FaceCollage: A Rapidly Deployable System for Real-time Head Reconstruction for On-The-Go 3D Telepresence

Fuwen Tan*
University of Virginia
fuwen.tan@virginia.edu

Chi-Wing Fu
The Chinese University of Hong Kong
and Shenzhen Key Lab, VCHIT, SIAT
cwfu@cse.cuhk.edu.hk

Teng Deng
Nanyang Technological University
dengteng@ntu.edu.sg

Jianfei Cai
Nanyang Technological University
asjfcai@ntu.edu.sg

Tat-Jen Cham
Nanyang Technological University
astjcham@ntu.edu.sg

ABSTRACT

This paper presents *FaceCollage*, a robust and real-time system for head reconstruction that can be used to create easy-to-deploy telepresence systems, using a pair of consumer-grade RGBD cameras that provide a wide range of views of the reconstructed user. A key feature is that the system is very simple to rapidly deploy, with autonomous calibration and requiring minimal intervention from the user, other than casually placing the cameras. This system is realized through three technical contributions: (1) a fully automatic calibration method, which analyzes and correlates the left and right RGBD faces just by the face features; (2) an implementation that exploits the parallel computation capability of GPU throughout most of the system pipeline, in order to attain real-time performance; and (3) a complete integrated system on which we conducted various experiments to demonstrate its capability, robustness, and performance, including testing the system on twelve participants with visually-pleasing results.

CCS CONCEPTS

• **Computing methodologies → Image and video acquisition**; *Graphics systems and interfaces*; • **Applied computing →** *Telecommunications*;

KEYWORDS

3D telepresence, RGBD sensors, face capture

1 INTRODUCTION

The grand vision in telepresence has always been to provide users with an immersive 3D experience so real that it would circumvent the need for people to physically meet face-to-face; it would reduce the time and cost of travel, bringing people working and living apart closer together. Various pioneering systems [3, 14, 26] have been developed over the last two decades, and 3D telepresence continues to be a major focus of research, with the well-publicized Holoportation system [25] being a recent example.

Despite this level of interest, nearly all videoconferencing activities today are still carried out with 2D systems, often on mobile devices with embedded cameras and running free software such as Skype. There are at least two reasons for this. One is technological cost, as many existing telepresence systems depend on expensive equipment for real-time and high quality 3D capture; the second is an effort barrier as such systems typically require extensive calibration and pre-processing, which meant that either they are not portable and dedicated spaces have to be set aside for system setup, or the users have to be knowledgeable and prepared to undertake a lengthy, often tedious setup process. With such systems, it is not yet practical to have 3D telepresence on-the-go, say in a hotel room or cafe.

The research in this paper is motivated by our belief that there is an important demand gap to fill between 2D basic videoconferencing software, and complex 3D telepresence systems. In order to facilitate personal 3D telepresence, we want to create a system that is able to capture a user's head in 3D with decent visual quality and in real-time; but more importantly one that is easily and rapidly deployable anytime and anywhere by an average user.

More specifically, the requirements for our 3D head reconstruction system are: (1) The system needs to run in real-time and be able to render the reconstructed head with decent visual quality. Geometric accuracy is less important than visual quality, which impacts the user experience more significantly. (2) We want to avoid the need to rig high-end sensors such as multi-camera arrays. Instead, we plan to rely on a small number (in this paper, only two) of commodity depth cameras such as the Kinect [1], which facilitates real-time depth acquisition at low cost. There are already some existing works [12, 17, 19] using these for telepresence. However, a major challenge with these depth cameras is substantial noise and missing data. (3) We want to avoid methods that rely on the availability of extensive computational power, such as computing clusters. We instead aim to use only a single desktop or laptop, leveraging an onboard GPU to meet our computational needs. (4) Most importantly, we want to avoid the need for an elaborate and tedious setup procedure for configuring and calibrating various sensors [11], which is prevalent in existing real-time 3D acquisition systems. Such a requirement would be beyond the capability and

*The majority of the work was done while working at Nanyang Technological University.

MM '17, October 23–27, 2017, Mountain View, CA, USA
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-4906-2/17/10…$15.00
https://doi.org/10.1145/3123266.3123281

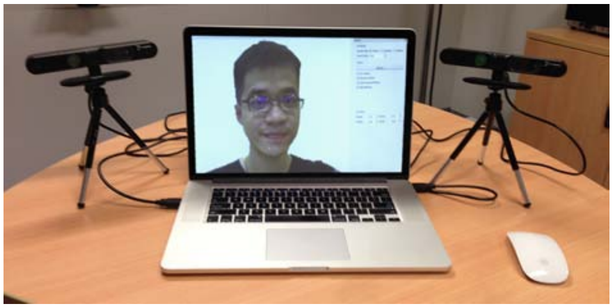

Figure 1: Our robust setup: with only two consumer grade RGBD cameras.

patience of normal users. Instead, we want a system that is potentially simple enough for a general user to set up without requiring one to tediously reference to a manual.

In this paper, we present a novel solution called *FaceCollage* to handle these issues, where a robust GPU-based method is developed to achieve real-time realistic face capture and composition with a pair of consumer-level RGBD cameras. FaceCollage has a number of key advantages over existing systems for on-the-go telepresence:

(1) It is *very simple to deploy*, which is our key requirement. The user only needs to casually place the two RGBD cameras facing the user, as is typified by Fig. 1. Thereafter, our system will automatically register the RGBD face data from the two cameras in around 15-30 seconds. Neither user intervention nor placement of markers (e.g., checkerboard or QR code) is required in our registration process. We are not aware of any other multi-sensor 3D telepresence system that has this level of simplicity in deployment.
(2) Once the devices have been deployed and calibrated, our method can effectively integrate the two hybrid color&depth streams acquired from the cameras and generate a live 3D rendering, covering a wider range of a dynamic face in 3D.
(3) Our method preserves the appearance and expression of user's face in real-time. In addition, the system can automatically adapt to sudden movement and progressively restore the reconstructed face from changes, which is not possible in many existing systems.

In terms of technical contributions, this paper incorporates three: (1) We have a fully automatic registration method by iteratively refining and registering a set of dense correspondences (without using reference patterns such as checkerboard and QR code) between the partial face models that the two depth cameras captured. (2) We leverage parallel computation on the GPU to support real-time boundary tracking, cross geometry filtering, and texture blending of the RGBD face data. By adopting CUDA acceleration for geometry processing and GLSL controls in the graphics hardware, we can capture the appearance and expression of human face with real-time performance (e.g., 25 fps). (3) we build an integrated 3D telepresence system and perform several experiments: robustness of the face registration and system performance. Our system has also been successfully tested with twelve participants with decent results.

2 RELATED WORK

We first review relevant telepresence systems and then methods for face acquisition and reconstruction.

