Pacific Graphics 2019
C. Theobalt, J. Lee, and G. Wetzstein
(Guest Editors)

Volume 38 (2019), Number 7

Global Texture Mapping for Dynamic Objects

Jungeon Kim, Hyomin Kim, Jaesik Park, and Seungyong Lee

POSTECH

Figure 1: *Dynamic objects with global texture maps optimized by our framework.*

Abstract

*We propose a novel framework to generate a global texture atlas for a deforming geometry. Our approach distinguishes from prior arts in two aspects. First, instead of generating a texture map for each timestamp to color a dynamic scene, our framework reconstructs a global texture atlas that can be consistently mapped to a deforming object. Second, our approach is based on a single RGB-D camera, without the need of a multiple-camera setup surrounding a scene. In our framework, the input is a 3D template model with an RGB-D image sequence, and geometric warping fields are found using a state-of-the-art non-rigid registration method [GXW*15] to align the template mesh to noisy and incomplete input depth images. With these warping fields, our multi-scale approach for texture coordinate optimization generates a sharp and clear texture atlas that is consistent with multiple color observations over time. Our approach is accelerated by graphical hardware, providing a handy configuration to capture a dynamic geometry along with a clean texture atlas. We demonstrate our approach with practical scenarios, particularly human performance capture. We also show that our approach is resilient on misalignment issues caused by imperfect estimation of warping fields and inaccurate camera parameters.*

CCS Concepts

• ***Computing methodologies*** → ***Shape modeling;***

1. Introduction

3D reconstruction using RGB-D cameras becomes a popular way to capture geometry from the real-world. KinectFusion [NIH*11] is one of the representative works that demonstrate the real-time reconstruction of a static scene using a sequence of depth images. This pioneering work has been extended by various groups [NZIS13, WSMG*16, CZK15, PZK17, DNZ*17] to cover large-scale scenes such as rooms and statues. Recently, methods on non-static or dynamic object reconstruction have been introduced, as the scene of interest may not be static in actual cases - the most representative example is human who can perform various motions.

The geometric deformation of human over time is becoming more important since it is directly related to content creation for emerging applications, such as virtual/augmented reality, 360° replay of sports games, commercial videos, and movies.

Dynamic scene reconstruction is often handled by reconstructing a shape for each timestamp with multiple cameras placed around the scene. This approach utilizes multi-view information as much as possible and shows outstanding results [DDF*17, DKD*16, CCS*15]. However, these multi-camera environments require extrinsic calibration and a specialized frame trigger for multi-frame synchronization. These often result in constrained and complicated studio environments.

To resolve this issue, dynamic scene reconstruction using a minimal setup of a single RGB-D camera has been studied recently. Starting from DynamicFusion [NFS15], VolumeDeform [IZN*16], BodyFusion [YGX*17], and the recent SMPL (skinned multi-person linear model) based DoubleFusion [YZG*18] perform real-time dynamic motion tracking and exhibit impressive reconstruction results. For acquiring better results from more challenging targets, template-based methods have also been proposed [LAGP09, GXW*15, GXW*18]. In this case, a template mesh is built in a preprocessing step, and the template mesh is non-rigidly aligned to input depth frames.

While single-camera based approaches have successfully demonstrated 3D geometry reconstruction of non-rigid objects, color or texture map generation for the reconstructed models is a separate and still demanding problem. Several methods have been proposed to restore color or texture information of a 3D rigid model reconstructed using an RGB or RGB-D image stream [GWO*10, ZK14, JJKL16, BKR17, FYY*18, LGY19]. However, those methods cannot be directly extended to handle non-rigid objects, where small misalignments of textures over time can produce visual artifacts that are easily noticeable by users. Recently Prada et al. [PKC*17] introduce a complete system that can capture high-quality geometry and texture map together for dynamic scenes. However, it needs to generate multiple texture maps that progressively modify a global texture map, and requires a controlled studio setup for capturing multi-view images at each frame.

In this paper, we introduce a practical system to generate a texture atlas for color mapping of non-rigid objects using a single RGB-D camera. Our single camera setup increases the convenience of usual consumer and avoids using any special-purpose environments such as turntable or studio setup. The core of our framework is an efficient method to produce a *single, clean, and accurate* texture atlas that can be *globally* applicable to a deformed template mesh over time. Our method can be configured with any existing template-based non-rigid tracking method [GXW*15, GXW*18] that recovers the geometry of a deforming 3D object over time.

To generate a global texture domain, we begin with initial 2D texture coordinates on several color images for each face of a template mesh. Our problem of producing a clean and accurate texture atlas can be formulated as local texture coordinate optimization using warping fields that are estimated by a state-of-the-art non-rigid registration method [GXW*15]. Texture coordinate optimization is accelerated by graphical hardware based on the approach of Jeon et al. [JJKL16]. To handle large misalignments due to various sources, such as inaccurately estimated non-rigid motions and camera parameter errors, we perform multi-scale optimization using a mesh pyramid that is generated by deformation-oriented mesh decimation (DOD) [HCC06].

In summary, our contributions are:

- A practical system for dynamic scene and color map acquisition using only a single RGB-D camera.
- An effective framework for global texture mapping of a deforming 3D mesh.
- Efficient texture coordinate optimization scheme using a mesh pyramid to handle large misalignments.
- Demonstration of the proposed approach using various dynamic objects, such as human performing a range of motions.

2. Related Work

We review relevant literature to our approach including dynamic mesh reconstruction and texture map handling.

