Scanning and Tracking Dynamic Objects with Commodity Depth Cameras

Mingsong Dou Henry Fuchs Jan-Michael Frahm*

Department of Computer Science
University of North Carolina at Chapel Hill

(a) Pre-aligned Point Clouds (b) Post-aligned Point Clouds (c)Cross Section of Aligned Surfaces (d) Fused Scan (e) Fused Scan with Color

Figure 1: Scanned Dynamic Objects. (a)&(b) show 150 frames of point clouds of one moving person before and after alignment. (c) shows the cross section of the aligned surfaces at the upper body and leg (surfaces are assigned different colors). (d)&(e) show the fused 3D model.

ABSTRACT

The 3D data collected using state-of-the-art algorithms often suffer from various problems, such as incompletion and inaccuracy. Using temporal information has been proven effective for improving the reconstruction quality; for example, KinectFusion [21] shows significant improvements for static scenes. In this work, we present a system that uses commodity depth and color cameras, such as Microsoft Kinects, to fuse the 3D data captured over time for dynamic objects to build a complete and accurate model, and then tracks the model to match later observations. The key ingredients of our system include a nonrigid matching algorithm that aligns 3D observations of dynamic objects by using both geometry and texture measurements, and a volumetric fusion algorithm that fuses noisy 3D data. We demonstrate that the quality of the model improves dramatically by fusing a sequence of noisy and incomplete depth data of human and that by deforming this fused model to later observations, noise-and-hole-free 3D models are generated for the human moving freely.

1 INTRODUCTION

Having a high quality 3D model of the scene is critical for many applications, such as novel view generation, model-based tracking, etc. Current state-of-the-art 3D reconstruction algorithms suffer various problems, such as incompletion (due to occlusions) and inaccuracy. Fig. 6 shows examples of captured 3D data from eight Kinect cameras, where holes and noise are obvious. In prior work, using temporal information has been proven useful for improving the reconstruction quality. For example, KinectFusion [21] aligns and fuses depth map from a moving camera to get a complete and comparatively high-quality model of a static scene. Unfortunately, it is not as straightforward to use temporal information for dynamic objects due to their nonrigid movements. The extended version of KinectFusion [16] uses the rigid transformation model for dynamic objects and weighted heavily on latest observations during fusion to ameliorate the artifacts from nonrigid movements.

In this paper, we explore a more sophisticated usage of temporal information in 3D reconstruction of dynamic objects. We introduce a 3D capture system that first builds a complete and accurate 3D model for dynamic objects (e.g. human body) by fusing a data sequence captured by commodity depth and color cameras, and then tracks the fused model to align it with following captures. One crucial component of our system is the nonrigid alignment of the depth data at different instants during both scanning and tracking stages. Inspired by the work of Li et al. [17][18], we integrate the measurement of both dense point cloud alignment and color consistency into an energy minimization problem, which is then solved efficiently by a gradient descent method. Our system also extends the volumetric fusing algorithm to accommodate noisy 3D data during the scanning stage. Specifically, we introduce a new representation of 3D data—the Directional Distance Function (DDF), which is an extension of Truncated Signed Distance Function [10] by adding a direction field pointing to the nearest points on the surface along with the signed distance field. The new data representation helps us to solve nonrigid matching algorithm more efficiently as well. We introduce the nonrigid matching algorithm in Sec. 2, the scanning system in Sec. 3, and the tracking system in Sec. 4.

1.1 Related work

KinectFusion shows its success on scanning static scenes in real-time in [21] and shows results on processing dynamic objects with rigid matching in [16]. Our work aims at scanning and tracking dynamic objects by employing a non-rigid alignment algorithm. However, as many other nonrigid matching algorithm algorithms, our system does not perform in real-time in current single-threaded

*e-mail:{doums, fuchs, jmf}@cs.unc.edu

CPU implementation. By taking advantage of the GPU technology and Hierarchy computation strategy as KinectFusion does, we expect boosted performance in our system.

Previous work on fusing multiple scans of a dynamic object either restricts the motion of the object and turns objects around with a turntable [25], or uses a human model database (e.g., SCAPE [2]) as a prior knowledge for human body reconstruction [15][27], which gives up the flexibility to scan other objects or humans wearing loose clothes. Our scanning system allows for dramatic movements of the subjects being scanned and makes no assumption on the shape of the object being scanned.