Pioneering work in 3D telepresence. Raskar et al. [26] proposed a semi-immersive system by integrating image-based acquisition and projector displays to create a collaborative office environment; they used an imperceptible structured light method to capture depth and reflectance of visible objects and participants. Gross et al. [14] developed a fully-immersive system (multiple calibrated cameras, LCD projectors, active illuminations, and head-mounted shutter glasses) to capture the whole body of a user in a CAVE$^{\text{TM}}$-like environment [10]. Matusik et al. [22] built an end-to-end 3D TV system that can support multiple users without requiring special glasses. While these systems offer immersive 3D experience, they are heavily dependent on high-end equipment, careful engineering and user calibration.

3D telepresence with depth sensors. Recently, several cost-effective solutions have been proposed by using low-cost commodity depth sensors like Kinect [1] and Prime Sense [2]. Maimone et al. [20] developed a system with six Kinects, where the color and depth streams from each Kinect were individually processed and blended together to form a dynamic 3D scene. Kuster et al. [17] introduced a similar framework with two Kinects and three RGB cameras; an online segmentation method is developed to restrict the processing and rendering on foreground objects. Zhang et al. [32] considered how to present 3D content to remote users, so that users can virtually see one another as if they shared a common physical space.

While these systems presented visually-pleasing results, they did not carefully consider the temporal coherency of the surface geometry and texture, thus usually leading to undesired distraction during active use. More recently, Kuster et al. [16] proposed a spatio-temporal fusion method to reconstruct a point set model of a dynamic scene; however, their method is not real-time yet. Moreover, all these systems required tremendous engineering effort, such as deployment and calibration of the fixed devices, which are nontrivial for general users.

Face capture by structured light and passive stereo. Realistic face capture with rich facial expressions has been an active research topic. Early representative works such as structured light systems [30, 33] can capture the geometry of a dynamic face in real-time. These systems were similar to Kinect-based methods [36] since Kinect V1 is inherently a structured light sensor. However, the reconstruction results were usually of low resolution and cannot sufficiently cover the human face.

Bradley et al. [6] developed a passive face capture system which utilized flow-based surface tracking for reconstructing a high resolution time-varying textured face. Beeler et al. [4] devised a similar single-shot capture method targeting at industrial-quality applications in movies and computer games, later extended to handle temporal anchor frames [5]. Although these works offered high-quality face capture with wrinkles and pores, they were restricted to studio settings and offline processing, so are not suitable for real-time 3D telepresence.

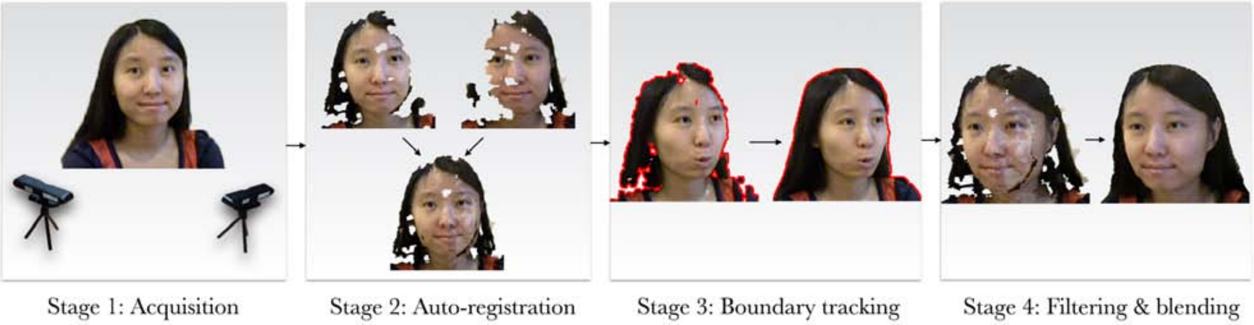
Stage 1: Acquisition
Stage 2: Auto-registration
Stage 3: Boundary tracking
Stage 4: Filtering & blending

Figure 2: Overview of the FaceCollage system: it consists of four key computational stages (see above), from RGBD acquisition to final rendering.

Template-based face reconstruction. Instead of reconstructing the geometry of user's face from 3D depth information, template-based methods such as [31] focus on approximating the user's face by fitting the 3D depth data with a pre-trained general 3D face template model. While state-of-the-art methods [8, 13, 18] can produce fabulous facial animations in real-time, these methods do not preserve the varying texture and hair of the user since these were not part of the template model. Recently, Zollhofer and his colleagues [37] proposed a real-time non-rigid 3D face reconstruction method using a pre-captured 3D face model of the user. However, since the method relied on a pre-capturing process, it cannot handle dramatic user movement.

Template-free dynamic reconstruction. Very recently, the DynamicFusion [23] and VolumeDeform [15] systems demonstrated the superiority of volumetric representation for real-time dynamic reconstruction. While impressive geometric reconstructions were presented, these systems did not consider the preservation of the original texture information, which is a critical concern in telepresence applications. Also, given the requirement of estimating a growing warp field, they cannot handle highly dynamic shapes.

3 OVERVIEW

Fig. 2 presents an overview of our system, which has the following four computational stages:

Stage 1: Acquisition setup. We use two Carmine 1.09 RGBD cameras from PrimeSense [2]. Typically, these two cameras are placed on the left and right hand sides of the display with the same baseline; see Fig. 1. Other than tripods, we may also simply mount the cameras on top of a laptop. Note that each camera can only observe partial views of the user's head, due to significant self-occlusion. Once installed, our system receives two streams of RGBD videos in VGA resolution (640 × 480) at 25fps from the cameras. Note that since our current setup can provide plausible frame rate, we do not need to perform any synchronization.

Stage 2: Auto-registration. Instead of using a tedious calibration process, e.g., using marker patterns, we have an automatic registration process, that requires minimal input from the user. First, we extract two sets of sparse face features simultaneously from the respective RGBD streams using an image-based face tracker [28]. Using these sparse features, we obtain an initial rough alignment between the two partial RGBD head models. To improve the alignment quality, we propose to estimate the front-facing direction of a user's face in a common 3D space and obtain dense correspondences between the two face models using a flow-based method [7]. We can then refine the correspondences and obtain a high-quality alignment automatically between the two face models (see Fig. 2 stage 2). Except for the initial alignment, which runs on the CPU, the rest of our method was implemented in CUDA to run on the GPU, in order to achieve real-time performance.

Stage 3: Depth-assisted boundary tracking. To suppress visual artifacts resulted from missing data and inaccurate depth acquisition around the face silhouette, we attempt to maintain a temporally-consistent boundary, subject to a computational constraint for real-time performance (see Fig. 2 stage 3). We progressively track and propagate an evolving boundary around user's face in each RGBD stream by adapting and extending a state-of-the-art curve evolving method [21]. In particular, we incorporate depth information into the framework not only to provide an initial estimate but also to constrain the curve evolution, so that our method can restore the boundary curve even after fast and large deformation.