Non-rigid geometry reconstruction KinectFusion [NIH*11] is a representative static scene reconstruction approach using a single depth camera. As a follow-up, DynamicFusion [NFS15] demonstrates real-time non-rigid reconstruction using a single depth camera. DynamicFusion utilizes point-to-plane error for the geometric alignment, so the alignment can be slipped away if the scene is planar. VolumeDeform [IZN*16] resolves this issue by adding a correspondence term using image feature in addition to the geometric error term in their energy equation. BodyFusion [YGX*17] is specialized for dynamic human body capture. It utilizes human skeleton embedding obtained from a single RGB-D image when it fuses depth images. Recently, DoubleFusion [YZG*18] introduces a two-layered representation to handle the case that a target subject is wearing clothes. A layer for inner body shape is explicitly parameterized by SMPL model [LMR*15], and the outer body shape is progressively reconstructed by fusing depth images.

Another group of methods assume that a template shape has been recovered, and deforms the template to be aligned with a depth image. This approach allows full dynamic motion capture using a single depth camera, where the core problem is non-rigid registration between a template model with input data [LAGP09, ZNI*14, GXW*15, GXW*18]. Many works use a non-rigid registration technique based on embedded deformation (ED) graph model [SSP07], which has shown successful registration performance in diverse literature [LVG*13, DTF*15, DKD*16, DDF*17] besides template model tracking. Guo et al. [GXW*15, GXW*18] use ED graph model, and reconstruct complex articulated motions using L_0 based motion regularizer without the usage of a predefined skeleton embedding.

Color mapping of reconstructed geometry In practice, when a depth camera is used for geometry scanning and its accompanying color camera is used for texture mapping, the misalignment issue appears whose main sources are camera model errors and inaccurate reconstructed geometry. Such misalignments often result in blurry texture maps that are not visually appealing for virtual/augmented reality applications. To resolve this issue, Gal et

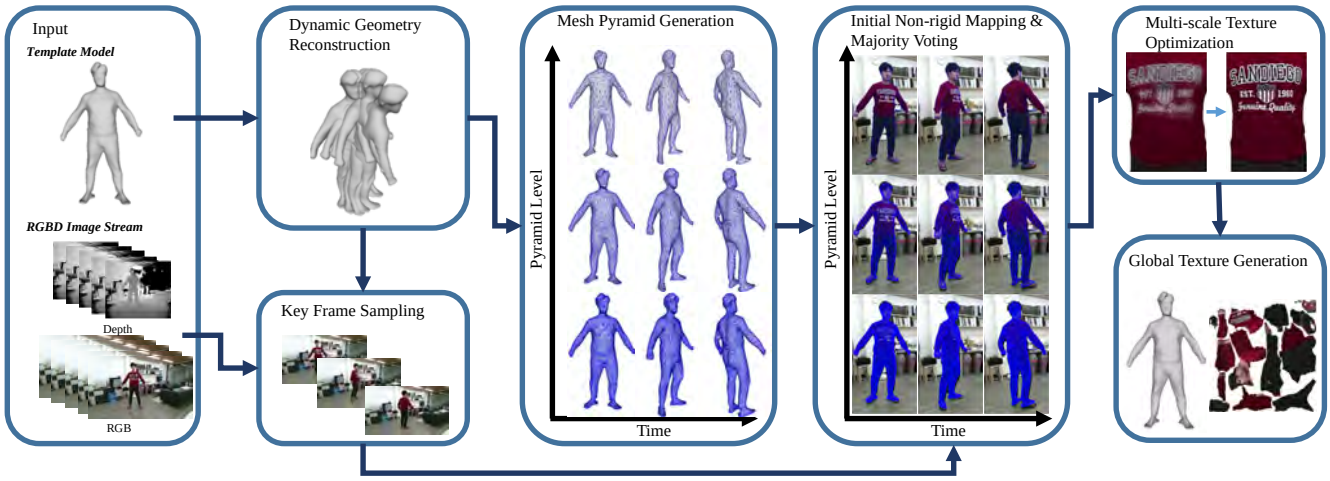


Figure 2: *Overview of the proposed framework to generate a global texture map for a deforming geometry. Given a template model and an RGB-D stream, our framework generates a deforming mesh using non-rigid registration between the depth images and the template model. Color images having different views of the target object are selected as key frames for texture mapping. Our multi-scale texture optimization generates a complete, clean, and sharp texture map that can be globally applied to the deforming mesh.*

al. [GWO*10] optimize projective mapping from each mesh triangle to one of color images by performing combinatorial optimization on image labels and 2D coordinates of projected triangles. Zhou and Koltun [ZK14] jointly optimize camera poses and non-rigid correction functions to increase the photometric consistency of the assigned color images. Their approach alters per-vertex colors during optimization, and thus requires a high-density mesh for generating a high-quality colorization of mesh. Jeon et al. [JJKL16] use a texture map for color information, instead of per-vertex colors, and optimize projected texture coordinates on input color images to maximize the photometric consistency. They demonstrated that their method is efficient and adopted well for both small- and large-scale models. Lin et al. [LCL*16] propose a practical multi-view camera system that fast reconstructs both geometry and texture of a 3D rigid object. When texturing an object, they estimate the optimal image label of each mesh triangle, similarly to Gal et al. [GWO*10]. Global patch-based optimization introduced by Bi et al. [BKR17] is robust to large misalignments between frames, but is known to be time-consuming due to heavy optimization step. Other works [FYY*18, LGY19] also deal with color enhancement of reconstructed rigid mesh models by means of assigning an optimal image label to each face of a model with variants of photometric consistency terms.

Recently, there are increasing demands to capture and display dynamic objects with high-quality color mapping for various applications. The method of Prada et al. [PKC*17] is one of the representative works in this area, but it requires evolving texture maps that are captured by a highly crafted multi-view camera system.

