DTexFusion: Dynamic Texture Fusion using a Consumer RGBD Sensor

Chengwei Zheng and Feng Xu

Abstract—In addition to 3D geometry, accurate representation of texture is important when digitizing real objects in virtual worlds. Based on a single consumer RGBD sensor, accurate texture representation for static objects can be realized by fusing multi-frame information; however, extending the process to dynamic objects, which typically have time-varying textures, is difficult. Thus, to address this problem, we propose a compact keyframe-based representation that decouples a dynamic texture into a basic static texture and a set of multiplicative *changing* maps. With this representation, the proposed method first aligns textures recorded from multiple keyframes with the reconstructed dynamic geometry of the object. Errors in the alignment and geometry are then compensated in an innovative iterative linear optimization framework. With the reconstructed texture, we then employ a scheme to synthesize the dynamic object from arbitrary viewpoints. By considering temporal and local pose similarities jointly, dynamic textures in all keyframes are fused to guarantee high-quality image generation. Experimental results demonstrate that the proposed method handles various dynamic objects, including faces, bodies, cloth, and toys. In addition, qualitative and quantitative comparisons demonstrate that the proposed method outperforms state-of-the-art solutions.

Index Terms—texture reconstruction, dynamic texture, novel view synthesis, 3D dynamic reconstruction.

1 Introduction

With the development of depth-sensing techniques, 3D reconstruction has become widespread in professional applications and end users' daily lives. As an increasing number of RGBD sensors are integrated into consumer equipment, such as tablets and smartphones, end users are able to record, edit, animate, and play with real 3D objects for various applications, including 3D measurement, 3D design, video games, virtual reality, and augmented reality. The acquisition of 3D geometry is critical for this purpose, and many techniques have achieved high performance in the reconstruction of either static objects [1], dynamic objects [2], or both [3].

However, without appearance, the reconstruction is not a complete reconstruction of the real world. Several recent studies have noted this and investigated appearance capture with consumer RGBD sensors for daily applications. However, there are several difficulties in capturing appearance. First, the lighting conditions are not controlled or pre-calibrated. Second, the underlying geometry is not accurate, as it is reconstructed from noisy depth input recorded by consumer sensors. Lastly, color images contain distortions, and the camera intrinsic and extrinsic parameters cannot be accurately estimated. To overcome these difficulties, previous works [4], [5] warped recorded multi-view images and synthesized the global texture map of a reconstructed 3D object. The texture map representation did not need to decouple lighting from object appearance, and the warping compensated for errors in depth and color input. However, these techniques use multi-view images to generate one static texture map, and it remains unclear how to handle dynamic objects with time-varying texture.

• *Chengwei Zheng and Feng Xu were with the School of Software, Tsinghua University, Beijing, China.*
E-mail: zhengcw18@gmail.com, xufeng2003@gmail.com

In this paper, we first propose a dynamic texture map representation. Following previous studies [4], [5], we base our technique on a per-vertex texture map, as it is simple and compatible with most animation pipelines. In modeling a dynamic object, we observe that the texture of the object in a particular pose may change in intensity but usually not in color tone; thus, it can be approximated by a basic map with different multiplicative *changing* maps. In addition, we observe that different *changing* maps, which correspond to different possible poses, actually lie in a low-dimensional space, and thus, several key *changing* maps can model almost all the texture variations in a sequence. Consequently, we propose a compact representation containing a basic three-channel texture map and several one-channel multiplicative *changing* maps. The formulation can be expressed as follows:

$$I(x) = M(x) \circ \left(\sum_{i=1}^{K} \beta_i(x) D_i(x) \right), \qquad (1)$$

where I is the synthesized texture map of an object in a particular pose, M is the basic texture map, D_i and β_i are the ith multiplicative *changing* map and its corresponding weight map, respectively, and K is the number of key *changing* maps for the motion sequence. x denotes a vertex of the object. $\circ$ denotes element-wise multiplication and is ignored in the following equations for simplicity. Note that with our representation, texture editing can be easily performed by simply changing the basic map M, which is demonstrated in Section 5.4.

Based on the representation, we further propose a method to reconstruct $M, D_1, ..., D_K$ of a dynamic object with an RGBD sequence, which includes a keyframe selection strategy and a uniform map optimization method. The former selects clear images (denoted as keyframes) with sufficient motion variation from the input sequence, while the latter decouples M and D_is from the keyframes. Given $M, D_1, ..., D_K$, we propose an optimization-based solution to synthesize the object in the sequence from novel viewpoints, where $\beta_i(x)$ for each object point is dynamically determined by jointly considering time, pose, and visibility information. As a consequence, we obtain complete, temporally smooth video results with correct texture dynamics of the moving object from the user-desired viewpoint. The contributions of this technique are as follows:

- A texture representation containing a basic texture map and several multiplicative *changing* maps, which is effective for dynamic texture reconstruction and editing.
- A two-step method consisting of keyframe selection and map optimization to reconstruct dynamic texture from a single-view RGBD input.
- An optimization framework to synthesize videos in novel views with correct dynamics.

2 Related Work

In this paper, we focus on generating the texture map of an object, and we discuss related techniques in this section. We first discuss techniques for handling static objects, followed by techniques for handling dynamic objects. In the literature, there are also many image-based rendering solutions [6], [7], [8]. However, as they do not generate a global texture map attached to a geometric model to fit the current pipelines for video games, VAR, and animation, they are not summarized in this paper.

2.1 Static Texture Reconstruction

There are many methods for registering images to the geometry model. Semi-automatic methods proposed in [9], [10], [11] use manually selected point correspondences to align multiview images. The point correspondences can also be automatically built by analyzing the 3D geometric features of the model and the 2D features of the images [12], [13], [14]. In addition to features, color consistency among multiview images can also be used to align images of the model with the diffuse assumption on the appearance [15], [16], [17], [18]. Moreover, the mutual information between projected 2D images of the texture model can also be used in the registration [19], [20].

In many daily applications for end-users, the geometry model and camera parameters have low accuracy; therefore, providing more freedom in the registration may generate more compelling results. Aganj et al. [21] first matched SIFT feature points among different views and then warped the images with thin-plate splines. Gal et al. [22] projected each triangle in a mesh to a single-input image and then optimized a 2D shift for each triangle. The optical flow between image pairs has also been used to non-rigidly warp images to obtain the final texture [23], [24]. Recently, Zhou and Koltun [4] proposed an iterative linear optimization method to jointly optimize camera parameters and correct distortion, which achieved favorable results. Goldlücke et al. [25] optimized a normal displacement map as well as the camera parameters to obtain super-resolved texture maps. Wu et al. [26] exploited high-dimensional BRDF information to help align the appearance with the low-quality geometry captured by a consumer RGBD sensor. Later, Bi et al. [5] used patch-based matching to provide greater flexibility in the reconstruction of the texture map, which effectively handled challenging cases. Recently, Fu et al. [27] and Li et al. [28] used global and local optimization to obtain a seamless texture of a static object. Fu et al. [29] further jointly optimized the camera poses, texture and geometry of the reconstructed model, and color consistency between the keyframes to recover fine-scale geometry and high-fidelity texture. Kim et al. [30] generated a global texture for a dynamic object; however, the texture was still static.

Other studies also allowed misalignment between the geometry and images; however, they focused on selecting one input image to texture a vertex of the model. A discrete labeling problem had to be solved to avoid visible seams in the results [31], [32], [33], [34]. Other texture composition techniques include sophisticated averaging [35] and Poisson reconstruction [36], [37]. However, none of the above-mentioned techniques can handle dynamic textures, where the misalignment is much stronger, as motion is more difficult to estimate accurately.

2.2 Dynamic Appearance Reconstruction

In addition to static objects, there are also several methods that focus on the appearance reconstruction of dynamic objects. Physical models can be used to reconstruct high-quality appearance [38], [39], [40]. However, a large multiview setup with controlled lighting is usually required. Recently, the development of deep learning techniques has provided new opportunities to capture appearance for animatable targets. Saito et al. [41] represented fine-scale texture details of human faces using mid-layer feature correlations from a deep convolutional neural network. Nagano et al. [42] used a GAN to obtain the texture of a human face for different expressions. Lombardi et al. [43] and Wei et al. [44] modeled face texture using a network encoder, which took multiview images as input and generated view-dependent textures. Wu et al. [45] reconstructed the high-quality color texture of a specific face by deep incremental learning with multiview inputs. Martin-Brualla et al. [46] used a deep architecture to produce high-resolution and high-quality images of the human body from a coarse rendering in real-time. Furthermore, Pandey et al. [47] proposed an end-to-end framework to synthesize free-viewpoint renderings of humans using a single RGBD camera. Although deep learning techniques generate high-quality results, they are restricted to specific objects, such as human faces, and a wide variety of objects have not yet been considered.