Stage 4: Cross geometry filtering and texture blending. We further improve the quality of the composed face model by combining and completing the geometry and texture of the left and right face models. Instead of processing the two models separately [17, 20], we adopt a cross-filtering scheme and use temporal processing to suppress flickering artifacts. We also exploit redundancy in spatiotemporal data to fill gaps in the face composition as well as to correct geometrical errors. The rendered textures are composed in a geometry-aware manner to further enhance quality, and subsequently displayed.

The individual stages are described in more detail below.

4 STAGE II: AUTO-REGISTRATION

We employ OpenNI2 [24] to obtain an RGBD video from each of the two depth cameras. The main goal of this stage is to determine the extrinsic transformation between the two cameras, so that we can better align the point clouds of the partial head models extracted from the cameras. In particular, we aim to avoid tedious calibration

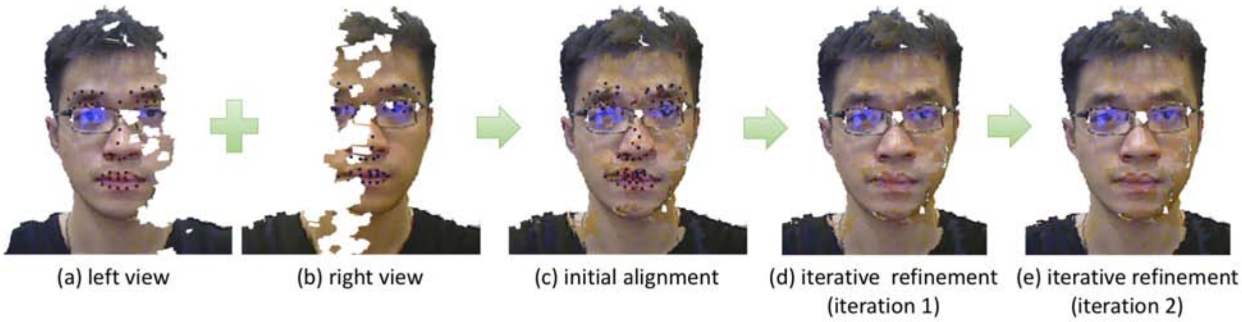
(a) left view
(b) right view
(c) initial alignment
(d) iterative refinement (iteration 1)
(e) iterative refinement (iteration 2)

Figure 3: Face alignment in Stage 2: (a&b) the partial face models captured from the left and right views; (c) initial alignment result; and (d&e) iterative refinement after the first and second iterations. Although face features may not be detected properly (see the eye regions in this example), we can still produce an initial registration, good enough as an initialization for iterative refinement. Also note the improvement after the iterative refinement, e.g., around the lip region.

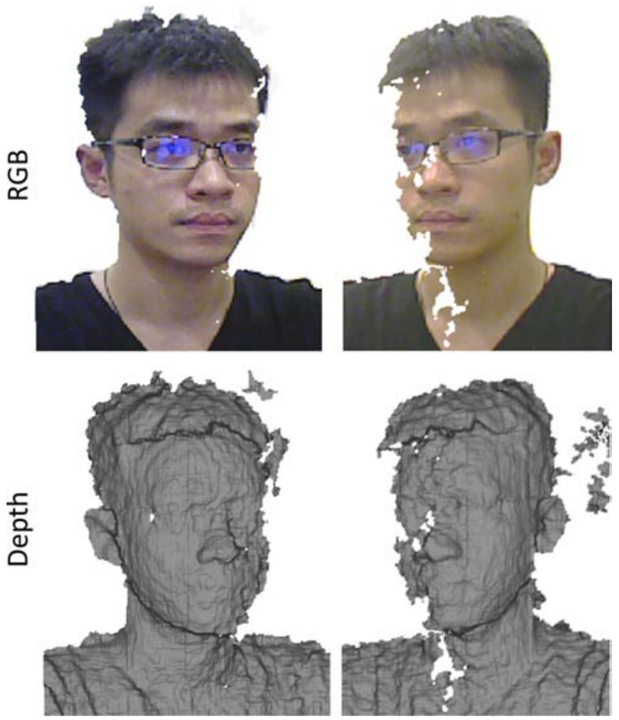

Figure 4: The raw partial face models (RGB & depth) captured from the left and right views.

with designated patterns [34] (e.g., checkerboard). Instead, we take a 3D registration approach. Our key idea is to calibrate the two cameras by directly registering the two partial 3D face models using an iterative refinement method based on the face features. Our calibration is fully automatic with high performance. It consists of the following three main steps.

Step 1: Face Data Collection and Preprocessing. The auto-registration process starts by extracting 30 consecutive RGBD frames from each depth camera (view). For each view, our method first locates the face region by thresholding with depth and merges the frames into a single depth image by averaging. Next, we reproject the face pixels in each depth image (left & right) to 3D as point clouds (partial face models) using the intrinsic camera parameters provided by OpenNI2, see Fig. 4 for results. To initiate the auto-registration process, the user only need to click one button in the user interface, while facing both cameras and remaining still for one to two seconds to avoid motion blur in the meantime.

Step 2: Initial Face Alignment. Registering two 3D point clouds with partial overlap without any prior information is a nontrivial problem. Fortunately, we have a strong prior that faces are a key part of the head models we want to register, and thus we can initialize the registration by taking advantages of various sparse face features extracted from the models.

To do so, we propose to first fit a 2D face template to each color image from the two cameras by [28]. We can then extract a set of face features per color image and compute the correspondence between face pixels in the two color images by using the extracted sparse face features. Since there is a one-to-one correspondence between the color pixels and the point cloud, we can further estimate an initial alignment (a rigid transformation) between the two partial face models by an SVD method [27], see Fig. 3(a-c) for an example.

Step 3: Iterative Refinements. Due to depth noise and sparsity of face features, registration using only facial features is not sufficient (see Fig. 3(c)). To overcome this problem, we propose to further find dense correspondences to refine the alignment. In particular, we first determine the front view, at which the rendered images from both left and right head models will have large overlap. This is done by applying PCA on those 3D points around face centers, where the smallest eigenvector of the covariance matrix is chosen as the front view direction of the face $\mathcal{D}$.