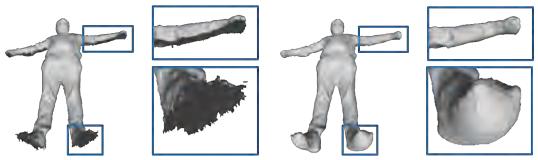
In contrast, we propose a more compact capturing pipeline that only requires a single RGB-D camera to capture geometry and its dynamic performance over time. In particular, our core contribution is to generate a temporally consistent texture atlas over time, which has not been introduced so far based on our best knowledge. Our texture optimization approach is designed to align high-resolution color images effectively. We adopt the approach of Jeon et al. [JJKL16] to formulate the texture map alignment problem as a texture coordinate optimization problem. On top of this, we propose multi-scale optimization of texture coordinates using a mesh pyramid. With our approach, the color image sequence is associated with the faces of a template, and the proposed multi-scale optimization generates optimal 2D texture coordinates for the faces. As a result, a texture atlas can be generated that enables the dynamic scene to have consistent and sharp appearance over time.

3. Overview

Fig. 2 illustrates the overall process of our framework to generate a complete texture atlas of a deforming mesh. Given a template mesh model and an RGBD stream, we first perform L0-L2 non-rigid registration [GXW*15] to estimate *warping fields* that align the template mesh to input depth maps. Then we sample *key frames* of color images based on the frame uniqueness scores that consider the visibility of deformed template meshes.

As the next step, we generate *mesh pyramids* of the deformed template meshes using deformation-oriented mesh decimation (DOD) [HCC06]. By projecting the mesh pyramids onto the key frames, we can determine the *valid key frames* for each triangle. With the projection, we can also obtain initial rough mappings onto the valid key frames for each triangle in the template mesh. We refine the sets of valid key frames using *majority voting* so that neighboring triangles in the template mesh have coherent sets of valid key frames. We then determine optimal projected texture coordinates of triangles on their valid key frames through our *multi-scale texture optimization*.

Finally, we build a texture atlas by applying mesh parameterization to the template mesh. For each triangle in the mesh, we collect sub-textures from the valid key frames using the optimized texture



(a) (b)

Figure 3: *Template mesh generation. (a) template model reconstructed by [YZG*18]. (b) Poisson surface reconstruction of (a).*

coordinates and blend them onto the proper position in the texture atlas determined by the mesh parameterization.

4. Dynamic Geometry Reconstruction

4.1. Template Model Acquisition

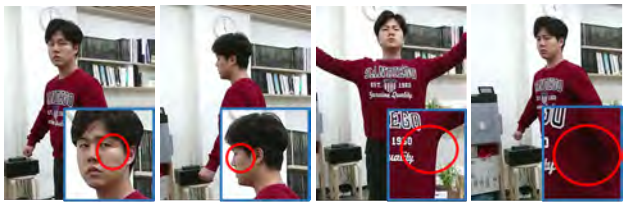
We configure our capturing system to fit for an ordinary scenario. We capture human subjects with a single camera in an indoor scene under natural illumination. Among a variety of reconstruction systems, we used Doublefusion [YZG*18] for building a template model of a human subject because of its robustness and real-time performance. For template model capture, the subject spreads his/her limbs and slowly turns around in front of a camera. If there are missing parts that are not visible to the camera, the subject changes his/her posture slightly. The remaining artifacts, like occluded parts or tearing holes, are processed with Poisson surface reconstruction [KH13] to guarantee watertightness of the recovered mesh. Fig. 3 shows an example.

4.2. Non-rigid Registration

Our framework uses non-rigid registration proposed by Guo et al. [GXW*15, GXW*18]. It aligns a template mesh model to a new input mesh, and the setting works well with our single camera-based geometry capture pipeline. The method basically carries out L_2 norm-based non-rigid registration and runs L_0 norm-based regularizer when articulated motions occur.

Given a template mesh, for each input RGB-D frame, the method of Guo et al. computes a warping field that determines the mapping of each triangle in the template mesh onto the input frame. We denote the warping field of a face f in the template mesh to the i-th input frame as G_f^i, where G_f^i determines the projected position of each vertex of f onto the i-th input frame. For more details, refer to [GXW*15, GXW*18].

We utilize the original implementation of Guo et al. for non-rigid alignment. It uses the Gauss-Newton method to optimize the warping field, but we observed that the Jacobian matrix of the energy function occasionally becomes rank-deficient for our human dataset. To relieve the problem, we adopted the idea of Li et al. [LVG*13] that adds a scaled identity matrix to the Gauss-Newton matrix.



(a) (b) (c) (d)

Figure 4: *Shading changes over time under uncontrolled indoor illumination. Although (a) and (b) indicate the same part, they exhibit different shading. In (c) and (d), the change is more drastic due to self-shadow.*

5. Global Texture Maps for Dynamic Objects

In this section, we describe detailed steps for our multi-scale texture optimization, which is the core of our framework to generate a global texture atlas.

5.1. Shape-aware Key Frame Selection

For color mapping of non-rigid objects, using all of the captured color images is redundant since most parts of the surface can be observable using a small number of different viewing directions. Existing works [ZK14, JJKL16] select color images having sharp textures within a pre-defined time window. However, the approach is intended for rigid scenes, and we propose a shape-aware approach to select key frames for texture mapping of dynamic objects.

Our key frame selection approach utilizes a frame uniqueness score. It is a joint measurement of viewing direction and object shape coverage for each frame. We first make a per-frame binary array to represent whether each face of the template mesh is visible or not at a certain timestamp. It can be determined by applying the warping fields to the template mesh and ray casting the resulting deformed meshes. Then, each visible face is evaluated with the angle between its normal direction and the viewing direction. The *uniqueness score* $Q(i)$ for a frame i aggregates the quality of the visible faces in frame i as follows:

$$Q(i) = \frac{\sum_{f \in \mathcal{F}^i} \frac{W_f^i}{N_f}}{|\mathcal{F}^i|}, \tag{1}$$

where $\mathcal{F}^i$ is the set of visible faces f in frame i, W_f^i is the dot product of the face f's normal and the camera's viewing direction, and N_f is the total number of frames that face f is visible.