Our work also relates to research into Motion/Performance Capture of articulated dynamic objects [13][26][4]. These works presume the surface model of the object being tracked is available. They also need a manual skinning procedure, i.e., attach the pre-scanned surface to a skeleton structure or kinematic tree. The skeleton parameters are then estimated by matching the skeleton-driven surface with the current observation, such as silhouette. A later surface refinement stage is usually employed to reduce the artifacts on the deformed surface according to skeleton model. For example, [13][26] used the technique from Laplacian Shape Editing [23]. Our system instead incorporates a scanning procedure that takes advantage of depth sensors and does not need the manual operations such as skinning or rigging. While the systems above typically need to use a blue-screen to get the object silhouette, our system has no such requirement. De Aguiar et al. [11] abandoned the skeleton and uses a coarse tetrahedral version of the scan instead, but it still involves manual procedures during initialization.

The skeleton model above only applies to articulated objects, such as human beings. Sumner et al. [24] proposed a more general motion model for shape manipulation—the Deformation Graph Model, which is capable of preserving geometry details while still provides desired properties such as simplicity and efficiency. Thus, it has been used to parameterize the non-rigid deformation and applied to shape matching [17], which in turn is used for dynamic shape completion [18] and surface tracking [7]. Inspired by these works, we apply the Deformation Graph Model on dynamic model scanning and tracking using the noisy data from commodity depth cameras, and we incorporate both dense depth and color information into the non-rigid matching framework.

Our work and most related works above could be categorized into one broad research area—Non-rigid Matching. Besides the research above that reduces the computation space by modeling the nonrigid movement with a rough skeleton model or a denser Deformation Graph Model, there exist other works that directly compute the deformed template surface that aligns with the target. Among those works, the Bronstein brothers proposed a framework of deforming surface while keeps their geometrically intrinsic properties unchanged [8][22]. This isometric invariance constraint penalizes the change of geodesic distance between any points on the surface, and thus prevents the surface from stretching and shrinking. These works are computationally intensive. We propose a simpler method to keep the isometric invariance.

Our work relates to the research into 3D Character Animation [5][9] as well. Baran et al. [5] automatically fit a skeleton to the character mesh and attach it to the surface. Chen et al. [9] designed a real-time system that transfers user's motion to any pre-scanned objects. The user's skeleton from a Kinect camera is attached to the Deformation Graph [24] of the character, and the character is then deformed accordingly. Instead of only transferring user's motion, we aim at a tight surface alignment between a user's pre-scan and the later observation.

2 NONRIGID ALIGNMENT

When deforming a **template surface** to match with a **target surface**, we want the corresponding points to be as close as possible. However, the point correspondence is unknown before the alignment. One general solution is an ICP-like method [17] (i.e. iteratively estimate the point correspondence and alignment parameters), but that is inefficient. Another option is to first pre-compute the Signed Distance Fields [10] for the target surface, through which the surface alignment error is measured directly without knowledge of point correspondence. Then the best alignment is found by minimizing the error with a gradient descent method. LM-ICP [12] uses the same idea for rigid object alignment, but estimates the descent direction with finite difference, which is problematic for noisy surfaces. We propose to represent the template surface as a Directional Distance Function by adding a direction field pointing to the nearest points on the surface along with the signed distance field. In this way, we can deduce an analytic solution for the derivatives of the measurement function, from which we compute the gradient descent direction. In addition, we integrate a color consistency constraint into the framework such that its derivative also has an analytical solution, which allows the problem to be solved efficiently and robustly.

The deformation between surfaces are parameterized by the Deformation Graph Model [24] which is reviewed in next section.

2.1 Review of Deformation Graph Model

In the Deformation Graph Model proposed by Sumner et al. [24], a deformation is represented by a collection of affine transformations. A number of nodes (typically several hundred) are uniformly sampled from the template surface. In addition to its location g_i, each node $\mathbf{n}_i$ has an affine transformation matrix A_i and a translation vector t_i associated with it, representing the local affine transformation around the graph node. Neighboring nodes connect to each other and collectively form a **Deformation Graph** $\mathbb{G} = \{\langle A_j, t_j, g_j \rangle\}_{j=1}^{J}$ (an example is shown in Fig. 2(c)). Any given point v on the template surface could be deformed by applying a linearly blended affine transformation from its neighboring graph nodes $\mathbb{N}$,

$$\tilde{v} = \sum_{j \in \mathbb{N}} w_j \left[A_j (v - g_j) + g_j + t_j \right], \tag{1}$$

where w_j is the blending weight and depends on v's geodesic distance to the graph node $\mathbf{n}_j$. The surface normal is transformed according to,

$$\tilde{n}_i = \sum_{j \in \mathbb{N}} w_j A_j^{-1\mathrm{T}} n_i. \tag{2}$$

During nonrigid alignment, $\{\langle A_j, t_j \rangle\}$ is estimated for the deformation graph by solving,