Some fusion methods reconstruct the appearance of general objects [48], [49], [50]; however, these methods do not consider the inaccuracy of the geometry and camera parameters, which leads to blurring. Taking the inaccuracy into account, Du et al. [51] proposed a solution toward real-time seamless texture montage build by leveraging geodesics-guided diffusion and temporal texture fields. Prada et al. [52] proposed motion graphs, which enable natural periodic motion, stochastic playback, and user-directed animations. Tsiminaki et al. [53] generalized multiview appearance super-resolution research to the temporal domain with a per-view, per-time frame warp. Other methods

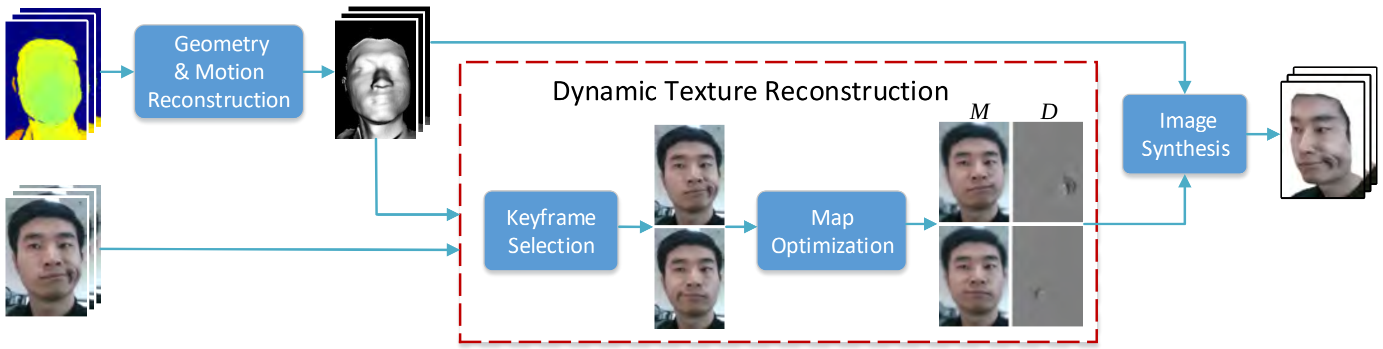

Geometry & Motion Reconstruction
Dynamic Texture Reconstruction
M
D
Keyframe Selection
Map Optimization
Image Synthesis

Fig. 1: Pipeline of proposed framework.

[54], [55] achieved 4D reconstruction with dynamic texture; however, they also relied on multiview inputs. For texture representation, PCA based additive models [56], [57], [58] were also used to generate dynamic texture, especially for human faces [59], [60]. Compared with these models, our multiplicative model that encodes changes in light intensity has clearer physical implications.

3 Method

Taking a monocular RGBD sequence as input, our technique generates the dynamic texture of an object in the sequence as well as its 3D surface geometry and motion. Our method can be divided into three main steps (Fig. 1). First, the input depth sequence is used to fuse the geometry and estimate the motion of the object using an existing technique [2]. Then, we select some keyframes in the color sequence and use them to reconstruct the dynamic texture of the object, which can represent the dynamic shading and shadow effects in the sequence. Finally, we synthesize high-quality images of the moving object from arbitrary viewpoints based on our texture representation and the reconstructed geometry and motion.

3.1 Geometry and Motion Reconstruction

After running [2] for the input sequence, the reconstructed object motion is applied to the fused object mesh after subdivision to reconstruct the mesh sequence, which carries our dynamic texture to synthesize novel high-quality image sequences. The fused geometry is in the pose of the first frame, called the canonical frame. The non-rigid motion from the local deformation of the object is represented by rotations and translations of the nodes, which are normally distributed on the surface. Additional details on how to generate and update the node graph can be found in [2]. It should be noted that the reconstructed 3D geometry and motion are not very accurate; however, our texture reconstruction compensates for the errors to achieve high-quality image synthesis.

3.2 Dynamic Texture Reconstruction

In this subsection, we describe how to reconstruct the dynamic texture of an object. The representation of the dynamic texture is formulated in Section 1; thus, we must estimate $\{M, D_1, ..., D_K\}$ for a dynamic object from an RGBD sequence. We propose a two-step solution, in which we first select a set of keyframes in the RGB sequence and then use a unified optimization to solve all the target maps.

3.2.1 *Keyframe Selection*

In the color sequence, we select keyframes based on both the image clarity and object pose. The former contributes to the quality of the final texture, while the latter helps record dynamic information in the input sequence. Given the latest keyframe, we consider the following Δt frames and calculate a score for each of them. The frame with the highest score is selected as a new keyframe. This process repeatedly runs until the last frame in the sequence is considered. The score is calculated as follows:

$$\sigma(i) = \frac{1}{\|X_i\|} \sum_{x \in X_i} Sobel(x) + c\frac{1}{n} \sum_{j=1}^{n} L_{ip}(N_j)^2, \quad (2)$$

where $\sigma(i)$ is the score of frame i, and X_i is the set of pixels in frame i that correspond to the object. Function $Sobel$ uses the Sobel operator to calculate the gradient magnitude of a pixel as a measurement of clarity. n represents the number of nodes, and $L_{ip}(N_j)$ indicates the Euclidean difference between the positions of node j after non-rigid motion in this frame, i, and the latest keyframe, p. c is a predefined coefficient.

3.2.2 *Map Optimization*

With the selected keyframes $S_i (i = 1, ..., K)$, we run our offline image-based optimization algorithm to obtain $\{M, D_1, ..., D_K\}$. The basic texture map M is designed to describe the texture of the object in the canonical pose (i.e., pose of the canonical frame). To solve for M, we assume that there is a set of intermediate maps M_i, each of which is the projection of M to the keyframe i with the pose and camera parameters of keyframe i. As M_i originates from the static basic texture, by combining the one-channel *changing* map D_i at the keyframe, we obtain a corresponding high-quality texture image $T_i = M_i D_i$ that has the correct texture variations at this pose, as illustrated in Fig. 2. Given these definitions, we can define the uniform energy for solving $\{M, D_1, ..., D_K\}$, which contains three terms: bidirectional similarity term, consistency term, and regularization term.

First, as the reconstructed geometry, motion, and camera parameters may contain errors, T_i should not be S_i. However, as formulated in [5], T_i should have a very low value of

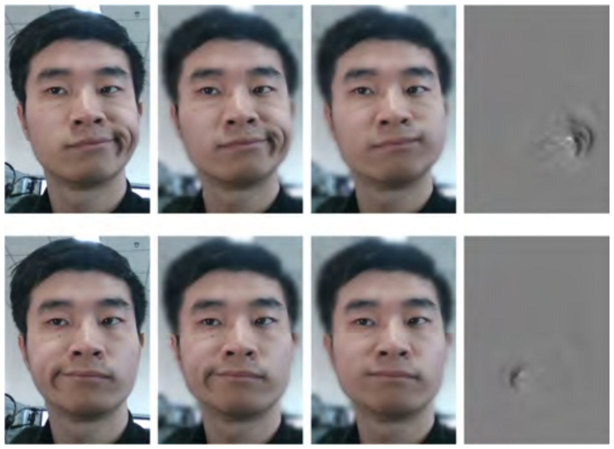


Fig. 2: Images of two keyframes after dynamic texture reconstruction. Column 1: S; column 2: T; column 3: M; column 4: D. (Gray scale [0, 255] in D corresponds to value [0, 2].)

$E_{\text{BDS}}(S_i, T_i)$ with S_i, where the bidirectional similarity term E_{BDS} is defined as follows:

$$E_{\text{BDS}}(S,T) = \frac{1}{L}\left(\sum_{s\subset S}\min_{t\subset T} dist(s,t) + \alpha\sum_{t\subset T}\min_{s\subset S} dist(s,t)\right). \tag{3}$$

Here, α is a predefined constant, L is the number of pixels in a patch, s and t can be any possible patches in images S and T, respectively, and the function $dist$ calculates the sum of squared RGB differences of all the pixels in patches s and t. Minimizing this energy function ensures that for every patch t in the target image T, there is a similar patch s in the source image S, and vice versa.