The score $Q(i)$ becomes high when many faces are completely visible in a frame i. The score also reflects the uniqueness of the frame as the visibility weight W_f is divided by N_f, penalizing faces visible in many frames. We select key frames based on high uniqueness scores but avoid selecting temporally adjacent frames to minimize the number of similar color images.

5.2. Valid Key Frame Refinement using Majority Voting

Once the key frames have been selected from the input sequence, we determine the *valid* key frames for each face f in the template

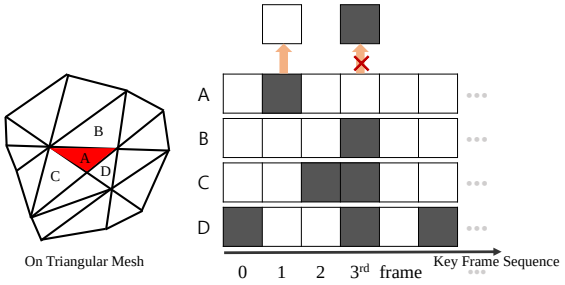


Figure 5: *Majority voting example for face 'A'. In the first key frame, face 'A' is set as visible (denoted as black cell in the table), but other adjacent faces are not visible (denoted as white). Majority voting updates the visibility of 'A' in the first key frame as invisible according to the consensus of adjacent faces.*

mesh, consisting of the key frames in which face f is visible. The valid key frames will be blended to obtain the sub-texture of the face in the final global texture atlas. In our framework, we capture an object without controlling the lighting condition in an indoor scene to simulate the ordinal capturing scenario. As a result, the shading of the same face can considerably change on different valid key frames (Fig. 4). Then, with visibility changes, the valid key frame sets of adjacent faces in the template mesh could consist of different key frames with shading changes. These practical issues may induce notable shading inconsistency between adjacent faces in the final texture mapping results.

In this regard, we design a way to enforce spatial coherency for the valid key frame sets. The valid key frame sets of faces can be represented as visibility index arrays for key frames, where an array represents the valid key frames of a face, summarizing the visibility of the face in each key frame (Fig. 5). We refine the sets of valid key frames using *majority voting* that updates the visibility of a face for a key frame based on the consensus from adjacent faces. That is, even though a face f was originally visible for a key frame i, it becomes invisible if the majority of the adjacent faces are invisible for frame i (face 'A' for the first frame in Fig. 5). However, this update does not change the visibility from invisible to visible regardless of the consensus from adjacent faces, as a key frame cannot become a valid key frame of a face if the face is invisible in the key frame (face 'A' for the third frame in Fig. 5). We iterate this update process over all faces of the template mesh. As a result, we obtain similar sets of valid key frames for spatially neighboring faces, which would help shading consistency for texture mapping.

We enforce a face to have at most k valid key frames for texture mapping. We repeatedly perform the majority voting process a few times while keeping the maximum number of valid key frames of faces less than k. To prevent the case that any face has no valid key frames, the number of valid key frames of each face is also kept to be above a predefined minimum value.

5.3. Global Texture Map Optimization

Given the template mesh, valid key frames, and warping fields $\{G_f^i\}$, which have been obtained in the previous stages, we compute optimal 2D texture coordinates of each vertex of the template mesh on valid key frames. By blending sub-textures extracted from the valid key frames using the optimized texture coordinates, we can generate a global texture map that can be consistently applied to any deformed mesh.

Single-level texture map optimization We first elaborate single-level texture map optimization, which will be extended to multi-scale optimization. After the processes in Sections 5.1 and 5.2, each face has the refined valid key frames, weights $\{W_f^i\}$ that will be used for blending to generate an atlas, and the initial texture coordinates in the valid key frames determined by warping fields $\{G_f^i\}$. The texture coordinates determine the sub-textures on valid key frames that correspond to the faces in the template mesh. With these components, we can perform texture map optimization, based on Jeon et al.'s work [JJKL16]. The optimization energy is to calculate the photometric inconsistency between sub-textures on valid key frames for the faces of the template mesh.

Let $\mathbf{V}$ be all vertices of the template mesh, and $\mathbf{F}_v$ be the one-ring neighbor faces of a vertex v. Then the energy is defined as:

$$E(\mathbf{u},\mathbf{P}) = \sum_{v\in\mathbf{V}} \sum_{f\in\mathbf{F}_v} \sum_{i\in\mathbf{I}_f} M(i,f), \tag{2}$$

where $\mathbf{u}$ denotes 2D texture coordinates of sub-textures to be optimized, $\mathbf{P}$ is a vector of proxy colors, and $\mathbf{I}_f$ is the set of valid key frames corresponding to f. M is the color inconsistency of sample pixels in a sub-texture, which is defined as:

$$M(i,f) = \sum_{s\in\mathbf{S}_f} ||C(u_i(s)) - P(s)||^2, \tag{3}$$

where $\mathbf{S}_f$ is the set of sample positions within a face f defined by fixed barycentric coordinates, $C(u_i(s))$ is the color value on the i-th valid key frame at the texture coordinates $u_i(s)$ corresponding to sample s, and $P(s) \in \mathbf{P}$ denotes the proxy color of sample s.

At the beginning of the optimization, for each sample s, we average the corresponding pixel values from the sub-textures to initialize the proxy color vector $\mathbf{P}$. In the following steps, we optimize $\mathbf{u}$ while $\mathbf{P}$ is fixed, and vice versa. This alternating optimization is repeated until convergence, resulting in a sharp texture. For more details of the optimization process, refer to [JJKL16].

