. Bump Images for 3-D face modeling

Following [5], we employ canonical 2-D maps to represent the 3-D human face. The main advantage of using 2-D canonical maps (also called Bump Images) compared to other standard 3-D scene representations such as volumes, cloud of points or Surfels is that it requires the less amount of memory with guaranteeing similar accuracy. We employ an extension of the Bump Image representation to improve accuracy of the obtained 3-D models. Namely, (1) we propose to use spherical coordinates instead of cylindrical coordinates (this allows us to reconstruct the whole head but not just the face) and (2) we use two additional displacement values for the polar and azimuthal angles, as well as RGB channels, as in [15].

The Bump Image is a 2-D unwrapped spherical map of the head. To build this spherical map, we have to first obtain points in the local coordinate system of the head. Once the points are transformed to the local coordinate system (see Section 3.2) we change the coordinate representation from local Cartesian coordinates to local spherical coordinates. The map is formed such that the horizontal distance on the map corresponds to ϕ and the vertical distance corresponds to θ. The value in the first channel can be represented by $R(\theta, \phi)$, which denotes the radius value at a pixel corresponding to a specified θ and ϕ. The θ and ϕ are rounded off by discarding the decimal values, which gives us a resolution of 1.0 degree. By losing out on decimal values as done in [5] we will obtain a really coarse map, hence (as proposed in [15]) we record the lost precision in the second and third channels of the Bump Image. The second channel encodes the lost precision of θ while forming the map, similarly the third channel encodes the lost precision of ϕ while forming the map. The lost precision can be computed as $\theta_{precision} = \theta - \lfloor\theta\rfloor$, $\phi_{precision} = \phi - \lfloor\phi\rfloor$. The remaining three channels encode the color (RGB) information.

3.2. Initialization

In order to initialize our proposed 3-D face representation we use the Kinect SDK v1.7 [13], which is equipped with skeletal tracking. The SDK tracking system provides a rough estimate of the location of the head. We make use of the tracked skeletal nodes of the head and the shoulder center to identify the approximate center of the head in the first RGB-D frame. After estimating the position of the head center a new coordinate system is assigned to the head with the origin located at the estimated head center. The radius of the head (we use the sphere to identify points that belong to the head) is then estimated using the median distance of points nearby the center of the head to roughly segment out the head from the rest of the depth image.

Let $\vec{\mathbf{x}}_{ls}, \vec{\mathbf{y}}_{ls}, \vec{\mathbf{z}}_{ls}$ be the local coordinate system of the head. The $\vec{\mathbf{y}}_{ls}$ axis is the unit vector aligned along the line connecting the head node and the shoulder center node tracked by the SDK tracking system. The remaining two axes can be obtained by computing two orthogonal vectors to the $\vec{\mathbf{y}}_{ls}$ axis. From the estimated axes and the head center c we can define a transformation T_{ws2ls} that transforms points from the sensor coordinate system (can also be termed as world coordinate system since we assume that the sensor is stationary) to the newly defined local coordi-

nate system of the head.

Suppose that $\mathbf{p} = (P_x, P_y, P_z, 1)^\top$ is a point in the world coordinate system, then the corresponding point $\mathbf{p}' = (P'_x, P'_y, P'_z, 1)^\top$ in the local coordinate system of the head can be computed by $\mathbf{p}' = (T_{ws2ls} \cdot T_{pose}) \cdot \mathbf{p}$. Namely,

$$\begin{bmatrix} P'_x \\ P'_y \\ P'_z \\ 1 \end{bmatrix} = \begin{bmatrix} X_l s_x & Y_l s_x & Z_l s_x & C_x \\ X_l s_y & Y_l s_y & Z_l s_y & C_y \\ X_l s_z & Y_l s_z & Z_l s_z & C_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot T_{pose} \cdot \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} \quad (1)$$

where T_{pose} is the 4×4 pose matrix of the head in the current RGB-D frame, (C_x, C_y, C_z) is the head center in world coordinate system and the terms $(X_{ls_x}, X_{ls_y}, X_{ls_z})$, $(Y_{ls_x}, Y_{ls_y}, Y_{ls_z})$, $(Z_{ls_x}, Z_{ls_y}, Z_{ls_z})$ respectively correspond to $\vec{\mathbf{x}}_{ls}$ axis , $\vec{\mathbf{y}}_{ls}$ axis and $\vec{\mathbf{z}}_{ls}$ axis of the local coordinate system of the head expressed in the world space coordinates.