Thus, the first energy term is constructed by summing E_{BDS} of all keyframes:

$$E_1 = \sum_{i=1}^{K} E_{\text{BDS}}(S_i, T_i). \tag{4}$$

Next, as formulated previously, the basic texture image M combined with the *changing* map D should be close to the dynamic texture image T. As we have an intermediate map M_i for each keyframe i, we can define the consistency for each pair of keyframes, expressed as follows:

$$E_C(T_j, D_j, M_i) = \sum_{x_i} w_j(y_j)\Big(T_j(y_j) - D_j(y_j)M_i(x_i)\Big)^2, \tag{5}$$

where x_i is a pixel position of keyframe i, y_j is a pixel position of keyframe j, and they correspond to the same surface point of the object. As the motions of all frames are solved in the first step, we can obtain these correspondences through projecting a pixel in keyframe i back to the object, unwarping to the canonical pose, warping to the pose of keyframe j, and projecting to the image domain. If the surface point corresponding to x_i is not visible in another keyframe, its term is removed. Similar to [5], the confidence weight w_j is calculated by $w_j = \cos^2\theta$, where θ is the angle between the viewing direction in keyframe j and the surface normal.

Thus, the second energy term is formulated as follows:

$$E_2 = \frac{1}{K}\sum_{j=1}^{K}\sum_{i=1}^{K} E_C(T_j, D_j, M_i). \tag{6}$$

Here, we use $\frac{1}{K}$ to counteract the increased scale caused by the second summation.

The previous two terms cannot solve the scale ambiguity between M and D; therefore, we propose a novel regularization term for this. As we assume that M is the texture map of the object in the canonical pose, we can minimize the following energy for regularization:

$$E_D(D_i) = \sum_{x_i}\lambda_i(x_i)\Big(D_i(x_i) - 1\Big)^2. \tag{7}$$

Here, $\lambda_i(x_i)$ is determined by the local non-rigid motion of the object point corresponding to x_i, which is expressed as $\lambda_i(x_i) = \frac{1}{L_{0i}(x_i)^2}$. $L_{0i}(x_i)$ is the displacement of point x_i from the canonical pose to the pose of keyframe i after the non-rigid motion. Thus, if a point does not move significantly in keyframe i compared to its position in the canonical frame, its corresponding pixel value in D_i should be close to 1, and vice versa. This is consistent with our assumption that D primarily describes the color change of an object point due to its 3D motion.

Thus, the third term consists of this regularization term for all keyframes:

$$E_3 = \sum_{i=1}^{K} E_D(D_i). \tag{8}$$

The energy function used in the reconstruction step can be constructed by summing all three terms above:

$$E = E_1 + \omega_2 E_2 + \omega_3 E_3, \tag{9}$$

where ω_2 and ω_3 are the weights of E_2 and E_3, respectively.

By minimizing (9) with the method in Section 4, all the maps T_i, M_i, D_i are obtained. To obtain the basic texture M for the fused geometry model, for each vertex, we find its corresponding pixel in every M_j (if visible), and compute the weighted average of their colors using the same weight w_j as in (5). We present some M_i from keyframes and the basic texture in Fig. 3.

3.3 Image Synthesis

After the reconstruction of $M, D_1, ..., D_K$, we can use (1) to synthesize the texture map for each pose in the input sequence. The main problem is to determine the combination weight β_i; then, the synthesized *changing* map, denoted $\widetilde{D}$, can be obtained. To solve this problem, we assume that $\widetilde{D}^t$ for the object pose in frame t should be close to some keyframes D_i. Motion and temporal similarities are used to determine this. Here we use superscript t to represent an ordinary frame t in the input sequence, not necessarily a keyframe.

For temporal similarity, only the neighboring keyframes of t are used. The closer they are in time, the more similar

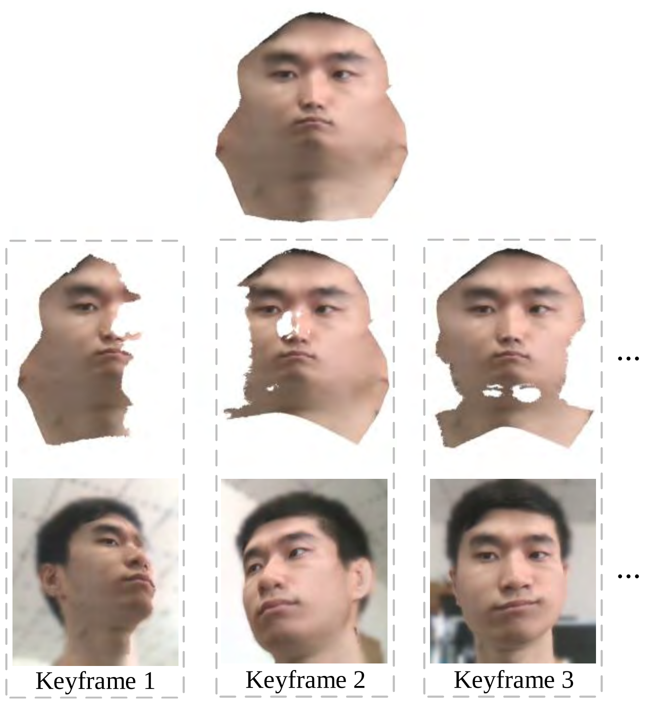


Fig. 3: M_i from each keyframe merged into the basic texture M. Mesh parameterization is only used to display the mesh in UV space, and is not necessary in our pipeline.

$\widetilde{D}^t$ and D_i should be. For a vertex v^t in $\widetilde{D}^t$, we construct the energy function as follows:

$$\widetilde{E}_1(v^t) = s_{tp}\Big(D_p(y_p) - \widetilde{D}^t(v^t)\Big)^2 + s_{tn}\Big(D_n(y_n) - \widetilde{D}^t(v^t)\Big)^2, \tag{10}$$

where p is the index of the previous keyframe of frame t, while n is the index of the next keyframe. y_p and y_n are pixels in keyframe p and n, respectively, both corresponding to vertex v^t. s_{tp} and s_{tn} are weights related to temporal similarity. We set $s_{tp} = \frac{x}{x+y}$ and $s_{tn} = \frac{y}{x+y}$, where x is the number of frames between frame t and the keyframe p, while y is number of frames between frame t and the keyframe n.

For motion similarity, $\widetilde{D}^t(v^t)$ should be close to the D values in the keyframes, which share similar non-rigid motion for the same vertex. Thus, we define

$$\widetilde{E}_2(v^t) = \sum_{j=1}^{K} m_{tj}(v^t)\Big(D_j(y_j) - \widetilde{D}^t(v^t)\Big)^2, \tag{11}$$

where m_{tj} is the motion similarity weight. We take $m_{tj}(v^t) \propto \frac{1}{L_{tj}(v^t)^2}$, with $L_{tj}(v^t)$ indicating the distance between the positions of vertex v^t after the non-rigid motion in frame t and keyframe j. According to this, frames with similar positions after non-rigid motion have greater motion similarity weights at this vertex. In addition, normalization is performed for each vertex v^t to ensure that the sum of the motion similarity weights $m_{tj}(v^t)$ is 1.

Given these two energy terms, each $\widetilde{D}^t$ of a frame t can be calculated by minimizing the energy function:

$$\widetilde{E}^t = \sum_{v^t} \widetilde{E}_1(v^t) + \widetilde{\omega}_2 \widetilde{E}_2(v^t), \tag{12}$$

where $\widetilde{\omega}_2$ determines the combination weight of the two terms.

4 Optimization

In this section, we discuss how to optimize the energy functions in (9) and (12).

4.1 Optimization for (9)

T_i, M_i, and D_i for each keyframe must be optimized in this step. We use an iterative optimization method to alternately optimize these three types of maps in each iteration until convergence.