In a single camera setup like ours, partial input data makes non-rigid registration less reliable. It results in inaccurate warping fields in some difficult scenes, including large motions and occlusions. As our optimization starts from initial texture coordinates calculated using the warping fields, large errors in the warping fields may cause texture optimization failure. This results in ghost artifacts as the sub-textures are not correctly aligned. To handle large misalignments, we introduce a *multi-scale approach* for more effective texture optimization.

Mesh pyramid construction Our multi-scale texture map optimization uses a mesh pyramid, which can be constructed by repeatedly simplifying the template mesh. We adopt Huang et al.'s work [HCC06] to simplify the sequence of deformed template meshes while minimizing shape distortions. The method is based on edge collapses, and naturally generates a *vertex tree* [Hop97],

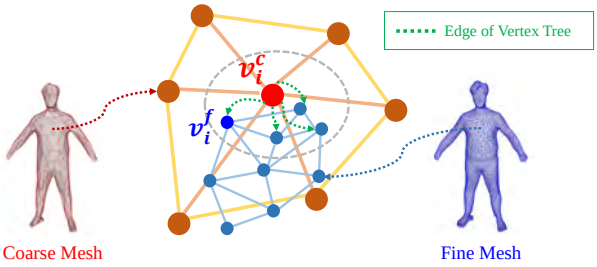


Figure 6: *Propagation of optimized texture coordinates from a coarse (red) to the finer (blue) mesh for multi-scale optimization. Vertices of two meshes are densely connected in the vertex tree.*

where the leaf nodes together represent the original mesh and any cut through intermediate nodes form a simplified mesh.

Multi-scale texture map optimization Our multi-scale texture optimization starts from the coarsest mesh and progresses to the finer meshes. At the coarsest level, we set the initial texture coordinates by utilizing estimated warping fields $\{G_f^i\}$, and then perform single-level texture coordinate optimization by minimizing Eq. (2). For finer levels, the optimized texture coordinates of a coarse mesh are propagated to provide the initial solution of Eq. (2) for further optimization. For propagation of texture coordinates from a coarse mesh to the finer mesh, we use the vertex tree constructed during mesh simplification.

Fig. 6 illustrates the texture coordinate propagation process. For a vertex v^f in the finer mesh (colored in blue), we first find the its parent vertex v^c (colored in red) in the coarse mesh using the vertex tree (colored in green). Then, we find the 1-ring neighbor vertices of v^c in the coarse mesh (connected with yellow edges). Finally, the initial texture coordinates of v^f are determined using weighted-average of the 2D texture coordinates of the coarse vertices:

$$u(v^f) = \sum_{v_j^c \in \mathbf{N}(v^f)} w_j u(v_j^c), \tag{4}$$

where $\mathbf{N}(v^f)$ is the neighborhood of the parent node of v^f and $u(\cdot)$ denotes the texture coordinates. The weight w_j is inversely proportional to the Euclidean distance between v_j^c and v^f.

GPU acceleration We use CUDA for GPU-accelerated optimization as in [JJKL16]. To deal with a mesh pyramid, the approach is extended for seamlessly handling multi-scale meshes. After computing the optimized texture coordinates for a coarse mesh on GPU, we temporarily copy the results to the host memory and perform the coarse-to-fine texture coordinate propagation. We then copy the propagated texture coordinates back to the GPU memory as the initial solution for the finer mesh, and continue the optimization.

5.4. Texture Atlas Generation

Our approach utilizes a template model with an assumption that the mesh connectivity does not change during deformation. We apply a mesh parameterization method [Mic11] to the template mesh and obtain a texture atlas that can be shared by all deformed template mesh. Then, for each triangle in the texture atlas, we collect all the corresponding sub-textures from valid key frames with optimized texture coordinates, and blend them with the pre-calculated weights W_f^i to determine the texture information. In this manner, we can build a sharp and clean global texture atlas that can be applied for any deformed mesh.

6. Results

Experiment details We validate our framework with a simple single-camera setup under natural indoor illumination condition. To this end, we scanned humans wearing diverse clothes, a blanket, and a human wearing hand puppet. All datasets shown in this paper were recorded by a Kinect for Windows v2 camera or ASUS Xtion PRO LIVE, except additional datasets provided by [GXW*15]. From the generation of a template model to reconstruction of a global texture atlas, all steps proceeded without any special-purpose environments, such as turntable and studio environment. Please check the supplementary video for a reference.

We generated all human template models using DoubleFusion [YZG*18]. For the blanket and the hand puppet scenes, we used DynamicFusion [NFS15] because DoubleFusion [YZG*18] is specialized for human reconstruction. We used fixed parameters for the two core parts of our framework, i.e., non-rigid registration and texture map generation. We set k to 30, which is the maximum number of assigned key frames for each triangle in the template mesh, and the minimum number is set to four. The number of mesh pyramid levels is set to five except that it is set to three for a blanket and a hand puppet scenes. The maximum iteration number for texture optimization is set to 200 at each mesh pyramid level.

Computation time All experiments were conducted on a workstation equipped with an Intel i7-8700K 3.7GHz CPU, 64GB RAM, and NVIDIA Titan Xp GPU. Our RGB-D sequences used for experiments contain 200 to 500 frames. After obtaining the warping fields using the method of Guo et al. [GXW*15,GXW*18], it takes about three minutes to perform the process described in Section 5.1 and 5.2 before texture optimization. Following [JJKL16], our optimization is accelerated by parallel calculation of gradients for 2D texture coordinates. With our test scenes, a single iteration of texture coordinate optimization takes about 90 $\sim$ 110 milliseconds, and more iterations are needed for coarser meshes to achieve large changes of texture coordinates. It takes about 2 $\sim$ 3 minutes in total to obtain the optimized texture coordinates for the global texture atlas of a deforming mesh.