Given the front view direction $\mathcal{D}$, we then average the 30 color images collected from each view in step 1 to obtain a mean image per view. We map these images to the corresponding point cloud, and render each point cloud along $\mathcal{D}$: $\boldsymbol{I}_L^{\mathcal{D}}$ and $\boldsymbol{I}_R^{\mathcal{D}}$ are the resulting images from the left and right views, respectively (see Fig. 3(a&b)), while $\boldsymbol{Z}_L^{\mathcal{D}}$ and $\boldsymbol{Z}_R^{\mathcal{D}}$ are the corresponding point clouds. It can be seen from Fig. 3(a&b) that there is a large number of missing face pixels, which is either due to partial camera view or caused by the noise and interference. These missing pixels can be easily identified by looking for pixels with invalid depth values. To stabilize the registration, we make use of the correspondence $\boldsymbol{I}_L^{\mathcal{D}} \leftrightarrow \boldsymbol{Z}_L^{\mathcal{D}} \leftrightarrow \boldsymbol{Z}_R^{\mathcal{D}} \leftrightarrow \boldsymbol{I}_R^{\mathcal{D}}$ to fill invalid pixels in $\boldsymbol{I}_L^{\mathcal{D}}$ with pixel values from associated locations in $\boldsymbol{I}_R^{\mathcal{D}}$, and vice versa.

Next, we compute dense correspondences between $\boldsymbol{I}_L^{\mathcal{D}}$ and $\boldsymbol{I}_R^{\mathcal{D}}$ by minimizing the Horn-Schunck energy [7]:

$$\begin{aligned} E(\boldsymbol{w}) \quad = \quad & \int_{\Omega} \boldsymbol{\psi}(|\boldsymbol{I}_L^{\mathcal{D}}(\boldsymbol{x}+\boldsymbol{w}) - \boldsymbol{I}_R^{\mathcal{D}}(\boldsymbol{x})|^2)\ d\boldsymbol{x} + \\ & \int_{\Omega} \alpha\boldsymbol{\psi}(|\nabla \boldsymbol{I}_L^{\mathcal{D}}(\boldsymbol{x}+\boldsymbol{w}) - \nabla \boldsymbol{I}_R^{\mathcal{D}}(\boldsymbol{x})|^2)\ d\boldsymbol{x} + \\ & \int_{\Omega} \beta|\nabla \boldsymbol{w}|^2\ d\boldsymbol{x}\ , \end{aligned} \tag{1}$$

where Ω is the image domain, $\boldsymbol{w}$ is the optimal displacement field (dense correspondences) between $\boldsymbol{I}_L^{\mathcal{D}}$ and $\boldsymbol{I}_R^{\mathcal{D}}$, $\boldsymbol{\psi}$ is a nonlinear mapping function, and α and β are tradeoff parameters. With dense correspondences, we can then apply the SVD method [27] again to obtain a more accurate rigid transformation. Note that we use all pixels in the optimization of (1) to enforce regularization, but only use the correspondences of the valid pixels to refine the alignment.

We perform the refinement iteratively. Since the initial face alignment has already brought the two partial head models close enough to each other, we found that two to three iterations are sufficient to produce good alignment results (see Fig. 3(d&e)). In our system, we implemented a CUDA program to compute the optimal displacement field in parallel, so as to efficiently perform the warping flow procedure [7]. We set α=0.1, β=0.1, $\boldsymbol{\psi}(s^2)$=$\sqrt{s^2+\epsilon^2}$, and $\epsilon = 0.001$ in (1) for all experiments.

5 STAGE III: DEPTH-ASSISTED BOUNDARY TRACKING

In addition to the problem of noisy depth measurements delivered by commodity depth sensors, another problem typically encountered is that object silhouettes in depth and color images are not accurately aligned [16], which often causes annoying flicking effects around head silhouettes. Theoretically, we can adopt off-the-shelf real-time RGBD segmentation methods to tackle this problem. However, generic methods such as [35] often require large amounts of computing resource, hindering the overall performance of our system. This provides motivation for developing an effective and efficient boundary tracking method tailored for our application, which can locally re-align the depth and the color images in each camera view across the time.

In our study, we observe that the non-alignment artifacts usually occur in a narrow spatial band near the true silhouette of the head, where reliable textures are available. This led to our proposal to evolve the curve of the depth silhouette towards the local contour of the color image, subject to constraints such as depth discontinuity. In particular, we adopt a recent active contour model [21], wherein the curve C is implicitly represented as the zero crossing of a level set function, $C(t) = \{\boldsymbol{x}|u(\boldsymbol{x},t) = 0\}$, and evolved by minimizing the GAC energy functional [9]:

$$E(C) = \int_0^{length(C)} g(I)(C(s))ds\ , \tag{2}$$

where $g(I)$ is an edge detection function defined as

$$g(I) = \frac{1}{\sqrt{1+\alpha|\nabla I|}}\ , \tag{3}$$

with $\alpha = 1000$. The local minima to (2) is reached at the steady states of the differential equation:

$$\frac{\partial u}{\partial t} = (\mathcal{K} + v) \cdot g(I)|\nabla u| + \nabla g(I)\nabla u\ , \tag{4}$$

where $\mathcal{K} = div(\frac{\nabla u}{|\nabla u|})$ is the Euclidean curvature of C, and $v = 1$ is a balloon weight. Eq. (2) can be solved numerically by successively applying the flow given by (4), while the $v \cdot g(I)|\nabla u|$ term and the

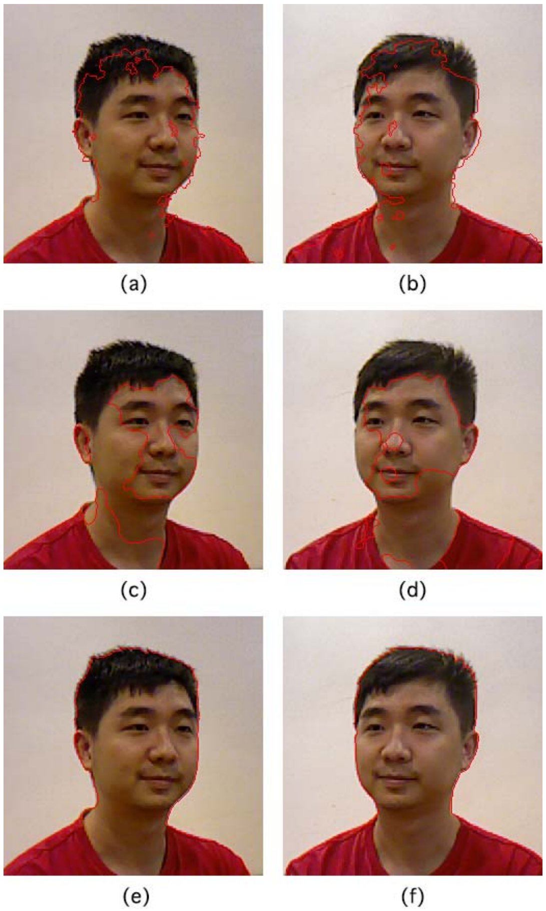

Figure 5: Boundary tracking results in Stage 3: (a&b) extracted by simply thresholding the raw depth input; (c&d) results with pure [21], produced by their released source codes; and (e&f) depth-assisted contour tracking.