Optimizing T_i. In each iteration, the optimization of T_i follows the method in [5]. First, we run a patch search algorithm to find every pair of patches s and t. Then, by setting the differential value of (9) to zero, the pixel values in T are updated as follows:

$$T_i(x_i) = \frac{\frac{1}{L}\sum_{u=1}^{U} s_u(y_u) + \frac{\alpha}{L}\sum_{v=1}^{V} s_v(y_v) + \frac{\omega_2}{K} w_i(x_i)\sum_{j=1}^{K} D_i(x_i) M_j(y_j)}{\frac{U}{L} + \frac{\alpha V}{L} + \omega_2 w_i(x_i)}, \tag{13}$$

where s_u is a patch in the source image S_i, whose most similar patch in T_i contains x_i. Thus, s_u is obtained from the forward pairs. As we use bidirectional pairs, we also similarly find s_v from the backward pairs. Therefore, s_u and s_v correspond to the two terms in (3), respectively. y_u and y_v are the corresponding pixels of x_i in s_u and s_v, respectively. Note that there can be more than one patch containing x_i; thus, there can be many paired patches in the source image, S_i. We use U and V to represent the total number of these patches in two directions, respectively.

Optimizing M_i. Then, we update M_i by fixing all T_i and D_i. In this case, only (6) is related to the update of M_i, and we obtain the optimal solution by setting the differential value to zero:

$$M_i(x_i) = \frac{\sum_{j=1}^{K} w_j(y_j) D_j(y_j) T_j(y_j)}{\sum_{j=1}^{K} w_j(y_j) D_j(y_j)^2}. \tag{14}$$

Optimizing D_i. The last step in each iteration is to optimize D_i with fixed T_i and M_i. By setting the differential value to zero again, we obtain

$$D_i(x_i) = \frac{\frac{\omega_2}{K} w_i(x_i) T_i(x_i) \sum_{j=1}^{K} M_j(y_j) + \omega_3 \lambda_i(x_i)}{\frac{\omega_2}{K} w_i(x_i) \sum_{j=1}^{K} M_j(y_j)^2 + \omega_3 \lambda_i(x_i)}. \tag{15}$$

It should be noted that in the optimization above if the surface point corresponding to x_i is not visible in some keyframes, we simply remove the related terms and their weights. In addition, we also use a multi-scale approach similar to [5] for improved results. T_i and M_i are initialized to S_i, and D_i to 1 in all pixels. Furthermore, to prevent overfitting, we do not optimize D_i in the first several iterations of each scale. The optimized maps during this process are displayed in Fig. 4.

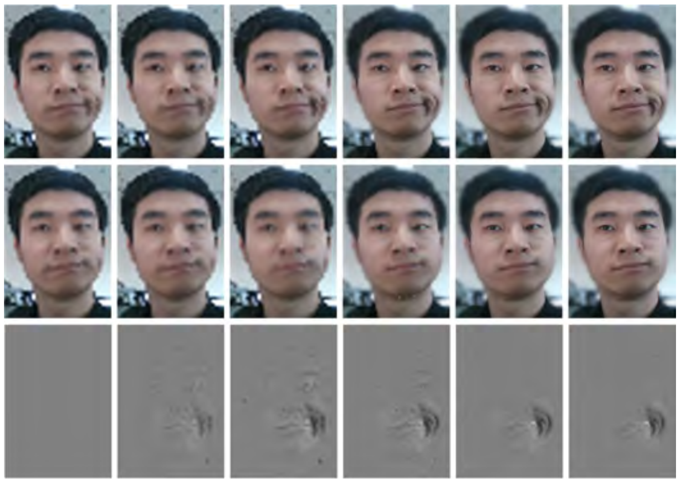


Fig. 4: Optimization variables changed during map optimization (from left to right). Row 1: T; row 2: M; row 3: D.

4.2 Optimization for (12)

As $\widetilde{E}_1(v^t)$ and $\widetilde{E}_2(v^t)$ are both quadratic terms, there is a closed-form solution for minimizing (12):

$$\widetilde{D}^t(v^t) = \frac{s_{tp}D_p(y_p) + s_{tn}D_n(y_n) + \widetilde{\omega}_2 \sum_{j=1}^{K} m_{tj}(v^t)D_j(y_j)}{s_{tp} + s_{tn} + \widetilde{\omega}_2 \sum_{j=1}^{K} m_{tj}(v^t)}. \quad (16)$$

From (16), we observe that the D value of the surface point v^t in frame t is actually a weighted average based on temporal and motion similarities, which is consistent with the formulation in (1). The difference is that (16) allows different weights for different vertices, which provides more flexibility to synthesize the desired texture. For example, although a vertex is not visible in frame t, its D value can also be calculated using (16) from keyframes where this vertex is visible and has a similar local motion to frame t.

When $\widetilde{D}^t$ is obtained, we can calculate the full texture of frame t as $M\widetilde{D}^t$. Then, we can project the texture to an arbitrary viewpoint by the geometry of frame t to synthesize the final output image.

5 Experiments

In this section, we first present the performance and the parameter settings of our technique. Then, we evaluate several key techniques of our system. Thereafter, we present our results on various dynamic objects, along with qualitative and quantitative comparisons with other methods. Sequence results can be found in our accompanying video.

5.1 Performance and Parameters

Our system ran on a computer with a 3.40-GHz four-core CPU, 16 GB RAM, and an NVIDIA GTX GeForce 1080 graphics card. We used Intel RealSense SR300 to record RGBD sequences at 30 fps. Map optimization was implemented on a GPU, and its running time was approximately linear with the number of keyframes, as more than 90% of the running time was used to find patch pairs between

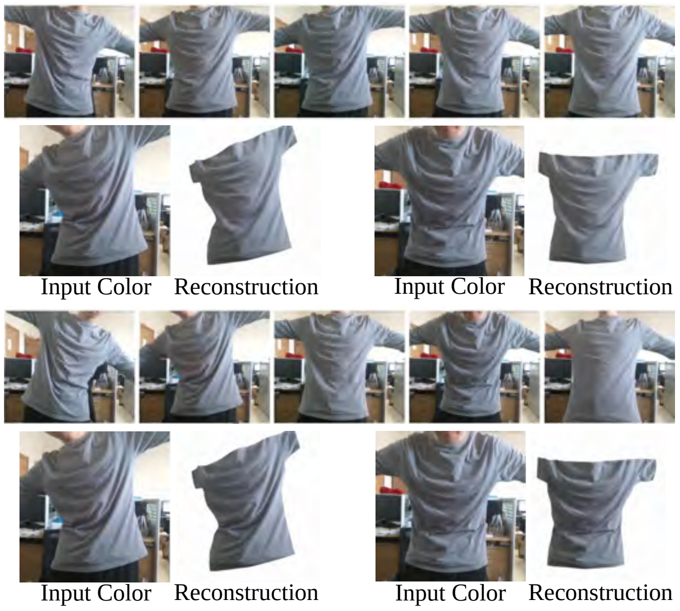


Fig. 5: Evaluation of keyframe selection. Rows 1 and 2: keyframes selected without considering pose information, and reconstruction results with inputs; rows 3 and 4: our selected keyframes and reconstruction results with inputs.

S and T. Each keyframe took approximately 1 min during optimization. For example, we selected seven keyframes out of an input sequence with 200 frames, which took 8 min to run the map optimization. After reconstructing the full sequence, image synthesis took 480 ms for each frame on a CPU for a model with $137,000$ vertices.

Our keyframe-based texture representation usually takes only 1–2% memory space compared to per-frame texture representation, which saves a texture for each frame. For example, to reconstruct a sequence of 600 frames with a mesh containing $144,000$ vertices, our texture representation consisting of 20 keyframe *changing* maps and a basic texture involves 13.25 MB, whereas the per-frame texture representation involves 1.04 GB. In addition, another 19.74 MB is required for geometry and motion. The average compression rate for the texture is 1.31% over 12 sequences with a total of $6,300$ frames.

For the parameters, we set the coefficient c in (2) to 1 when the color values ranged from 0 to 255 and the distances were measured in centimeters. The keyframe selection window Δt varied between 30 and 60 for different sequences. A smaller selection window and dense keyframes do not greatly improve the results, as the keyframes selected under our parameters are able to cover different dynamics. A larger selection window and fewer keyframes may skip important dynamics and lead to poor reconstruction. In map optimization, we set the patch size to 14×14, α in (3) to 2, and ω_2 and ω_3 in (9) to 3 and 500, respectively. A total of 10 scales were used to optimize T, M, and D, from 128×72 to $1,280 \times 720$. In image synthesis, we set the only predefined parameter $\widetilde{\omega}_2$ in (12) to 1.