Texture mapping of dynamic human models We tested our texture map optimization with various dynamic human models. As shown in Fig. 11, directly using initial texture coordinates for texture mapping produces serious blur artifacts, due to inaccurately estimated warping fields and camera parameters. Our single-level texture optimization shows prominent improvement over näive initial mapping, but it does not cover large misalignments caused by dynamic mesh deformations. In contrast, our multi-scale approach provides a good solution for dynamic objects.

Non-human objects Fig. 7 shows the results of our texturing method applied to other dynamic scenes with non-human objects.

(a) (b) (c)

Figure 7: *Results on non-human objects. (a) Template models. (b) Näive texture mapping using initial warping fields. (c) Our results.*

Although the deforming geometry is diverse and the estimated warping fields are not perfect, our method exhibits texture mapping results with considerably good visual quality.

Valid key frame refinement Our framework does not handle varying lighting conditions, so the final blended sub-textures stored in the global texture atlas may have inconsistent shading. Fig. 8 shows an example that our valid key frame refinement algorithm relieves the inconsistency problem.

Robustness The fourth row of Fig. 11 shows that our method can generates a consistent texture map even with quite inaccurate warping fields. Fig. 9 demonstrates that our method can still generate high quality texture mapping results although the estimated extrinsic parameters of the camera are inaccurate.

Limitations We assume that the estimated warping fields are reliable to some degree. If non-rigid alignment fails due to fast object motions, it may lead to a failure of texture optimization. Fig. 10 shows an example, which was made by skipping every two frames from the original RGB-D image sequence to simulate a fast motion.

Although our approach optimizes the texture coordinates to generate a clean texture atlas, it cannot handle texture variations on the same part of the model over time as a single global texture atlas is used for a dynamic object. Fig. 10 shows an example of the eye movement during the capture process.

Our approach optimizes texture coordinates without refining the geometry, so texture drift may occur if geometry of template mesh is not accurate. In addition, our framework assume that the connectivity of the template mesh does not change over time. Our future work includes resolving these limitations.

7. Conclusion

We proposed a novel framework to generate a complete texture atlas for a dynamic object in single RGB-D camera setup. Our framework combines sub-texture coordinates with warping fields to formulate an optimization energy for texture mapping of dynamic objects. To this end, our framework performs key frame sampling using uniqueness score, majority voting to refine valid key frame sets, and multi-scale texture coordinate optimization using a coarse to fine mesh pyramid. Especially, our multi-scale texture coordinate optimization enables us to handle large misalignments that occurred by inaccurately estimated warping fields or camera parameters. We demonstrated that our framework is reliable and resilient to inaccurate camera parameters and warping fields. Using only a single fixed RGB-D camera to obtain optimized texture atlas for a dynamic object, our framework is more flexible than studio setups.

(a) (b)

Figure 8: *Effect of our refinement of valid key frames using majority voting. (a) Result without refinement. (b) Result with refinement. By enforcing neighboring faces to have similar valid key frames, we obtain seamless textures around the neck.*

(a) (b) (c)

Figure 9: *Resilience to inaccurate camera parameters. (a) Input RGB-D frame. We apply our approach to a dataset provided by [GXW*15] which exhibits misalignments as highlighted in red ellipses. (b) Näive texture mapping. (c) Multi-scale texture optimization. Our texture optimization robustly reconstructs a clean textured mesh even if the camera parameters are not accurate.*

Acknowledgements This work was supported by the Ministry of Science and ICT, Korea, through Giga Korea grant (GK19P0300), IITP grant (IITP-2015-0-00174), and NRF grant (NRF-2017M3C4A7066317).

References

[BKR17] BI S., KALANTARI N. K., RAMAMOORTHI R.: Patch-based optimization for image-based texture mapping. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 106:1–106:11. 2, 3

[CCS*15] COLLET A., CHUANG M., SWEENEY P., GILLETT D., EVSEEV D., CALABRESE D., HOPPE H., KIRK A., SULLIVAN S.: High-quality streamable free-viewpoint video. *ACM Transactions on Graphics (TOG) 34*, 4 (2015), 69:1–69:13. 2

[CZK15] CHOI S., ZHOU Q.-Y., KOLTUN V.: Robust reconstruction of indoor scenes. In *CVPR* (2015). 1

[DDF*17] DOU M., DAVIDSON P., FANELLO S. R., KHAMIS S., KOWDLE A., RHEMANN C., TANKOVICH V., IZADI S.: Motion2fusion:

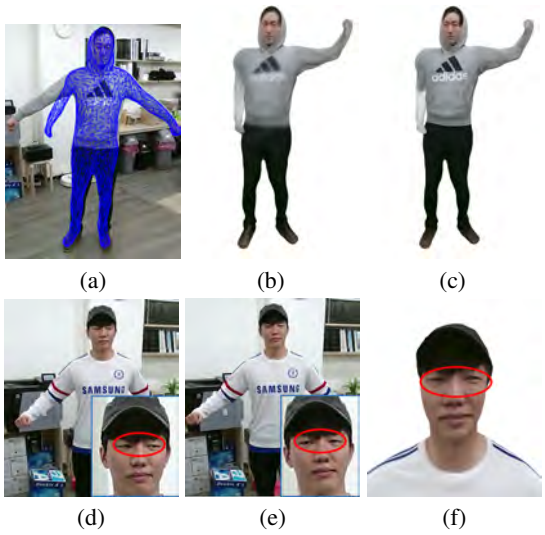


Figure 10: *Failure cases due to wrong warping fields (top) and extreme texture variation (bottom). In the top row, to obtain a fast motion, we skipped every two frames from the original RGB-D image sequence. (a) Wrong motion estimation. (b) Before texture map optimization. (c) Background texture remains on the right arm even after optimization. In the bottom row, the pupil movement of the subject incurs texture variation in the valid key frames shown in (d) and (e). Although texture coordinates are optimized by our scheme, the texture variation leads to blurriness in the eye parts of the final textured model in (f).*