$$\min_{\{\langle A_j, t_j \rangle\}} w_{rot} E_{rot} + w_{reg} E_{reg} + w_{con} E_{con} \tag{3}$$

where,

$$\begin{aligned} E_{rot} = \sum_j & ((a_1^T a_2)^2 + (a_1^T a_3)^2 + (a_2^T a_3)^2 + \\ & (1 - a_1^T a_1)^2 + (1 - a_2^T a_2)^2 + (1 - a_3^T a_3)^2) + \\ & c(\det(A_j) - 1)^2, \end{aligned} \tag{4}$$

which constrains the column vectors a_1, a_2, a_3 of A_j to being orthogonal and unitary. Unlike [24], we constrain the determinant of A_j to being 1, which prevents flipping the surface normals. c is a constant, and we let $c = 100$ in our experiments.

An additional regularization term E_{reg} ensures the smoothness of the deformation:,

$$E_{reg} = \sum_{j=1}^{J} \sum_{k \in \mathbb{N}(j)} \| A_j (g_k - g_j) + g_j + t_j - (g_k + t_k) \|_2^2. \tag{5}$$

E_{reg} constrains that when deforming $\mathbf{n}_k$ with its neighbor $\mathbf{n}_i$'s affine transformation, it does not deviate dramatically from the deformation with its own affine transformation. The third term E_{con} comes from the matched key points $\{\langle v_i, q_i\rangle\}$ of two surfaces,

$$E_{con} = \sum_i \|\tilde{v}_i - q_i\|_2^2, \tag{6}$$

where $\tilde{v}_i$ is the deformed v_i from Eq. 1. In our case, the key points are Lucas-Kanade corner points that are converted to 3D points from 2D image locations using their observed depth.

The energy terms above are inadequate to align noisy 3D data from commodity depth cameras. Therefore, we include two more terms—dense point cloud alignment E_{dns_pts} (Section 2.2) and color consistency E_{clr} (Section 2.3), transforming Equation 3 to

$$\begin{aligned}\min_{\{\langle A_j, t_j\rangle\}} \quad & w_{rot}E_{rot} + w_{reg}E_{reg} + w_{con}E_{con} + \\ & w_{dns\_pts}E_{dns\_pts} + w_{clr}E_{clr}\end{aligned} \tag{7}$$

2.2 Directional Distance Function and Measurement of surface alignment

The matched key points in Sec. 2.1 are sparse features on the surface; their alignment does not represent the global surface alignment. Thus the dense alignment must be measured. Different from Li et al. [17] who iteratively estimate dense point correspondence, we represent the target surface as a distance field so that the surface alignment can be efficiently measured. At each voxel of this volume data, we record its distance $\mathscr{D}(\cdot)$ and direction $\mathscr{P}(\cdot)$ to its closest point on the surface. This representation is an extension to the Signed Distance Function (SDF) [10], and we call it the Directional Distance Function (DDF). Then, the energy function for dense point cloud alignment is defined by,

$$E_{dns\_pts} = \sum_i \|\mathscr{D}(\tilde{v}_i)\|_2^2 \tag{8}$$

where $\tilde{v}_i$ is the deformed template surface point v_i via Eq. 1.

Calculating the DDF takes no more effort than recording the position of nearest point on surface and subtracting its position to get $\mathscr{P}(\cdot)$. The pseudo-code of calculating the DDF from a surface or depth map is given in Appendix A. Note that the voxel whose closest surface point lies at the boundary of an open surface is set to null (or empty), which prevents the undesired surface extension when recovering a triangle mesh from a DDF. The surface boundary is identified either as pixels on a depth map that have depth discontinuity with their neighbors or the vertices on a triangular mesh that do not share its edge with other triangles.

$\mathscr{P}(\cdot)$ in the Directional Distance Function is especially helpful when minimizing Eq. 7. Since the energy function is in least squares form, it can be efficiently solved via a gradient descent-like method (e.g. Gauss-Newton algorithm) as long as the Jacobian matrix J is provided. To solve this nonlinear least squares problem, we use the Levenberg-Marquardt algorithm [19] implemented by the Google Ceres solver [1]. The Jacobians for first three terms of Eq. 7 are straightforward; we will illustrate those for E_{dns_pts} here and E_{clr} in next section. One interesting fact about SDF $\mathscr{D}(\cdot)$ is that its gradient is a unit vector aligned with the direction pointing to the closest surface point, i.e., $\mathscr{P}(\cdot)$. More precisely, since we use a negative SDF for voxels inside a surface,

$$\nabla\mathscr{D} = \begin{cases} -\mathscr{P}, & \text{if } \mathscr{D} > 0 \\ \mathscr{P}, & \text{if } \mathscr{D} < 0 \\ \text{normal}, & \text{if } \mathscr{D} = 0. \end{cases} \tag{9}$$