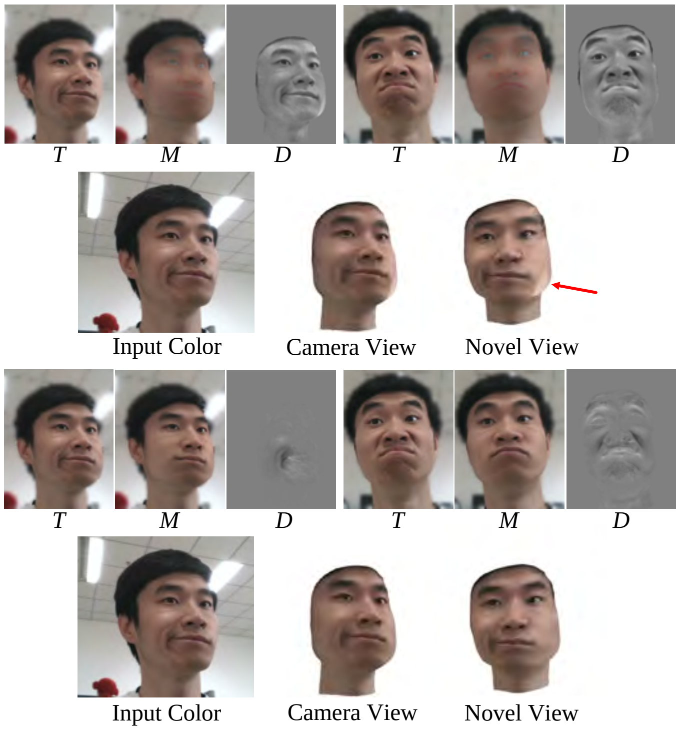


Fig. 6: Evaluation of the regularization term in map optimization. Rows 1 and 2: maps after optimization without our regularization term, and reconstruction results with inputs; rows 3 and 4: maps after optimization with our regularization term, and reconstruction results with inputs.

5.2 Evaluation

Evaluation of keyframe selection. In our keyframe selection method, node motion is taken into account to calculate a score for each frame. We compared our method with the method used in [4], [5], which only used time information and clarity for selecting keyframes. The results are presented in Fig. 5. From the results, we can see that our method was able to select keyframes with different object poses, whereas the compared method skipped these frames but selected frames with similar poses, which is not effective for dynamic texture synthesis.

Evaluation of the regularization term in map optimization. We then evaluated the regularization term in (8), which contributes to decoupling the dynamics of the texture into the D map. We performed optimizations with and without this term, and the results are presented in Fig. 6. We can see that without our regularization term, there was ambiguity between M and D, and the static texture was incorrectly decomposed into the D map. Our regularization term successfully solved the ambiguity. Moreover, misalignment may have been solved as color changing and reconstructed into D without our regularization term, which caused artifacts in the reconstruction results, as displayed in Fig. 6.

Evaluation of texture representation. Our method multiplied the basic texture with one-channel *changing* maps to obtain dynamic texture. For texture representation, other methods [56], [57], [58] used additive texture, which added three-channel color changes to the basic texture. Using our optimization strategy, a three-channel *changing* map of each

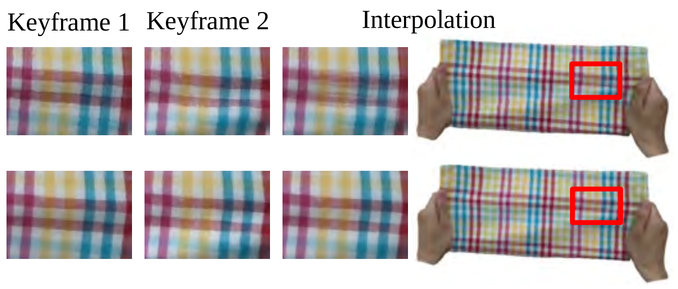


Fig. 7: Evaluation of texture representation. Top: results of additive texture; bottom: our results. Both in the canonical pose.

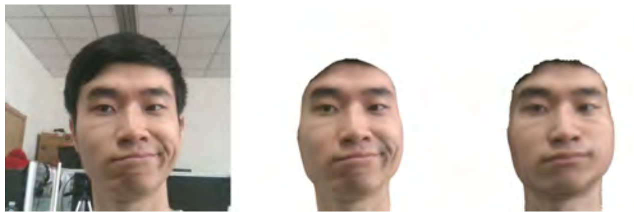


Fig. 8: Comparison with [48]. Left: input color image; middle: our result; right: the result of [48].

keyframe had too much freedom to obtain a favorable result. For each keyframe, the basic texture added with a three-channel map was able to fit the input keyframe image even though they were not aligned. Thus, added maps that were not aligned with the basic texture or each other led to artifacts when interpolating between keyframes, as illustrated in Fig. 7.

Evaluation of image synthesis. Finally, we evaluated our image synthesis method, which considers both temporal and motion similarities. In our accompanying video, we compare our synthesized sequence to a sequence without temporal similarity and a sequence without motion similarity. We can see that without temporal similarity, the result contained some jitter, as the used keyframes had sudden changes. Without motion similarity, the dynamics of the result may not have matched the input correctly, as the texture between two keyframes was always linearly interpolated.

5.3 Results and Comparison

Our technique can synthesize motion sequences from different viewpoints. We tested our methods on various dynamic objects, including faces, bodies, clothes, and toys. We present part of the results in Fig. 14, where each result contains a reference input, our synthesized result from the same viewpoint as the input, and results in two novel viewpoints. In addition, we present the texture of a human face in UV space as well as an illustration of weight maps and *changing* maps in our accompanying video, in which comparisons with the following alternative solutions can also be found.

Comparison with [48]. In [48], the authors reconstructed a dynamic object with geometry, motion, surface albedo, and lighting in real-time. The surface color was calculated by the surface normal, albedo, and lighting. If the geometry detail

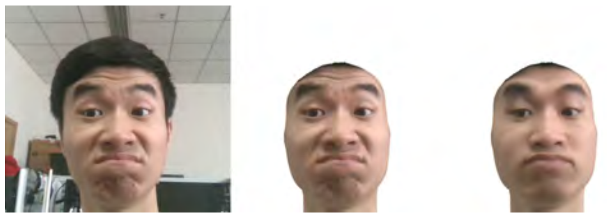


Fig. 9: Comparison with static texture. Left: input color image; middle: our result; right: the result of [5].

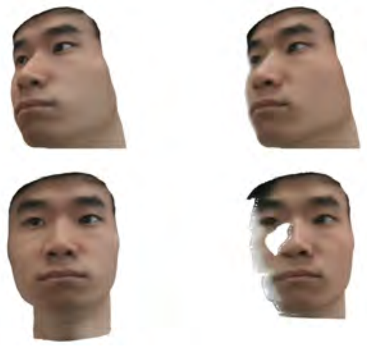


Fig. 10: Comparison with direct projection. Left: our results; right: results of direct projection. The top row presents the results from the camera viewpoint, while the bottom row presents the results from a novel viewpoint.

could not be reconstructed correctly, the incorrect normal led to poor results, as illustrated in Fig. 8. In addition, the inaccuracies in the geometry, non-rigid motion, camera poses, and optical distortions of the input images were not well considered in [48]; thus, the method in [48] fused misaligned data, obtaining blurred results. In contrast, our method obtained a high-quality texture despite these inaccuracies. Furthermore, the geometry in [48] was represented in voxel space, which limits the spatial resolution by the memory size.

Comparison with static texture. In [5], the authors proposed a state-of-the-art method for static texture generation, and we extended this method to handle dynamic objects by DynamicFusion [2]. We also used the keyframes selected by our method for [5] to generate its static texture. The results are presented in Fig. 9. The texture synthesized by [5] did not change for different poses of a dynamic object, whereas our method generated dynamics related to the object poses.

Comparison with direct projection. A naive method to reconstruct the dynamic texture of an object is to recode every color image in the sequence and directly project each vertex to the current image to obtain its color. This method suffers from input noise and cannot obtain the full texture. In addition, it has a very high memory cost. The comparison is presented in Fig. 10, where the direct projection method fails to synthesize the novel viewpoint.