Note that, for every pixel in the depth image the SDK assigns a player index indicating that a particular point belongs to a foreground, which in our case is the target user. This indexing helps to segment the user from the background. To segment the head we consider only those points that are within a radius of 20.0 cm of the estimated head center. There are many factors to consider for approximating the circumference of the human head like region, gender, athletic, non-athletic, etc. We chose to approximate the radius of the head as 20.0 cm which is a large estimate, so as to take into consideration all categories of people. To further refine our estimate, we transform the point cloud to the local coordinate system and then transform it again to the spherical coordinates and estimate the median of the radii of the points converted to spherical coordinates. The median value is a rough estimate of the radius of the head. Using this estimated radius, the point cloud is segmented again, resulting in removing erroneous points.

3.3. Data Preparation

In the frame-to-global-model fusion framework, incoming RGB-D frames have first to be aligned to the current global model. To do so we use the GPU version of GICP, as proposed in [6], which proved to be fast with sufficient accuracy. In this approach, a predicted RGB-D frame to which the incoming RGB-D frame is aligned has to be generated from the global model. By doing so, projected point association can be used for fast GPU point matching. To generate the predicted RGB-D image with respect to the lastly computed pose of the head we use OpenGL Frame Buffer Object (FBO) capacities together with the 2-D grid organization of data given by the Bump Image. More precisely, the 2-D grid provides us with a natural 3-D quadrangulation of the global model that can be rendered in the camera image plane using OpenGL rendering capacities.

With the advent of CUDA and OpenGL interoperability ([10], [11]), we can generate RGB-D images efficiently as we can map the FBO buffer's memory and read back the refined data and reproject it to the world coordinate system from OpenGL's view context. Since the FBOs reside in a GPU's texture cache, the reading and writing process is executed with minimum overhead.

3.4. Head Modeling

The recent work on dense 3-D modeling using an RGB-D camera has demonstrated the superiority of the frame-to-global-model (FGM) framework over the frame-to-reference-frame (FRF) framework. This is why we strive to employ the FGM framework for reconstructing 3-D models of the head. The FGM framework consists of dynamically updating the global model with live depth measurements while using it to register incoming depth images. To be more precise, the global model (Bump Image in our case) is initialized with the first depth image (see Section 3.2) and then augmented by the subsequent frames. For each subsequent frame, a predicted depth image is generated from the current global model, to which the incoming depth image is aligned.

To efficiently perform the FGM framework two tasks are of major importance: (1) integration part (*i.e.* how to update the global model with incoming frames) and (2) depth image prediction (*i.e.* how to quickly and accurately generate a depth image from the updated global model). For the task (1) we employ the view-centric integration strategy (as proposed in [15]) that takes into account the directional bias of the noise in the depth image, and we contribute in the task (2) by demonstrating the potential of interoperability between OpenGl and Cuda for fast depth image generation using a spherical Bump Image (see Section 3.3).

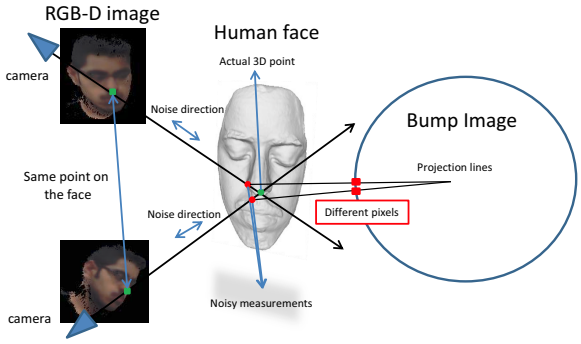

Figure 2: The same point projects onto different pixels.