Thus, the Jacobians $\frac{\partial}{\partial p_k}\mathscr{D}(\tilde{v}_i) = \nabla\mathscr{D}|_{\tilde{v}_i}\frac{\partial}{\partial p_k}\tilde{v}_i$, where p_k is the k-th deformation parameter. When computing DDF in practice, we align $\mathscr{P}$ to surface normal when $|\mathscr{D}| < \varepsilon$, which is analogous to using the point-to-plane distance instead of the point-to-point distance for ICP. In our implementation, ε is set to 1.5 cm.

Sometimes there are parts on the target surface where front and back surfaces are close enough that the deformed surface point $\tilde{v}_i$ is attracted to the wrong surface during iterations. Fortunately, $\nabla\mathscr{D}|_{\tilde{v}_i}$ is the approximation of the normal of $\tilde{v}_i$'s closest point on the target. When the normal $\tilde{n}_i$ on the deformed reference surface does not agree with $\nabla\mathscr{D}|_{\tilde{v}_i}$, this means $\tilde{v}_i$ is heading to the wrong side of the target surface. To resolve this, we let $\frac{\partial}{\partial p_k}\mathscr{D}(\tilde{v}_i) \leftarrow 0$, if $\nabla\mathscr{D}|_{\tilde{v}_i} \cdot \tilde{n}_i < 0$, nullifying the attraction from the wrong part of the target.

Algorithm 1: Scanning Pipeline

Set the data from the first frame as the reference;
foreach *new frame* **do**
- **1.** fuse depth maps to get the DDF;
- **2.** match the reference surface to the new observation;
- **3.** transform the DDF from target to reference;
- **4.** fuse the transformed DDF into MDDF at reference;
- **5.** generate the reference surface from MDDF, and map colors to the surface;

2.3 Color Consistency

When deforming the template surface to the target, the matched points must have similar color (or texture). The E_{clr} term helps resolve alignment ambiguities when a near-symmetric part on the surface rotates or moves, such as head turns and arm rotations. In our scanning and tracking system, the template surface is the currently accumulated 3D model from the depth and color of previous frames, and it is represented by a triangle mesh with a 3D color vector c_i attached at each vertex. The target surface is the current observation of the dynamic object, and its raw representation is a set of depth maps $\{Z_k(\cdot)\}$ and color images $\{I_k(\cdot)\}$.

All the depth and color cameras are calibrated under the same world coordinate system, and $P_k(\cdot)$ projects a 3D point to the k-th image coordinate. Thus, the color consistency term in Eq. 7 is

$$E_{clr} = \sum_k \sum_i \|\delta_k(\tilde{v}_i) \cdot [I_k(P(\tilde{v}_i)) - c_i]\|_2^2, \tag{10}$$

where $\delta_k(\tilde{v}_i)$ is the visibility term; $\delta_k(\tilde{v}_i) = 1$ when $\tilde{v}_i$ is visible to the k-th color camera, and 0 when invisible. Visibility checking is performed with a z-buffer algorithm. We also set $\delta_k = 0$ for vertices whose outward normals point away from the camera, to prevent holes in the incomplete front surface from erroneously letting parts of back-facing surfaces pass the z-buffer test.

The Jacobians for E_{clr} also have an analytic solution: $\frac{\partial}{\partial p_i}I_k^c(P(\tilde{v}_i)) = \nabla I_k^c \cdot \frac{\partial}{\partial \tilde{v}_i}P \cdot \frac{\partial}{\partial p_i}\tilde{v}_i$, where ∇I_k^c is the image gradient for the c-th channel of the k-th color image. Note that the visibility check needs to be performed at each iteration of the gradient descent method since each iteration produce a differently deformed template surface. Fortunately, it does not take much more effort since we need to project vertices to image spaces anyway.

This color consistency term essentially estimates the optical flow that matches 3D surface points to 2D image coordinates, while classical 2D optical flow performs matching in the image space [3]. Even though the 3D flow technique [14] estimate the dense 3D motion with RGB-D data, the matching is still confined in the 2D image space. In addition, our algorithm does not require that the color and depth image be aligned, making it possible to use high resolution color cameras.