We also performed quantitative comparisons with previous studies using the method in [48] and static texture [5]. In this comparison, we generated results in the input viewpoints and could thus calculate numerical errors by assuming the recorded sequence to be the ground truth. Specifically, we used the average RGB distance between the corresponding pixels in the recorded image and the reconstructed image as the reconstruction error. The results indicate that our method consistently achieved the lowest errors in the sequence, as illustrated in Fig. 11. We did not compare our method with direct projection here, as its reconstruction error is always zero in this measurement. However, as discussed previously and illustrated in the accompanying video, it cannot reconstruct novel views as well as other methods. There are three peaks with particularly large errors in [5] and [48], which correspond to three poses in the sequence, as displayed in Fig. 11. The methods of [5] and [48] could not reconstruct large wrinkle changes caused by these poses. The errors of [5] and [48] decreased in the rest poses but were still larger than ours. The method of [48] updated surface albedo by fusing data from previous frames, and the frames close to the current frame in time had greater weights in the fusion, while the static texture [5] was not updated; thus, the error curve changes of [48] and [5] were not identical. It should be noted that for our method, the frames with very low errors were the selected keyframes. Additional results, as well as comparisons, can be found in our accompanying video.

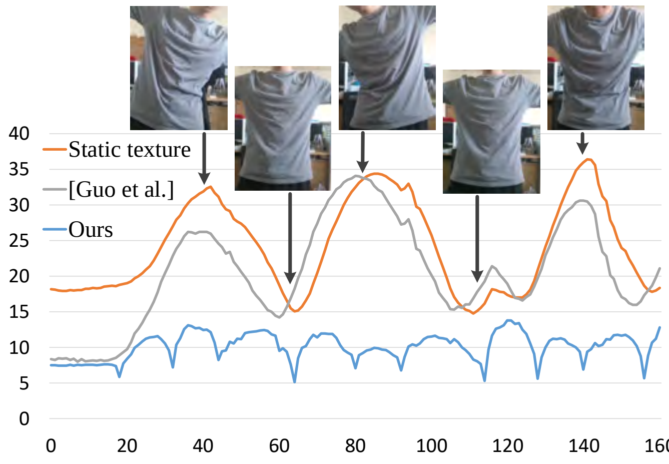


Fig. 11: Numeric comparisons with [48] and static texture [5] (160 frames presented). The error in each frame is calculated by the average RGB difference. The average error of the entire sequence (540 frames) is 22.327 for static texture [5], 20.217 for [48], and 9.495 for our method. Selected input color frames are presented at the top.

We also tested our method on the RGBD dataset of [61], and some sequence results are provided in our accompanying video. This work [61] focused on motion tracking and did not reconstruct the appearance color of the object; therefore, we did not compare our method with [61]. Our method was not effective in some sequences of [61] due to motion tracking failures, which we discuss in Section 6.

5.4 Application

As we represent our dynamic texture using a basic texture and a set of multiplicative *changing* maps, it is convenient to perform texture editing. In our method, only the basic texture should be edited, while the *changing* maps can remain the same. Fig. 12 presents some frames of the dynamic

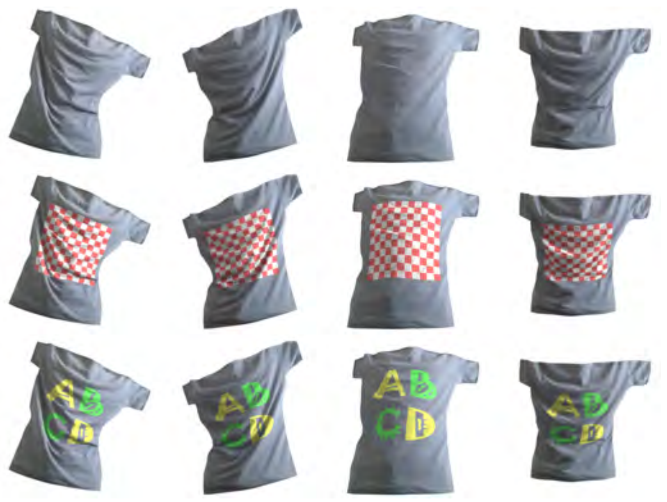
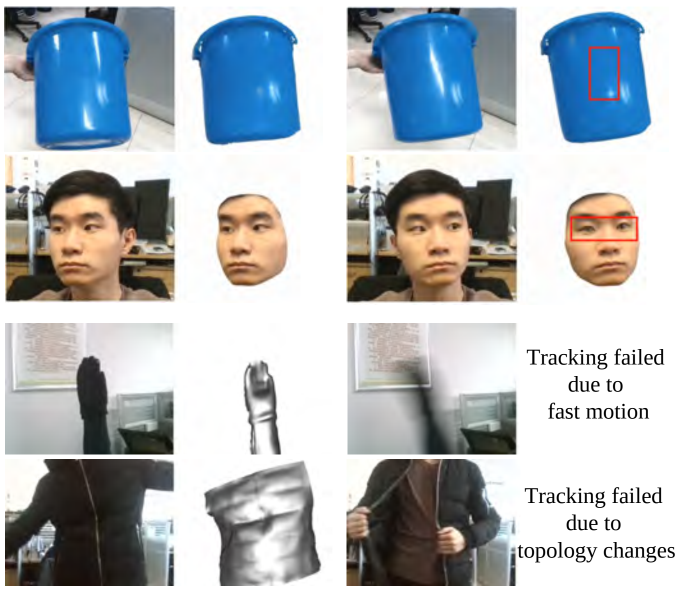


Fig. 12: Texture editing (four frames). Row 1: reconstruction results without editing; rows 2 and 3: results after texture editing.


Tracking failed due to fast motion
Tracking failed due to topology changes
Fig. 13: Failure cases. Rows 1 and 2: texture reconstruction failure due to specular reflectance and eyeball movement, respectively; rows 3 and 4: motion tracking failure due to fast motion and topology changes, respectively.

texture after editing using our method, while the full results can be found in our accompanying video.

6 Discussion

Our method cannot handle texture changes caused by global head motion, as our non-rigid motion reconstruction method [2] cannot distinguish global head motion from camera motion. Therefore, we assumed that an object had no global motion or that the global motion had little effect on the texture in our experiments; otherwise, the parts in our pipeline that relied on the motion of the object would not be effective. Motion tracking may fail when fast motion and topology changes occur, which further led to failure in our reconstruction. Some movements that cannot be reconstructed correctly, such as eyeball movements, can also cause artifacts in our reconstruction. Eye motion tracking [62] can be used to handle this situation.

Additionally, our method fuses the dynamic texture information of multiple frames together to seek a favorable result, and we cannot correctly generate dynamics that have not been recorded. Our representation of dynamic texture is effective for Lambertian surfaces; however, highlights and specular reflectance cannot be handled properly. Deep learning may be used to improve performance. We present some inputs and results of failure cases in our accompanying video as well as in Fig. 13, and for motion tracking failures, we present their reconstructed geometries.

7 Conclusions

In this paper, we propose a novel keyframe-based texture representation that can represent texture changes caused by objects' non-rigid motion in uncalibrated daily lighting conditions and is lightweight to easily reconstruct dynamic texture using a consumer RGBD sensor. Based on the representation, we also propose a dynamic texture reconstruction method and a novel view synthesis method to allow the representation to be easily used in real applications. Our texture reconstruction is robust to distortions and noise in color and depth inputs, as well as errors in the estimated object motion. In addition, our image synthesis method fully explores the information in a recorded sequence to achieve smooth and high-quality synthesis. Texture editing can also be easily implemented using our texture representation.