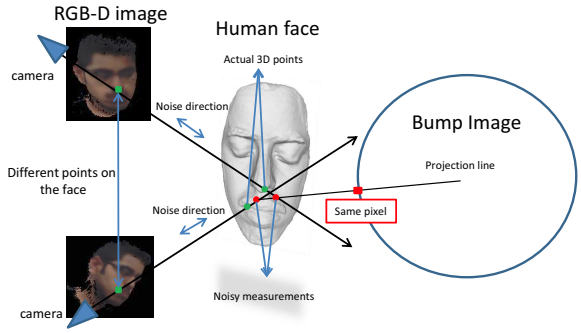

Figure 3: Two different points project onto the same pixel.

Depth measurements' integration. As proposed in [6] or in [5] we employ the running average to integrate new measurements in the global model while reducing input noise. In order to minimize the noise a temporal mean filter is employed and points lying within $1cm$ deviation to the global model are subjected to mean filtering. Note that a crucial, implicit assumption for this approach

Algorithm 1 Merge

Require: Two aligned depth images D_1 and D_2, their color images rbg_1 and rgb_2, and a projected confidence image $\pi Mask$ corresponding to D_2.

Ensure: A merged depth image D_{merge}, a color image rgb_{merge} and an updated projected confidence image $\pi Mask_{merge}$.

for ($i \in [1 : col]$, $j \in [1 : row]$) (col and row are the number of columns and rows respectively in the depth images) **do**

  if $\|D_1(i,j) - D_2(i,j)\| \leq 1.0$ cm **then**

    $D_{merge}(i,j) \leftarrow \frac{D_1(i,j)+D_2(i,j)\pi Mask(i,j)}{1+\pi Mask(i,j)}$

    $rgb_{merge}(i,j) \leftarrow \frac{rgb_1(i,j)+rgb_2(i,j)\pi Mask(i,j)}{1+\pi Mask(i,j)}$

    $\pi Mask_{merge}(i,j) \leftarrow \pi Mask(i,j) + 1$

  else if $\pi Mask(i,j) > 1.0$ **then**

    $D_{merge}(i,j) \leftarrow D_2(i,j)$

    $rgb_{merge}(i,j) \leftarrow rgb_2(i,j)$

    $\pi Mask_{merge}(i,j) \leftarrow \pi Mask(i,j)$

  else if $D_1(i,j) > 0.0$ **then**

    $D_{merge}(i,j) \leftarrow D_1(i,j)$

    $rgb_{merge}(i,j) \leftarrow rgb_1(i,j)$

    $\pi Mask_{merge}(i,j) \leftarrow 1.0$

  else

    $D_{merge}(i,j) \leftarrow 0$

    $rgb_{merge}(i,j) \leftarrow (0.0, 0.0, 0.0)$

    $\pi Mask_{merge}(i,j) \leftarrow 0$

  end if

end for

return D_{merge}, rgb_{merge} and $\pi Mask_{merge}$

to be efficient is that all measurements that are averaged together must come from the same point on the head. A glaring example is that averaging points belonging to the nose with those belonging to the ear does not work. This is why registering incoming frame has to be done before integration and is a crucial process.

However, even if the registration process is successful a problem arises due to noise when integrating new depth measurements into the Bump Image as seen in [5]. The fact is that (as shown in Figs. 2 and 3), due to noise the same point viewed in two different frames may be projected into different pixel coordinates in the Bump Image, and also two different points of the head may be projected into the same pixel of the Bump Image. This results into erroneous averaging computations. In order to avoid this problem, the integration process should be executed directly in the camera plane domain rather than in the Bump Image domain. This is because the noise in a depth image obtained with an RGB-D camera is mainly distributed along the viewing direction.

From the current predicted depth image we first align the incoming depth image to the predicted one using the GPU GICP [6]. We then generate a new depth image from the Bump Image with the lastly estimated head pose, as well as its corresponding projected confidence image. To generate the projected confidence image, we use a confidence mask (with the same dimension as the

Algorithm 2 UpdateBumpImage

Require: A depth image D, a color image rgb and its corresponding projected confidence image $\pi Mask$, a Bump Image $Bump$ and its confidence map $Mask$.