$\mathcal{K} \cdot g(I)|\nabla u|$ term can be approximated by a dilation operator and a discrete morphological curvature operator, respectively [21].

While the morphological approach is simple and fast, it is only applied to textures. Since depth is available, we extend the method to handle the captured RGBD data with the following additional modifications:

1) To generate the initial curve $u_{n+1}(\boldsymbol{x}, 0)$ in the (n+1)-th frame, we first compute a binary mask $M_{n+1}(\boldsymbol{x})$ by thresholding the input depth map, and then set

$$u_{n+1}(\boldsymbol{x}, 0) = \frac{1}{2}\left(M_{n+1}(\boldsymbol{x}) + u_n(\boldsymbol{x})\right) , \tag{5}$$

where $u_n(\boldsymbol{x})$ is the output curve of the n-th frame. In this way, we will not only incorporate depth information into the evolution of the curve, but also obtain temporally more consistent results across frames.

2) Instead of the morphological curvature operator, we use a joint bilateral filter over $u(\boldsymbol{x})$ to smooth the curve with both color and depth information.

Fig. 5 shows some boundary tracking results. We can see that compared to purely applying [21], our method is more robust and less sensitive to the initialization.

6 STAGE IV: CROSS-GEOMETRY FILTERING AND TEXTURE BLENDING

Given the refined boundary from the previous stage, the purpose of this stage is to remove artifacts and fill in the holes within the boundary, as well as to combine the geometry and textures of the two views in a consistent manner. This stage consists of the following three steps. Note that, these three steps are only applied to pixels within the region inside the refined boundary. This can be achieved by incorporating a masking operation in each step.

Step 1: Geometry preprocessing for removing artifacts. Due to the low output quality of commodity sensors, the raw depth data exhibits obvious artifacts such as isolated patches and holes. To remove isolated patches, our method first examines a small local neighborhood around each pixel. Specifically, for each pixel p_i, we extract the 3D coordinates P_j of pixels within an $N \times N$ window around p_i, and count the number of points C_i that are at a distance between r_{in} and r_{out} from P_i (the 3D coordinates of p_i):

$$C_i = \frac{1}{|N(P_i)|} \sum_{P_j \in N(P_i)} I(r_{in} < \|P_j - P_i\| < r_{out}) , \tag{6}$$

where $I(\cdot)$ is an indicator function and $N(P_i)$ is a set of valid pixels within the local window around p_i. Intuitively, if p_i belongs to an isolated patch, C_i is small. Thus, if C_i is smaller than a preset threshold C_t, we will remove p_i. In all our experiments, we set $N = 41$, $r_{in} = 0.02$, $r_{out} = 0.08$, and $C_t = \frac{100}{N \cdot N} \approx 0.06$. Fig. 6 shows representative results of the isolated patch removal substep.

To fill the missing data caused by the sensors and the removal of isolated patches, we observe that holes in one view can be reasonably compensated using data from the other view. Thus, our method synthesizes a new depth map $D_{\tilde{v}_0}$ under view v_0 by rendering the geometry of its counterpart view using the calibration parameters estimated in Section 4. The missing data in the raw depth map D_{v_0} under view v_0 can then be filled in by using $D_{\tilde{v}_0}$. Fig. 6 shows some example results of hole filling. Any remaining unfilled hole will be left for the subsequent step to handle.

Step 2: Cross filtering for geometry consistency. At this point we have two well-aligned depth maps with reasonable but not yet the best possible quality. We further reinforce the spatial and temporal consistency by applying a cross-filtering process in geometry. For each view V, we denote the preprocessed depth map at time t as $\mathbf{d}_V(\mathbf{x}, t)$ and texture as $\mathbf{c}_V(\mathbf{x}, t)$. Similar to the idea introduced in the hole-filling step, our method warps the depth map and the texture of the other view to V to obtain $\mathbf{d}_{\tilde{V}}(\mathbf{x}, t)$ and $\mathbf{c}_{\tilde{V}}(\mathbf{x}, t)$. Our method performs a spatial-temporal filtering as

$$\begin{aligned} \mathbf{f}_V(\mathbf{x}, t) &= \frac{1}{N_{(\mathbf{x},t)}} \sum_{(\mathbf{x}', t') \in N_{(\mathbf{x},t)}} (w_c \; w_d \; w_s \; \mathbf{d}_V(\mathbf{x}', t') \\ &\quad + \tilde{w}_c \; \tilde{w}_d \; \tilde{w}_s \; \mathbf{d}_{\tilde{V}}(\mathbf{x}', t')), \end{aligned} \tag{7}$$

where

$$\begin{aligned} w_c &= \exp(-\frac{\mathbf{c}_V(\mathbf{x}', t') - \mathbf{c}_V(\mathbf{x}, t)}{2\sigma_c^2}) , \\ \tilde{w}_c &= \exp(-\frac{\mathbf{c}_{\tilde{V}}(\mathbf{x}', t') - \mathbf{c}_V(\mathbf{x}, t)}{2\sigma_c^2}) , \\ w_d &= \exp(-\frac{\mathbf{d}_V(\mathbf{x}', t') - \mathbf{d}_V(\mathbf{x}, t)}{2\sigma_d^2}) , \end{aligned}$$

$$\begin{aligned} \tilde{w}_d &= \exp(-\frac{\mathbf{d}_{\tilde{V}}(\mathbf{x}', t') - \mathbf{d}_V(\mathbf{x}, t)}{2\sigma_d^2}) , \text{ and} \\ w_s &= \tilde{w}_s = \exp(-\frac{\|\mathbf{x}' - \mathbf{x}\|^2 + (t' - t)^2}{2\sigma_s^2}) . \end{aligned}$$

In Eq. (7), $N_{(\mathbf{x},t)}$ is the weight normalization term, w_c, w_d and w_s are the color range kernel, the depth range kernel and the spatial-temporal kernel, respectively, similar to the joint bilateral filter, while $\tilde{w}_c$, $\tilde{w}_d$ and $\tilde{w}_s$ are the respective kernels for the wrapped depth map from the other view. Hence, our cross-view filtering method performs a volume filtering in spatial-temporal depth space, incorporating the geometry and texture of both views. Here, we set the spatial window size of the filtering to 5, the temporal window size to 3, and the kernel sizes $\sigma_c = 0.05$ (within a normalized color range of [0,1]), $\sigma_d = 0.006$ and $\sigma_s = 1.0$ for all experiments.