References

[1] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *2011 IEEE International Symposium on Mixed and Augmented Reality*. IEEE, 2011, pp. 127–136.
[2] R. A. Newcombe, D. Fox, and S. M. Seitz, "Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 343–352.
[3] H. Zhang and F. Xu, "Mixedfusion: Real-time reconstruction of an indoor scene with dynamic objects," *IEEE transactions on visualization and computer graphics*, vol. 24, no. 12, pp. 3137–3146, 2018.
[4] Q.-Y. Zhou and V. Koltun, "Color map optimization for 3d reconstruction with consumer depth cameras," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, p. 155, 2014.
[5] S. Bi, N. K. Kalantari, and R. Ramamoorthi, "Patch-based optimization for image-based texture mapping," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–11, 2017.
[6] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen, "Unstructured lumigraph rendering," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 2001, pp. 425–432.
[7] P. Hedman, T. Ritschel, G. Drettakis, and G. Brostow, "Scalable inside-out image-based rendering," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, p. 231, 2016.
[8] P. E. Debevec, C. J. Taylor, and J. Malik, *Modeling and rendering architecture from photographs*. University of California, Berkeley, 1996.
[9] T. Franken, M. Dellepiane, F. Ganovelli, P. Cignoni, C. Montani, and R. Scopigno, "Minimizing user intervention in registering 2d images to 3d models," *The Visual Computer*, vol. 21, no. 8-10, pp. 619–628, 2005.
[10] E. Ofek, E. Shilat, A. Rappoport, and M. Werman, "Multiresolution textures from image sequences," *IEEE Computer Graphics and Applications*, no. 2, pp. 18–29, 1997.

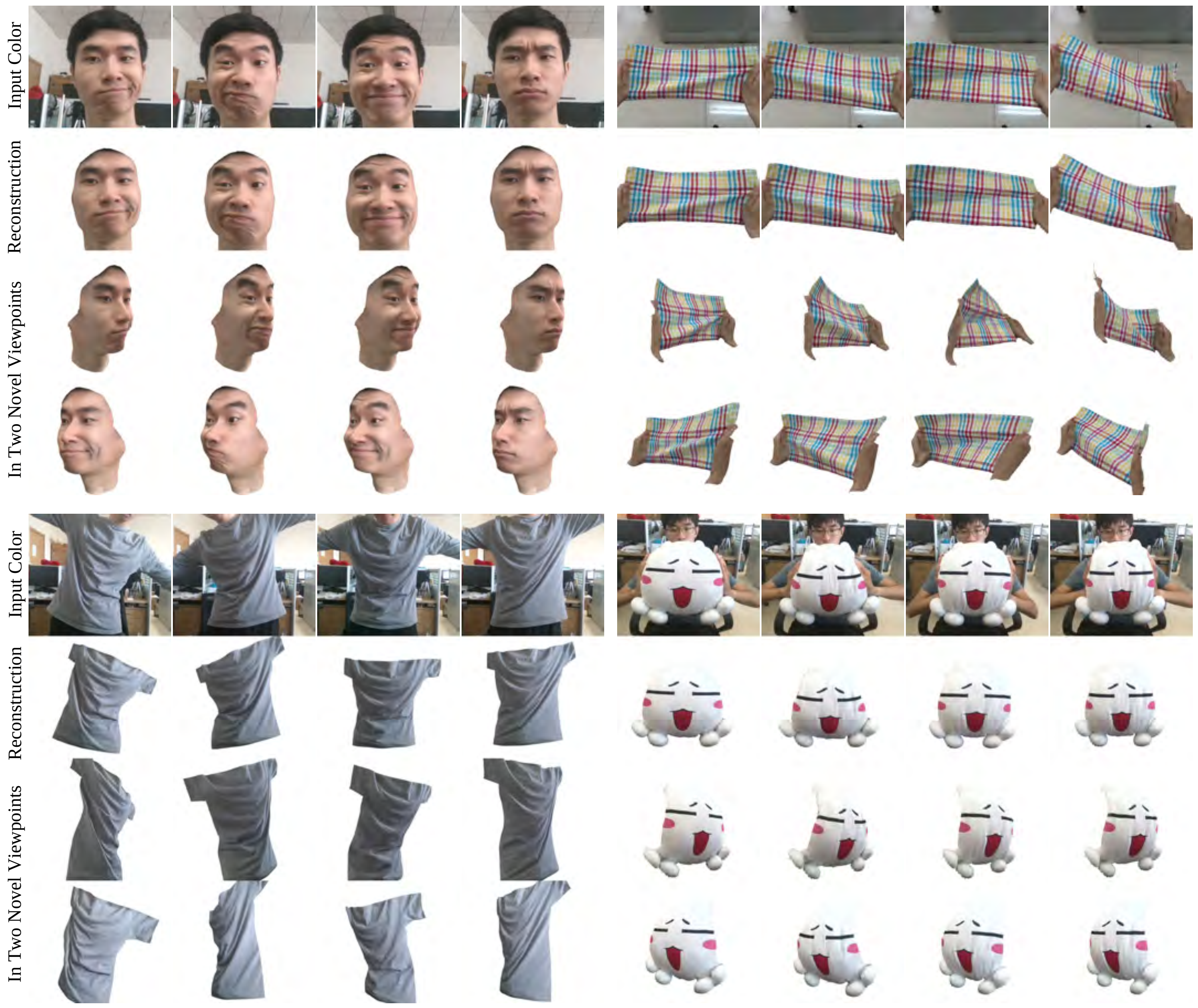


Fig. 14: Results of sequences of different dynamic objects.

[11] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. H. Salesin, "Synthesizing realistic facial expressions from photographs," in *Computer graphics proceedings, annual conference series*. Association for Computing Machinery SIGGRAPH, 1998, pp. 75–84.

[12] H. P. Lensch, W. Heidrich, and H.-P. Seidel, "A silhouette-based algorithm for texture registration and stitching," *Graphical Models*, vol. 63, no. 4, pp. 245–262, 2001.

[13] I. Stamos and P. K. Allen, "Geometry and texture recovery of scenes of large scale," *Computer vision and image understanding*, vol. 88, no. 2, pp. 94–118, 2002.

[14] L. Liu and I. Stamos, "A systematic approach for 2d-image to 3d-range registration in urban environments," *Computer Vision and Image Understanding*, vol. 116, no. 1, pp. 25–37, 2012.

[15] A. E. Johnson and S. B. Kang, "Registration and integration of textured 3d data," *Image and vision computing*, vol. 17, no. 2, pp. 135–147, 1999.

[16] F. Bernardini, I. M. Martin, and H. Rushmeier, "High-quality texture reconstruction from multiple scans," *IEEE Transactions on Visualization and Computer Graphics*, vol. 7, no. 4, pp. 318–332, 2001.

[17] K. Pulli and L. G. Shapiro, "Surface reconstruction and display from range and color data," *Graphical Models*, vol. 62, no. 3, pp. 165–201, 2000.

[18] K. Pulli, S. Piiroinen, T. Duchamp, and W. Stuetzle, "Projective surface matching of colored 3d scans," in *Fifth International Conference on 3-D Digital Imaging and Modeling (3DIM'05)*. IEEE, 2005, pp. 531–538.

[19] M. Corsini, M. Dellepiane, F. Ponchio, and R. Scopigno, "Image-to-geometry registration: a mutual information method exploiting illumination-related geometric properties," in *Computer Graphics Forum*, vol. 28, no. 7. Wiley Online Library, 2009, pp. 1755–1764.

[20] M. Corsini, M. Dellepiane, F. Ganovelli, R. Gherardi, A. Fusiello, and R. Scopigno, "Fully automatic registration of image sets on approximate geometry," *International journal of computer vision*, vol. 102, no. 1-3, pp. 91–111, 2013.

[21] E. Aganj, P. Monasse, and R. Keriven, "Multi-view texturing of imprecise mesh," in *Asian Conference on Computer Vision*. Springer, 2009, pp. 468–476.

[22] R. Gal, Y. Wexler, E. Ofek, H. Hoppe, and D. Cohen-Or, "Seamless montage for texturing models," in *Computer Graphics Forum*, vol. 29, no. 2. Wiley Online Library, 2010, pp. 479–486.

[23] M. Dellepiane, R. Marroquim, M. Callieri, P. Cignoni, and R. Scopigno, "Flow-based local optimization for image-to-geometry projection," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 3, pp. 463–474, 2012.

[24] M. Eisemann, B. De Decker, M. Magnor, P. Bekaert, E. De Aguiar, N. Ahmed, C. Theobalt, and A. Sellent, "Floating textures," in *Computer graphics forum*, vol. 27, no. 2. Wiley Online Library, 2008, pp. 409–418.

[25] B. Goldlücke, M. Aubry, K. Kolev, and D. Cremers, "A super-

resolution framework for high-accuracy multiview reconstruction," *International journal of computer vision*, vol. 106, no. 2, pp. 172–191, 2014.