Ensure: The updated Bump Image $Bump$ and its confidence map $Mask$.

for ($i \in [1 : col]$, $j \in [1 : row]$) **do**

  $\mathbf{p}(i,j) \leftarrow$ vertex at pixel (i,j) in D

  $\mathbf{p}'(i,j) \leftarrow$ vertex in local coordinate system of the head (Equation (1)).

  $(k,l) \leftarrow$ the pixel coordinates in $Bump$ corresponding to the spherical coordinates of $\mathbf{p}'(i,j)$

  if $Mask(k,l) < \pi Mask(i,j)$ **then**

    $Bump(k,l) \leftarrow (R(\theta,\phi), \theta_{precision}, \phi_{precision}, rgb(i,j))$ (Section 3.1)

    $Mask(k,l) \leftarrow \pi Mask(i,j)$

  end if

end for

return $Bump$ and $Mask$

Bump Image) where each pixel of the confidence mask counts the number of times where the point in the Bump Image has been observed. We then perform the running average between pixels in the incoming depth image (with confidence of 1.0) and those of the newly generated depth image (with confidence of the value in the projected confidence image) to generate a merged depth image with updated confidence values. The merged depth image becomes the predicted depth image for the next incoming frame. All pixels of the new predicted depth image are projected onto the Bump Image and the values of a pixel (i,j) in the Bump Image are replaced if and only if a pixel in the predicted depth image project onto this pixel (i,j) and if its new confidence value is higher than the one at the pixel (i,j) in the Bump Image. This process is detailed in Algorithms 1, 2 and 3.

Algorithm 3 3-D head modeling

Require: A predicted depth image D_k, a Bump Image $Bump_k$, its corresponding confidence map $Mask_k$, an input depth image D_1 and the color image rgb_1.

Ensure: A new predicted depth image D_{k+1} and an updated Bump Image $Bump_{k+1}$ with its corresponding confidence map $Mask_{k+1}$.

$T_{curr} \leftarrow$ AlignGICP(D_1, D_k) ([6])

$T_{pose} \leftarrow T_{pose} * T_{curr}$

(D_2, rgb_2, $\pi Mask$)$\leftarrow$ RenderMesh($Bump_k$, $Mask_k$, T_{pose}^{-1}) (Section 3.3)

(D_{k+1}, rgb_{k+1}, $\pi Mask_{k+1}$)$\leftarrow$ Merge(D_1, rgb_1, D_2, rgb_2, $\pi Mask$)

($Bump_{k+1}$, $Mask_{k+1}$)$\leftarrow$ UpdateBumpImage(D_{k+1}, rgb_{k+1}, $\pi Mask_{k+1}$, $Bump_k$)