Step 3: Alpha blending for texture consistency. Finally, we adopt the standard view-dependent alpha blending technique to blend the textures from both views:

$$I(P) = \frac{w_1 \cdot I_l(P) + w_2 \cdot I_r(P)}{w_1 + w_2}, \tag{8}$$

$$w_1 = (n_P \circ v_l)^2 , \tag{9}$$

$$w_2 = (n_P \circ v_r)^2 , \tag{10}$$

where $I(P)$ represents the output at position P from the fragment shader, $I_l(P)$ and $I_r(P)$ are the texture inputs at position P, n_P is the unit normal of position P, and v_l and v_r are the unit direction vectors of the two cameras.

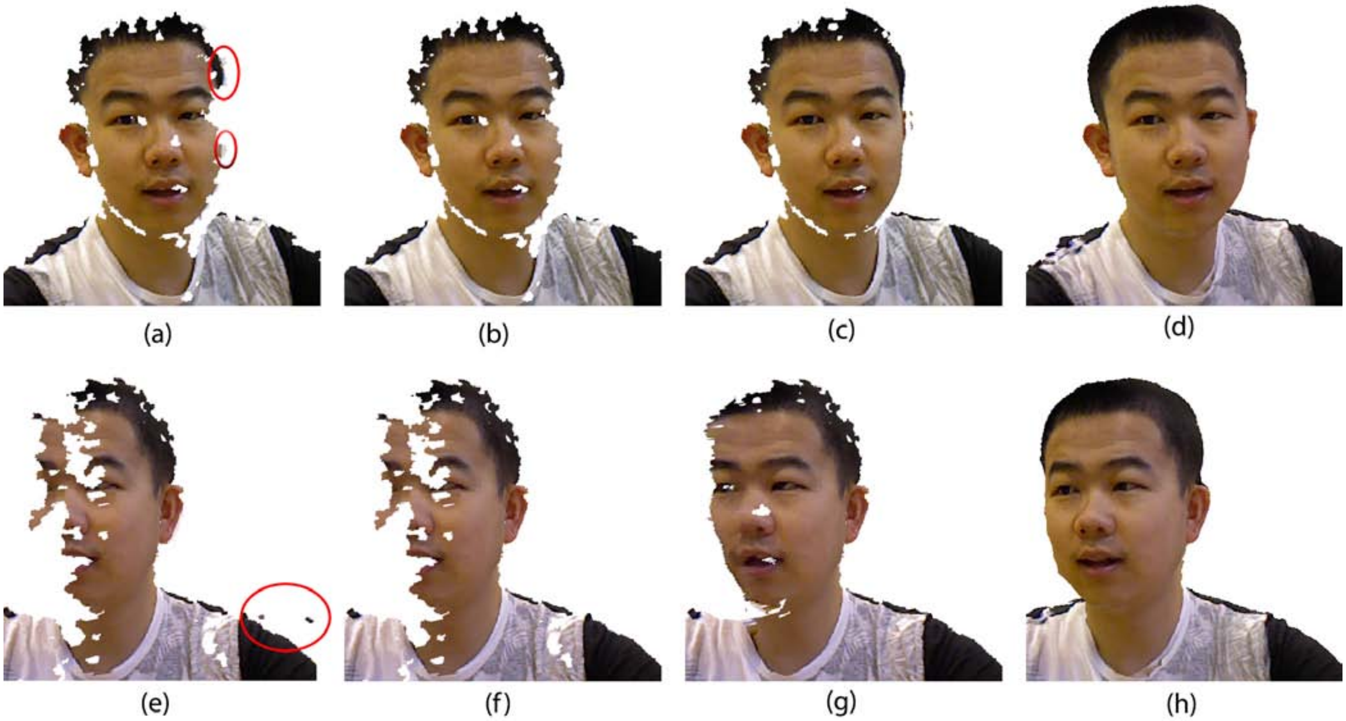

Figure 6: Filtering and texture blending: (a&e) the partial face models before boundary refinement. Please see the inconsistent boundary and the isolated patches circled in red; (b&f) the partial face models after refining the boundary and removing the isolated patches; (c&g) the partial face models after hole filling; and (d&h) the collaged face models after cross filtering and texture blending.

7 EXPERIMENTAL RESULTS

7.1 Implementation

As stated previously, most components in our pipeline were programmed and run on the GPU, except the initial face alignment, which is step 2 in Stage 2. Specifically, we used the OpenNI2 package to read data streams from the sensors, and implemented the rest of the steps of Stage 2, Stage 3 and the cross-geometry filtering step of Stage 4 in CUDA, while the alpha blending step was implemented in GLSL. Our system successfully ran on two systems: i) a desktop with an Intel Core i7 CPU, 16GB RAM and a GTX TITAN GPU, and also ii) an Apple MacBook Pro 2013 with a GeForce GT 650m. The results presented in this section are produced using the desktop system, which achieves a frame rate of 25 fps, while the laptop system runs at 15 fps with similar quality in the results.

7.2 Results

We tested our system with 12 participants. The results of individual components have been shown in Figs. 3, 5 and 6. Representative visual results for the final outputs are shown in Fig. 7. It can be seen that our system achieves decent visual quality for the collaged 3D faces from stereo depth cameras. *To best appreciate the actual experience of setting up and using this system, please refer to the supplementary video for more results as well as records of a live demonstration.* To the best of our knowledge, we are not aware of any other existing system that can be rapidly deployable and yet achieves real-time 3D head reconstruction (particularly from multiple RGBD video streams) with similar quality.

Time performance. We also evaluated the average execution time of each key component in our system, as shown in Table 1. For the auto-registration part, the actual execution time varied between 13 - 32 seconds in our experiments due to different numbers of iterations required. However, the user only need to do this step once after putting up the system. After the registration, our system processes each frame in 39.56 ms and achieved 25 frames per second with an output resolution of 1024x768. We encourage the reader to watch the supplementary video, which demonstrates the efficiency and the robustness of our system.

Limitations. There are still several limitations in our current system that may inspire future work. First, the usage of commodity sensors limits the display resolution and RGB quality captured. Second, in a few rare cases (e.g., poor lighting condition) when the user's face cannot be successfully detected, our system could fail to register the data from the two RGBD sensors. This can be further improved by adopting more robust face detection and tracking methods that incorporate not only textural but also geometrical cues [29]. Third, however efficient, the boundary tracking method cannot handle scenes with excessively complex background. Also, when handling dramatic user movement, our method exhibits synchronization artifacts before restoring the reconstructed face. This