[26] H. Wu, Z. Wang, and K. Zhou, "Simultaneous localization and appearance estimation with a consumer rgb-d camera," *IEEE transactions on visualization and computer graphics*, vol. 22, no. 8, pp. 2012–2023, 2015.

[27] Y. Fu, Q. Yan, L. Yang, J. Liao, and C. Xiao, "Texture mapping for 3d reconstruction with rgb-d sensor," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4645–4653.

[28] W. Li, H. Gong, and R. Yang, "Fast texture mapping adjustment via local/global optimization," *IEEE transactions on visualization and computer graphics*, vol. 25, no. 6, pp. 2296–2303, 2018.

[29] Y. Fu, Q. Yan, J. Liao, and C. Xiao, "Joint texture and geometry optimization for rgb-d reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5950–5959.

[30] J. Kim, H. Kim, J. Park, and S. Lee, "Global texture mapping for dynamic objects," *Computer Graphics Forum*, vol. 38, no. 7, pp. 697–705, 2019.

[31] V. Lempitsky and D. Ivanov, "Seamless mosaicing of image-based texture maps," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–6.

[32] S. N. Sinha, D. Steedly, R. Szeliski, M. Agrawala, and M. Pollefeys, "Interactive 3d architectural modeling from unordered photo collections," in *ACM Transactions on Graphics (TOG)*, vol. 27, no. 5. ACM, 2008, p. 159.

[33] L. Velho and J. Sossai Jr, "Projective texture atlas construction for 3d photography," *The Visual Computer*, vol. 23, no. 9-11, pp. 621–629, 2007.

[34] M. Waechter, N. Moehrle, and M. Goesele, "Let there be color! large-scale texturing of 3d reconstructions," in *European Conference on Computer Vision*. Springer, 2014, pp. 836–850.

[35] M. Callieri, P. Cignoni, M. Corsini, and R. Scopigno, "Masked photo blending: Mapping dense photographic data set on high-resolution sampled 3d models," *Computers & Graphics*, vol. 32, no. 4, pp. 464–473, 2008.

[36] M. Chuang, L. Luo, B. J. Brown, S. Rusinkiewicz, and M. Kazhdan, "Estimating the laplace-beltrami operator by restricting 3d functions," in *Computer graphics forum*, vol. 28, no. 5. Wiley Online Library, 2009, pp. 1475–1484.

[37] H. Li, E. Vouga, A. Gudym, L. Luo, J. T. Barron, and G. Gusev, "3d self-portraits," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, p. 187, 2013.

[38] P. Debevec, T. Hawkins, C. Tchou, H.-P. Duiker, W. Sarokin, and M. Sagar, "Acquiring the reflectance field of a human face," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 145–156.

[39] O. Alexander, M. Rogers, W. Lambeth, J.-Y. Chiang, W.-C. Ma, C.-C. Wang, and P. Debevec, "The digital emily project: Achieving a photorealistic digital actor," *IEEE Computer Graphics and Applications*, vol. 30, no. 4, pp. 20–31, 2010.

[40] P. Gotardo, J. Riviere, D. Bradley, A. Ghosh, and T. Beeler, "Practical dynamic facial appearance modeling and acquisition," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, pp. 1–13, 2018.

[41] S. Saito, L. Wei, L. Hu, K. Nagano, and H. Li, "Photorealistic facial texture inference using deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5144–5153.

[42] K. Nagano, J. Seo, J. Xing, L. Wei, Z. Li, S. Saito, A. Agarwal, J. Fursund, H. Li, R. Roberts *et al.*, "pagan: real-time avatars using dynamic textures," in *SIGGRAPH Asia 2018 Technical Papers*. ACM, 2018, p. 258.

[43] S. Lombardi, J. Saragih, T. Simon, and Y. Sheikh, "Deep appearance models for face rendering," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 68:1–68:13, Jul. 2018. [Online]. Available: http://doi.acm.org/10.1145/3197517.3201401

[44] S.-E. Wei, J. Saragih, T. Simon, A. W. Harley, S. Lombardi, M. Perdoch, A. Hypes, D. Wang, H. Badino, and Y. Sheikh, "Vr facial animation via multiview image translation," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, p. 67, 2019.

[45] C. Wu, T. Shiratori, and Y. Sheikh, "Deep incremental learning for efficient high-fidelity face tracking," in *SIGGRAPH Asia 2018 Technical Papers*. ACM, 2018, p. 234.

[46] R. Martin-Brualla, R. Pandey, S. Yang, P. Pidlypenskyi, J. Taylor, J. Valentin, S. Khamis, P. Davidson, A. Tkach, P. Lincoln *et al.*, "Lookingood: enhancing performance capture with real-time neural re-rendering," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, pp. 1–14, 2018.

[47] R. Pandey, A. Tkach, S. Yang, P. Pidlypenskyi, J. Taylor, R. Martin-Brualla, A. Tagliasacchi, G. Papandreou, P. Davidson, C. Keskin *et al.*, "Volumetric capture of humans with a single rgbd camera via semi-parametric learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9709–9718.

[48] K. Guo, F. Xu, T. Yu, X. Liu, Q. Dai, and Y. Liu, "Real-time geometry, albedo, and motion reconstruction using a single rgb-d camera," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 3, p. 32, 2017.

[49] M. Dou, P. Davidson, S. R. Fanello, S. Khamis, A. Kowdle, C. Rhemann, V. Tankovich, and S. Izadi, "Motion2fusion: Real-time volumetric performance capture," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 6, p. 246, 2017.

[50] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, and S. Sullivan, "High-quality streamable free-viewpoint video," *ACM Transactions on Graphics (ToG)*, vol. 34, no. 4, p. 69, 2015.

[51] R. Du, M. Chuang, W. Chang, H. Hoppe, and A. Varshney, "Montage4d: interactive seamless fusion of multiview video textures," in *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 2018, pp. 1–11.

[52] F. Prada, M. Kazhdan, M. Chuang, A. Collet, and H. Hoppe, "Motion graphs for unstructured textured meshes," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 1–14, 2016.

[53] V. Tsiminaki, J.-S. Franco, and E. Boyer, "High resolution 3d shape texture from multiple videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1502–1509.

[54] T. Tung, S. Nobuhara, and T. Matsuyama, "Simultaneous super-resolution and 3d video using graph-cuts," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2008, pp. 1–8.

[55] D. Casas, M. Volino, J. Collomosse, and A. Hilton, "4d video textures for interactive character appearance," in *Computer Graphics Forum*, vol. 33, no. 2. Wiley Online Library, 2014, pp. 371–380.

[56] F. De la Torre and M. J. Black, "Robust principal component analysis for computer vision," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 1. IEEE, 2001, pp. 362–369.

[57] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *Journal of the ACM (JACM)*, vol. 58, no. 3, p. 11, 2011.

[58] A. Boukhayma, V. Tsiminaki, J.-S. Franco, and E. Boyer, "Eigen appearance maps of dynamic shapes," in *European Conference on Computer Vision*. Springer, 2016, pp. 230–245.

[59] V. Blanz, T. Vetter *et al.*, "A morphable model for the synthesis of 3d faces." in *Siggraph*, vol. 99, no. 1999, 1999, pp. 187–194.

[60] O. Alexander, M. Rogers, W. Lambeth, M. Chiang, and P. Debevec, "The digital emily project: photoreal facial modeling and animation," in *Acm siggraph 2009 courses*. ACM, 2009, p. 12.

[61] A. Božič, M. Zollhöfer, C. Theobalt, and M. Nießner, "Deepdeform: Learning non-rigid rgb-d reconstruction with semi-supervised data," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020, pp. 7000–7010.

[62] Q. Wen, F. Xu, and J.-H. Yong, "Real-time 3d eye performance reconstruction for rgbd cameras," *IEEE transactions on visualization and computer graphics*, vol. 23, no. 12, pp. 2586–2598, 2016.

Chengwei Zheng received a B.S. degree in software engineering from Tsinghua University, Beijing, China, in 2018. He is currently working toward a Ph.D. degree in the School of Software, Tsinghua University. His research interests include dynamic reconstruction and 3D animation.

Feng Xu received a B.S. degree in physics from Tsinghua University, Beijing, China, in 2007 and Ph.D. in automation from Tsinghua University, Beijing, China, in 2012. He is currently an associate professor in the School of Software, Tsinghua University. His research interests include face animation, performance capture, and 3D reconstruction.