return D_{k+1}, $Bump_{k+1}$ and $Mask_{k+1}$

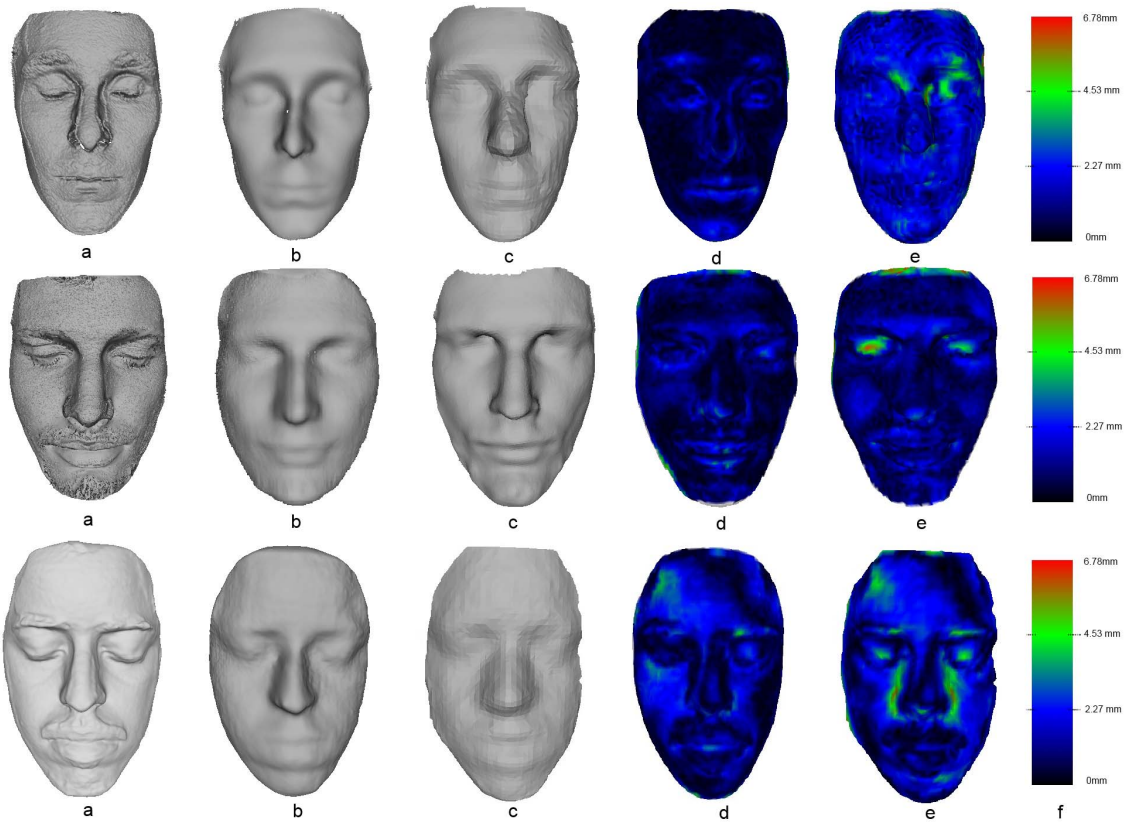

Figure 4: (a) Model obtained by laser scanner (b) Model obtained by our system (c) Model obtained by Kinect Fusion algorithm (d) Heatmap obtained for face reconstructed by our system (e) Heatmap obtained for face reconstructed by Kinect Fusion (f) Color LUT, indicates the color used to represent deviation

4. Experiments

To demonstrate the effectiveness of our proposed method, we evaluated our algorithm by comparing our results with the one obtained with a high accuracy laser scanner. We also compared our results with Kinect Fusion [6], a state-of-the-art dense 3-D modeling system using an RGB-D camera that also employs the frame-to-global-model strategy but is purely focusing on accuracy of the generated 3-D model.

We employed a Konica Minolta Vivid 910 range scanner to obtain the ground truth of the faces shown in Fig. 4 (a). Note that we are comparing only the facial geometry since the laser scanner fails in recognizing hair and the data is very noisy in hairy regions of the head. We used the software "Kinect Fusion Explorer-D2D" provided with the Kinect for Windows Developer Toolkit v.1.7.0 [13] to generate the 3-D meshes shown in Fig. 4 (c). Note that when using the laser scanner or Kinect Fusion, the user should remain still during the scanning process, while with our method the user can be moving in front of the camera. Our method was run on an Intel Xeon processor clocked at 3.47 GHz. The GPU in use is a NVIDIA GeForce GTX 580. Our method runs at 30 fps when the Bump Image in use is of resolution (360×180), at 22 fps for an increased resolution of $2 * (360 \times 180)$ and at 20 fps for a resolution of $3 * (360 \times 180)$. With a resolution of $3 * (360 \times 180)$ the Bump Image consumed 13.35MB of memory (6 floats for each pixel). In comparison, Kinect fusion consumed 1000MB of memory for a volume of $(1.0 \times 1.0 \times 1.0)m$ with 640 subdivisions in each dimension (1 float for each voxel).