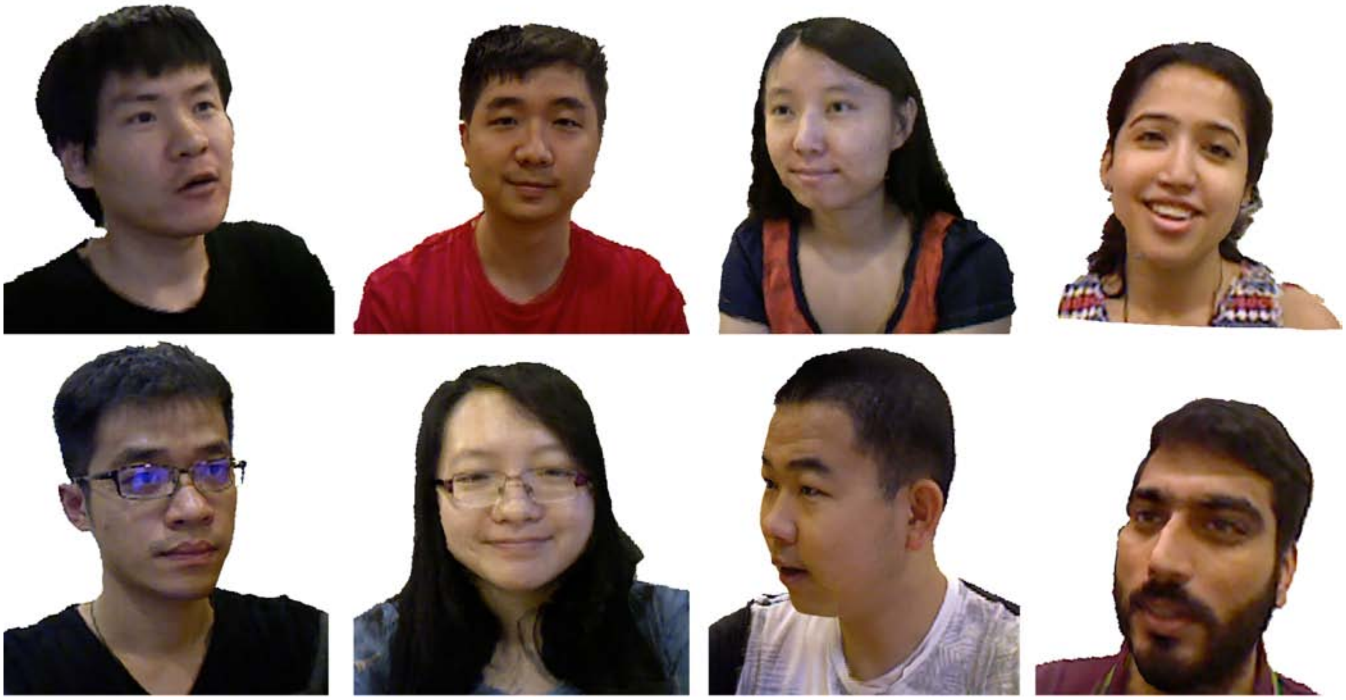

Figure 7: Qualitative examples of the final output of our system from different viewpoints. Please refer to the supplementary video for animated versions of these results.

	One-time	Online		
Stage	Auto-registration	Boundary tracking	Cross filtering	Alpha blending
Time (ms)	19977.8	26	8.3	5.26

Table 1: Average running time of each key component in our system; note that auto-registration is a one-time process.

is a common problem for per-frame reconstruction methods, which deserve further research.

8 CONCLUSION

In this paper, we presented FaceCollage, a robust and real-time method for 3D telepresence using a pair of consumer-grade RGBD cameras. Our method is built upon a series of four carefully-designed working stages: data acquisition, auto-registration, depth-assisted boundary tracking, and cross-geometry filtering and texture blending. These stages are mostly implemented and developed on the GPU to best trade-off between quality and real-time performance. To demonstrate the efficiency and robustness of our work, we built a prototype system and tested it with twelve participants. Our system shows very promising results that have not be seen in any of the existing consumer-grade RGBD cameras based real-time face composition systems. It provides a new solution, which is robust and affordable, for personal 3D telepresence.

ACKNOWLEDGMENTS

The authors would like to thank the reviewers for the various valuable comments. This research is supported by the **BeingTogether Centre**, a collaboration between Nanyang Technological University (NTU) Singapore and University of North Carolina (UNC) at Chapel Hill. The **BeingTogether Centre** is supported by the National Research Foundation, Prime Minister's Office, Singapore under its International Research Centres in Singapore Funding Initiative. C.-W. Fu was supported by the Shenzhen Science and Technology Program (JCYJ20170413162617606).

REFERENCES

[1] 2010. Kinect for Windows. (2010). http://www.microsoft.com/en-us/kinectforwindows/

[2] 2013. Carmine 1.09 3D-Sensor. (2013). http://www.primesense.com/solutions/3d-sensor

[3] S. Beck, A. Kunert, A. Kulik, and B. Froehlich. 2013. Immersive Group-to-Group Telepresence. *IEEE Transactions on Visualization and Computer Graphics* 19, 4 (2013), 616–625.

[4] Thabo Beeler, Bernd Bickel, Paul Beardsley, Bob Sumner, and Markus Gross. 2010. High-Quality Single-Shot Capture of Facial Geometry. *ACM Transactions on Graphics (SIGGRAPH)* 29, 3 (2010), 40:1–40:9.

[5] Thabo Beeler, Fabian Hahn, Derek Bradley, Bernd Bickel, Paul Beardsley, Craig Gotsman, Robert W. Sumner, and Markus Gross. 2011. High-quality passive facial performance capture using anchor frames. *ACM Transactions on Graphics (SIGGRAPH)* 30 (2011), 75:1–75:10.

[6] Derek Bradley, Wolfgang Heidrich, Tiberiu Popa, and Alla Sheffer. 2010. High Resolution Passive Facial Performance Capture. *ACM Transactions on Graphics (SIGGRAPH)* 29, 3 (2010).

[7] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. 2004. High Accuracy Optical Flow Estimation Based on a Theory for Warping. In *European Conference on Computer Vision (ECCV)*. 25–36.

[8] Chen Cao, Yanlin Weng, Stephen Lin, and Kun Zhou. 2013. 3D Shape Regression for Real-time Facial Animation. *ACM Transactions on Graphics (SIGGRAPH)* 32, 4 (2013), 41:1–41:10.

[9] V. Caselles, R. Kimmel, and G. Sapiro. 1995. Geodesic active contours. In *IEEE International Conference on Computer Vision (ICCV)*. 694–699.

[10] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. 1993. Surround-screen Projection-based Virtual Reality: The Design and Implementation of the CAVE. In *SIGGRAPH*. 135–142.

[11] Teng Deng, Jianfei Cai, Tat-Jen Cham, and Jianmin Zheng. 2017. Multiple consumer-grade depth camera registration using everyday objects. *Image and Vision Computing* 62 (2017), 1 – 7.

[12] Mingsong Dou and Henry Fuchs. 2014. Temporally enhanced 3D capture of room-sized dynamic scenes with commodity depth cameras. In *IEEE Virtual Reality (VR)*. 39–44.