Real-time volumetric performance capture. *ACM Transactions on Graphics (TOG) 36*, 6 (2017), 246:1–246:16. 2

[DKD*16] DOU M., KHAMIS S., DEGTYAREV Y., DAVIDSON P., FANELLO S. R., KOWDLE A., ESCOLANO S. O., RHEMANN C., KIM D., TAYLOR J., KOHLI P., TANKOVICH V., IZADI S.: Fusion4d: Real-time performance capture of challenging scenes. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 114:1–114:13. 2

[DNZ*17] DAI A., NIESSNER M., ZOLLÖFER M., IZADI S., THEOBALT C.: Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface re-integration. *ACM Transactions on Graphics (TOG) 36*, 3 (2017), 24:1–24:18. 1

[DTF*15] DOU M., TAYLOR J., FUCHS H., FITZGIBBON A., IZADI S.: 3d scanning deformable objects with a single rgbd sensor. In *CVPR* (2015). 2

[FYY*18] FU Y., YAN Q., YANG L., LIAO J., XIAO C.: Texture mapping for 3d reconstruction with rgb-d sensor. In *CVPR* (2018). 2, 3

[GWO*10] GAL R., WEXLER Y., OFEK E., HOPPE H., COHEN-OR D.: Seamless montage for texturing models. *Computer Graphics Forum (CGF) 29*, 2 (2010), 479–486. 2, 3

[GXW*15] GUO K., XU F., WANG Y., LIU Y., DAI Q.: Robust non-rigid motion tracking and surface reconstruction using L0 regularization. In *ICCV* (2015), pp. 3083–3091. 1, 2, 3, 4, 6, 7

[GXW*18] GUO K., XU F., WANG Y., LIU Y., DAI Q.: Robust non-rigid motion tracking and surface reconstruction using L0 regularization. *IEEE Transactions on Visualization and Computer Graphics (TVCG) 24*, 5 (2018), 1770–1783. 2, 4, 6

[HCC06] HUANG F.-C., CHEN B.-Y., CHUANG Y.-Y.: Progressive deforming meshes based on deformation oriented decimation and dynamic connectivity updating. In *SCA* (2006). 2, 3, 5

[Hop97] HOPPE H.: View-dependent refinement of progressive meshes. In *ACM SIGGRAPH* (1997). 5

[IZN*16] INNMANN M., ZOLLHÖFER M., NIESSNER M., THEOBALT C., STAMMINGER M.: Volumedeform: Real-time volumetric non-rigid reconstruction. In *ECCV* (2016). 2

[JJKL16] JEON J., JUNG Y., KIM H., LEE S.: Texture map generation for 3d reconstructed scenes. *The Visual Computer 32*, 6-8 (2016), 955–965. 2, 3, 4, 5, 6

[KH13] KAZHDAN M., HOPPE H.: Screened poisson surface reconstruction. *ACM Transactions on Graphics (TOG) 32*, 3 (2013), 29:1–29:13. 4

[LAGP09] LI H., ADAMS B., GUIBAS L. J., PAULY M.: Robust single-view geometry and motion reconstruction. In *ACM SIGGRAPH Asia* (2009). 2

[LCL*16] LIN S., CHEN Y., LAI Y.-K., MARTIN R. R., CHENG Z.-Q.: Fast capture of textured full-body avatar with rgb-d cameras. *The Visual Computer 32*, 6-8 (2016), 681–691. 3

[LGY19] LI W., GONG H., YANG R.: Fast texture mapping adjustment via local/global optimization. *IEEE Transactions on Visualization and Computer Graphics (TVCG) 25*, 6 (2019), 2296–2303. 2, 3

[LMR*15] LOPER M., MAHMOOD N., ROMERO J., PONS-MOLL G., BLACK M. J.: SMPL: A skinned multi-person linear model. *ACM Transactions on Graphics (TOG) 34*, 6 (2015), 248:1–248:16. 2

[LVG*13] LI H., VOUGA E., GUDYM A., LUO L., BARRON J. T., GUSEV G.: 3d self-portraits. *ACM Transactions on Graphics (TOG) 32*, 6 (2013), 187:1–187:9. 2, 4

[Mic11] MICROSOFT: *UVAtlas*, 2011. URL: https://github.com/Microsoft/UVAtlas. 6

[NFS15] NEWCOMBE R. A., FOX D., SEITZ S. M.: Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *CVPR* (2015). 2, 6

[NIH*11] NEWCOMBE R. A., IZADI S., HILLIGES O., MOLYNEAUX D., KIM D., DAVISON A. J., KOHI P., SHOTTON J., HODGES S., FITZGIBBON A.: Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR* (2011). 1, 2

[NZIS13] NIESSNER M., ZOLLHÖFER M., IZADI S., STAMMINGER M.: Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (TOG) 32*, 6 (2013), 169:1–169:11. 1

[PKC*17] PRADA F., KAZHDAN M., CHUANG M., COLLET A., HOPPE H.: Spatiotemporal atlas parameterization for evolving meshes. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 58:1–58:12. 2, 3

[PZK17] PARK J., ZHOU Q.-Y., KOLTUN V.: Colored point cloud registration revisited. In *ICCV* (2017). 1

[SSP07] SUMNER R. W., SCHMID J., PAULY M.: Embedded deformation for shape manipulation. In *ACM SIGGRAPH* (2007). 2