Figure 4 shows examples of quantitative results on the accuracy of the generated 3-D models of the face obtained with our proposed method and with Kinect Fusion. To quantify the accuracy of the obtained results we compared them with the 3-D models obtained with the laser scanner as shown in Fig. 4 (a) (they are considered as the ground truth). For each example, we build heat maps that indicate the deviation of the 3-D models obtained with our proposed method and with Kinect Fusion from the ground truth. Namely, we first aligned the 3-D models to the ground truth models using the GICP algorithm (the pose of the 3-D models were initialized manually), and then generated depth images for all 3-D models with respect to the pose of the head obtained with the laser scanner. The heat maps shown in Figure 4 (d) and (e) were obtained by computing the absolute residual error in the depth value for each pixel[1]. From Fig. 4 we can see that our model is very close to the laser scan, in fact the mean deviations from the ground truth are 0.78 mm for the first face, 0.92 mm for the second face and 1.3 mm for the third face, while those from Kinect fusion are 1.6

[1]Note in Fig. 4 points that do not belong to the face were removed manually using MeshLab only for the experiment of generating heat maps.

mm, 1.7 mm and 1.7 mm respectively. Fig. 5 shows the distribution of errors in the heat maps generated in Fig. 4. From the error distribution we can see that most of the points of the 3-D models obtained using our proposed method lie close to the ground truth. We stress that our proposed method performs at least as well as Kinect Fusion in terms of accuracy while requiring drastically less amount of memory.

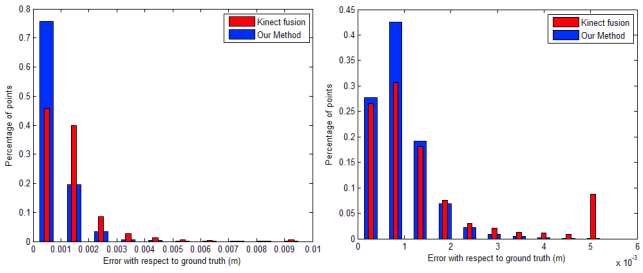

Figure 5: (left) Error Distribution in Face-1 (Fig. 4), (right) Error Distribution in Face-2 (Fig. 4)

Figures 6 and 7 demonstrate the advantage of using the FGM framework over the FRF framework [5]. We can see that our proposed method was able to reconstruct the 3-D models of the head even though the pose of the head became very different from the first reference frame. In the extreme case shown in Fig. 6 at some point there is no more overlapping regions between the current frame and the reference frame. In this situation the FRF framework does not work. In contrast, our method also performed wells with respect to zooming-in or zooming-out effects (Fig. 7), which are challenging for the FRF framework since it implies registering cloud of points in different resolutions.

One of the advantages of using the Bump Image for representing the final model is the ease of varying resolution. Because of the compactness of the representation even with a system of base configuration our proposed method can produce a dense model. Figure 8 shows results obtained with three different resolutions for the Bump Image. As we can see the higher the resolution becomes, the finer the details become. Note that even with the lowest resolution, the obtained 3-D models are of descent quality. Note that, however, increasing the resolution may not always be judicious. Indeed, for a pixel in the Bump Image even though the resolution increases less points are accumulated for the running average, which results in a more spiky model. Then it requires more time to obtain a smooth model and the user must move slower to avoid misalignment that may come from noisy normal estimates.

5. Conclusion And Discussion

We have presented a novel method for the automatic creation of accurate and compact 3-D face models. By using the FGM framework and the view-centric integration method, which takes into account the directional nature of the noise of the RGB-D camera, we obtained highly accurate 3-D face models. Experimental results compared against results obtained with a laser scanner confirm the accuracy of the 3-D face models obtained with our proposed method. The use of 2-D Bump Images to record texture and local geometry realized the low memory use.