[13] Pablo Garrido, Levi Valgaert, Chenglei Wu, and Christian Theobalt. 2013. Reconstructing Detailed Dynamic Face Geometry from Monocular Video. *ACM Transactions on Graphics (SIGGRAPH ASIA)* 32, 6 (2013), 158:1–158:10.

[14] Markus Gross, Stephan Würmlin, Martin Naef, Edouard Lamboray, Christian Spagno, Andreas Kunz, Esther Koller-Meier, Tomas Svoboda, Luc Van Gool, Silke Lang, Kai Strehlke, Andrew Vande Moere, and Oliver Staadt. 2003. Blue-c: A Spatially Immersive Display and 3D Video Portal for Telepresence. *ACM Transactions on Graphics (SIGGRAPH)* 22, 3 (2003), 819–827.

[15] Matthias Innmann, Michael Zollhöfer, Matthias Nießner, Christian Theobalt, and Marc Stamminger. 2016. VolumeDeform: Real-time Volumetric Non-rigid Reconstruction. In *Proc. Eur. Conf. Computer Vision (ECCV)*.

[16] Claudia Kuster, Jean-Charles Bazin, Cengiz Öztireli, Teng Deng, Tobias Martin, Tiberiu Popa, and Markus Gross. 2014. Spatio-Temporal Geometry Fusion for Multiple Hybrid Cameras using Moving Least Squares Surfaces. *Computer Graphics Forum (Eurographics)* 33, 2 (2014), 1–10.

[17] C. Kuster, T. Popa, C. Zach, C. Gotsman, and M. Gross. 2011. FreeCam: A Hybrid Camera System for Interactive Free-Viewpoint Video. In *International Symposium on Vision, Modeling and Visualization (VMV)*.

[18] Hao Li, Jihun Yu, Yuting Ye, and Chris Bregler. 2013. Realtime Facial Animation with On-the-fly Correctives. *ACM Transactions on Graphics (SIGGRAPH)* 32, 4 (2013).

[19] Andrew Maimone, Jonathan Bidwell, Kun Peng, and Henry Fuchs. 2012. Enhanced personal autostereoscopic telepresence system using commodity depth cameras. *Computers & Graphics* 36, 7 (2012), 791 – 807.

[20] A. Maimone and H. Fuchs. 2011. Encumbrance-free telepresence system with real-time 3D capture and display using commodity depth cameras. In *IEEE International Symposium on Mixed and Augmented Reality*. 137–146.

[21] P. Marquez-Neila, L. Baumela, and L. Alvarez. 2014. A Morphological Approach to Curvature-Based Evolution of Curves and Surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 36, 1 (2014), 2–17.

[22] Wojciech Matusik and Hanspeter Pfister. 2004. 3D TV: A Scalable System for Real-time Acquisition, Transmission, and Autostereoscopic Display of Dynamic Scenes. *ACM Transactions on Graphics (SIGGRAPH)* 23, 3 (2004), 814–824.

[23] R. A. Newcombe, D. Fox, and S. M. Seitz. 2015. DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition (CVPR)*. 343–352.

[24] Occipital, Inc. 2013. *OpenNI Programmer's Guide*. Occipital, Inc., San Francisco, CA.

[25] Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L. Davidson, Sameh Khamis, Mingsong Dou, Vladimir Tankovich, Charles Loop, Qin Cai, Philip A. Chou, Sarah Mennicken, Julien Valentin, Vivek Pradeep, Shenlong Wang, Sing Bing Kang, Pushmeet Kohli, Yuliya Lutchyn, Cem Keskin, and Shahram Izadi. 2016. Holoportation: Virtual 3D Teleportation in Real-time. In *ACM Symposium on User Interface Software and Technology (UIST)*. 741–754.

[26] Ramesh Raskar, Greg Welch, Matt Cutts, Adam Lake, Lev Stesin, and Henry Fuchs. 1998. The Office of the Future: A Unified Approach to Image-based Modeling and Spatially Immersive Displays. In *SIGGRAPH*. 179–188.

[27] S. Rusinkiewicz and M. Levoy. 2001. Efficient variants of the ICP algorithm. In *International Conference on 3-D Digital Imaging and Modeling*. 145–152.

[28] J.M. Saragih, S. Lucey, and J.F. Cohn. 2009. Face alignment through subspace constrained mean-shifts. In *IEEE International Conference on Computer Vision (ICCV)*. 1034–1041.

[29] Lu Sheng, Jianfei Cai, Tat-Jen Cham, Vladimir Pavlovic, and King Ngi Ngan. 2017. A Generative Model for Depth-based Robust 3D Facial Pose Tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[30] Yang Wang, Xiaolei Huang, Chan-Su Lee, Song Zhang, Zhiguo Li, Dimitris Samaras, Dimitris Metaxas, Ahmed Elgammal, and Peisen Huang. 2004. High resolution acquisition, learning and transfer of dynamic 3-D facial expressions. *Computer Graphics Forum (Eurographics)* 23, 3 (2004), 677–686.

[31] Thibaut Weise, Sofien Bouaziz, Hao Li, and Mark Pauly. 2011. Realtime Performance-Based Facial Animation. *ACM Transactions on Graphics (SIGGRAPH)* 30, 4 (2011).

[32] Cha Zhang, Qin Cai, P.A. Chou, Zhengyou Zhang, and R. Martin-Brualla. 2013. Viewport: A Distributed, Immersive Teleconferencing System with Infrared Dot Pattern. *IEEE Multimedia* 20, 1 (2013), 17–27.

[33] Li Zhang, Noah Snavely, Brian Curless, and Steven M. Seitz. 2004. Spacetime Faces: High Resolution Capture for Modeling and Animation. *ACM Transactions on Graphics (SIGGRAPH)* 23, 3 (2004), 548–558.

[34] Zhengyou Zhang. 2000. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 22, 11 (2000), 1330–1334.

[35] M. Zhao, C. W. Fu, J. Cai, and T. J. Cham. 2015. Real-Time and Temporal-Coherent Foreground Extraction With Commodity RGBD Camera. *IEEE Journal of Selected Topics in Signal Processing* 9, 3 (April 2015), 449–461.

[36] Michael Zollhöfer, Michael Martinek, Günther Greiner, Marc Stamminger, and Jochen Süßmuth. 2011. Automatic Reconstruction of Personalized Avatars from 3D Face Scans. *Computer Animation and Virtual Worlds (CASA)* 22, 2-3 (2011), 195–202.

[37] Michael Zollhöfer, Matthias Nießner, Shahram Izadi, Christoph Rhemann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew Fitzgibbon, Charles Loop, Christian Theobalt, and Marc Stamminger. 2014. Real-time Non-rigid Reconstruction using an RGB-D Camera. *ACM Transactions on Graphics (SIGGRAPH)* 33, 4 (2014), 156:1–156:12.