[WSMG*16] WHELAN T., SALAS-MORENO R. F., GLOCKER B., DAVISON A. J., LEUTENEGGER S.: Elasticfusion: Real-time dense slam and light source estimation. *International Journal of Robotics Research (IJRR) 35*, 14 (2016), 1697–1716. 1

[YGX*17] YU T., GUO K., XU F., DONG Y., SU Z., ZHAO J., LI J., DAI Q., LIU Y.: Bodyfusion: Real-time capture of human motion and surface geometry using a single depth camera. In *ICCV* (2017). 2

[YZG*18] YU T., ZHENG Z., GUO K., ZHAO J., DAI Q., LI H., PONS-MOLL G., LIU Y.: Doublefusion: Real-time capture of human performances with inner body shapes from a single depth sensor. In *CVPR* (2018). 2, 4, 6

[ZK14] ZHOU Q.-Y., KOLTUN V.: Color map optimization for 3d reconstruction with consumer depth cameras. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 155:1–155:10. 2, 3, 4

[ZNI*14] ZOLLHÖFER M., NIESSNER M., IZADI S., REHMANN C., ZACH C., FISHER M., WU C., FITZGIBBON A., LOOP C., THEOBALT C., STAMMINGER M.: Real-time non-rigid reconstruction using an rgb-d camera. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 156:1–156:12. 2

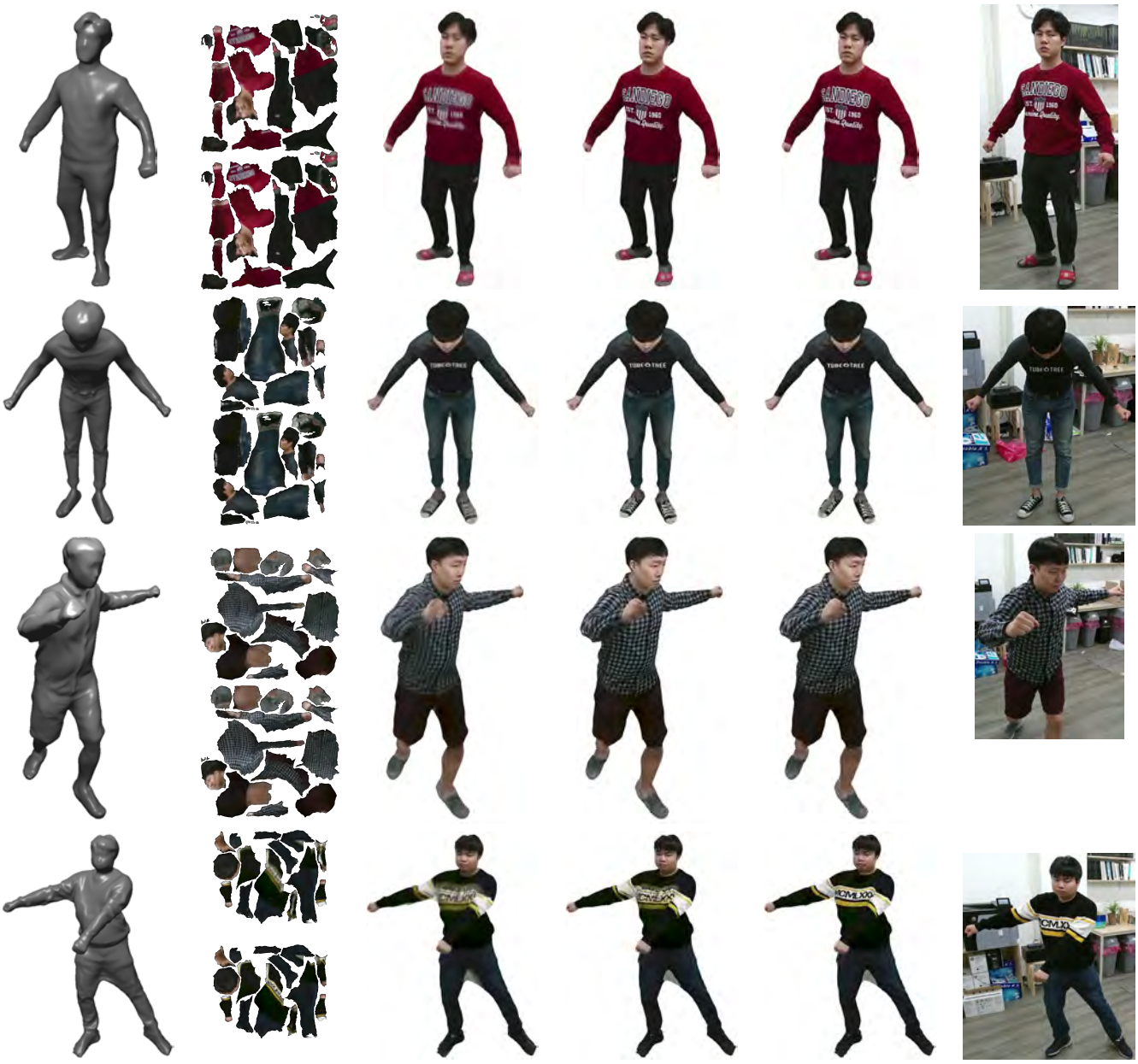


Figure 11: *Global texture mapping on dynamic human models. (a) Geometry of template models. (b) Global texture maps without (upper) and with (lower) multi-scale texture optimization. (c) Texture mapping using initial warping fields. (d) Single-scale texture optimization. (e) Multi-scale texture optimization. (f) Corresponding input color images. Texture mapping results in (e) are clearly better than those in (c) and (d). Our multi-scale optimization produces reliable texture mapping results that are comparable to the input color images.*