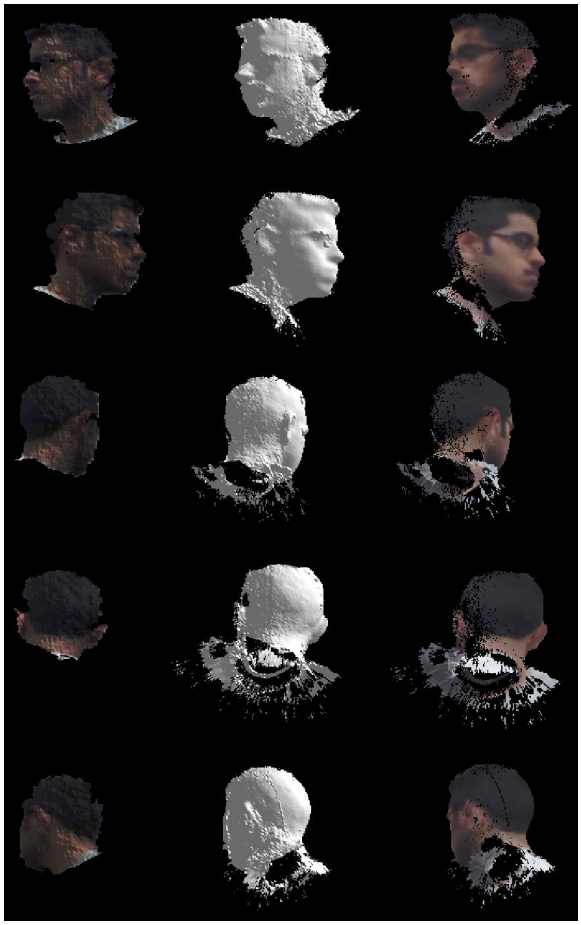

Figure 6: 360 degree tracking sequence (left) Incoming frame (middle) Global model (right) Re-projected Bump Image (final model)

Due to the compactness of the representation, accuracy and speed of our proposed method, 3-D telepresence applications like Skype or Face Time can be made viable with current internet speeds. The 6-D Bump Image represents both the local geometry of the head and the RGB information which is sufficient to re-project the head back in 3-D. For any peer-to-peer video conferencing application we can transmit live Bump Images and re-project it to 3-D at the receiving end. In addition to the compactness of the Bump Images, the images can be scaled based on the bandwidth available at run-time. During low bandwidth conditions the Bump Images can be scaled down to represent a coarse geometry of the face, similarly during high bandwidth conditions a higher resolution of Bump Image can be transmitted like $3 * (360 \times 180)$ which is still of a lower resolution than the $720p$ video that applications like Skype are capable of streaming. Note that our proposed method can be extended for reconstructing other parts of the body to produce a full body model. Note also that our proposed method can be extended to record facial expressions to provide a more realistic feel for video conferencing applications.

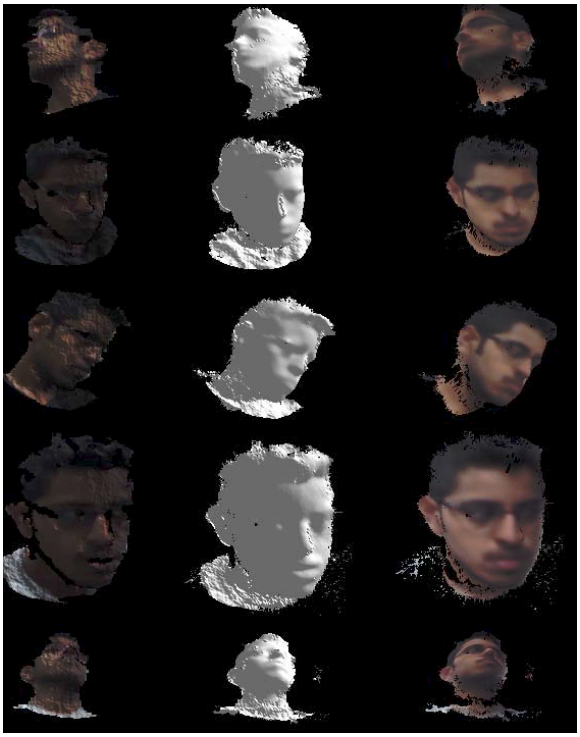

Figure 7: Various Poses tracked by the system (left) Incoming frame (middle) Global model (right) Re-projected Bump Image (final model)

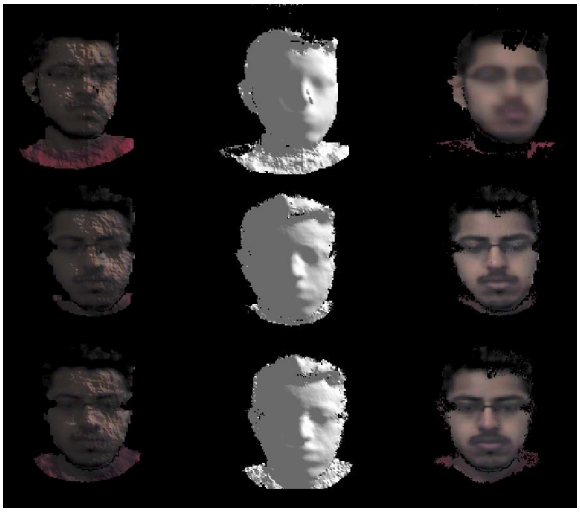

Figure 8: (top) Bump Image of (360×180) resolution (middle) Bump Image of $2 * (360 \times 180)$ resolution (bottom) Bump Image of $3 * (360 \times 180)$ resolution

References

[1] V. Blanz and T. Vetter. Face recognition based on fitting a 3d morphable model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1063–1074, 2003. 2

[2] P. Breuer, K.-I. Kim, W. Kienzle, B. Scholkopf, and V. Blanz. Automatic 3d face reconstruction from single images or video. In *Proc. of the 8th IEEE International Conference on Automatic Face Gesture Recognition (FG '08)*, pages 1–8, 2008. 2

[3] Chaoyang, L. Wang, Y. Wang, F. Matsushita, K., and Soong. Binocular photometric stereo acquisition and reconstruction for 3d talking head applications. In *Interspeech 2013 submission*, 2013. 3

[4] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. Rgb-d mapping: Using depth cameras for dense 2d modeling of indoor environments. *Proc. of International Symposium on Experimental Robotics*, 2010. 1

[5] M. Hernandez, J. Choi, and G. Medioni. Laser scan quality 3-d face modeling using a low-cost depth camera. *Proc. of the 20th European Signal Processing Conference (EUSIPCO)*, pages 1995–1999, 2012. 1, 2, 3, 4, 5, 7

[6] 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 *Proc. of UIST*, pages 559–568, 2011. 1, 2, 4, 5, 6

[7] B. Kainz, S. Hauswiesner, G. Reitmayr, M. Steinberger, R. Grasset, L. Gruber, E. Veas, D. Kalkofen, H. Seichter, and D. Schmalstieg. Omnikinect: real-time dense volumetric data acquisition and applications. *Proc. of the 18th ACM symposium on Virtual reality software and technology*, pages 25–32, 2012. 1

[8] I. Kemelmacher-Shlizerman and R. Basri. 3d face reconstruction from a single image using a single reference face shape. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(2):394–405, 2011. 3

[9] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. *Proc. of the 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 127–136, 2011. 1, 2

[10] H. Nguyen. *Gpu gems 3*. Addison-Wesley Professional, first edition, 2007. 4

[11] NVIDIA. *NVIDIA CUDA Programming Guide 2.0*. 2008. 4

[12] H. Pfister, M. Zwicker, J. Baar, and M. Gross. Surfels: Surface elements as rendering primitives. *Proc. of ACM Transactions on Graphics (SIGGRAPH'00)*, 2000. 1

[13] M. K. SDK. Version 1.7. 3, 6

[14] A. Segal, D. Hhnel, and S. Thrun. Generalized-icp. *Robotics: Science and Systems'09*, pages –1–1, 2009. 2

[15] D. Thomas and A. Sugimoto. A flexible scene representation for 3d reconstruction using an rgb-d camera. *Proc. of International Conference on Computer Vision (ICCV 13)*, 2013. 1, 2, 3, 4

[16] M. Zollhofer, M. Martinek, G. Greiner, M. Stamminger, and J. Submuth. Automatic reconstruction of personalized avatars from 3d face scans. *Comput. Animat. Virtual Worlds*, 2(2-3):195–202, 2011. 1, 2, 3