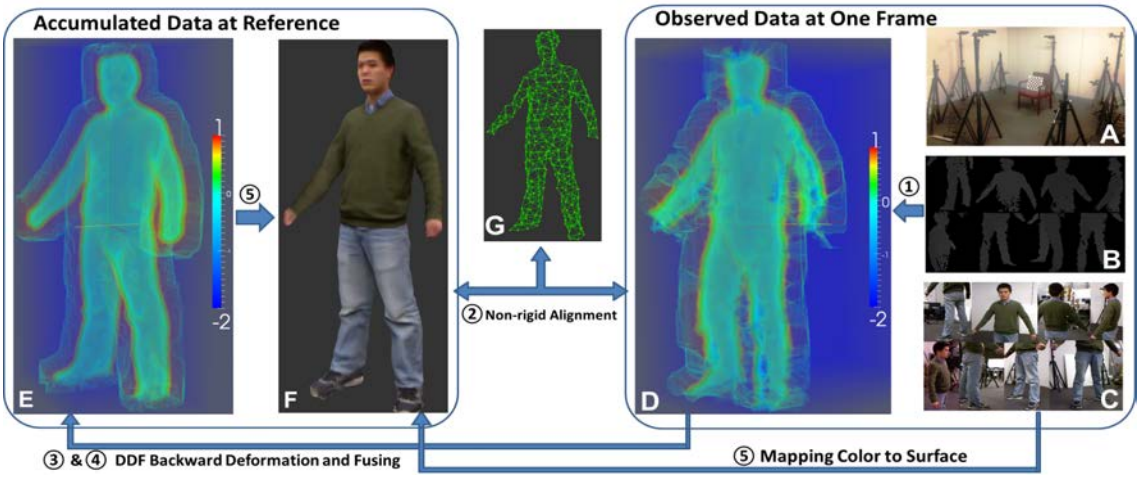


Figure 2: The Scanning System. ①–⑤ the steps in Algorithm 1; (A) the Eight-Kinect System Setup; (B) depth maps; (C) color images; (D) volume visualization of the fused DDF from 8 depth maps; (E) volume visualization of the accumulated DDF at the reference so far; (F) the accumulated reference surface so far; (G) the deformation graph on the surface (F).

3 Data Fusion for Dynamic Objects

Data fusion for dynamic objects is to acquire a complete and accurate model from a sequence of depth maps and color images captured by several commodity depth and color cameras. In our system, eight Kinects are used and placed in a circle with radius of 1.8 meters. Four of them cover the upper space, and the other four cover the lower space. Our system setup is shown in Fig. 2(a).

We use a similar data fusion pipeline as KinectFusion [21] for a static scene. Initially, the data of the first frame is set as the reference; then we repeatedly estimate the deformation between the reference and newly observed data, and fuse the observed data to the reference. More specifically, we perform following steps for each newly observed frame:

1. Convert depth maps from the Kinects to Directional Distance Functions (DDFs), and fuse them into one DDF $\mathscr{F}_{trg}$ using the method introduced later in Section 3.2.

2. Sample a Deformation Graph from the reference surface, and estimate its parameters with the nonrigid alignment algorithm introduced in Section 2 to align the reference surface with the new observation ($\mathscr{F}_{trg}$ and color images). The parameters for this forward deformation is computed.

3. Compute the backward deformation (from target to reference) according to the forward deformation, and transform the fused DDF from step 1 to the reference, i.e., $\mathscr{F}_{trg} \rightarrow \mathscr{F}_{ref}$. Details are provided in Section 3.1.

4. Fuse $\mathscr{F}_{ref}$ into the Multi-Mode Directional Distance Function (MDDF) at reference. Unlike DDF, MDDF has multiple distance values and direction vectors at each voxel. A detailed introduction of MDDF and fusion of multiple DDFs is presented in Section 3.2.

5. Finally, generate the reference surface from the MDDF and texture it by all color images observed so far. To texture the surface, we deform the surface to various frames according to the estimated forward deformations and project each vertex to the image spaces, the color pixels the vertex falls on are averaged to obtain one 3D color vector.

The scanning procedures are summarized in Algorithm 1, and a graphical illustration is given in Figure 2. Note that we always align the fused data to following frames, which helps to deal with the error accumulation problem [21]. Also note that we do not directly estimate the backward deformation parameters by deforming the target to the reference. This is because the generated deformation graph on the noisy target surface tends to be problematic.

Algorithm 2: DDF transformation $\mathscr{F}_{trg} \rightarrow \mathscr{F}_{ref}$

foreach *i-th voxel of* $\mathscr{F}_{trg}$ *at location* p_i *with direction to nearest surface point denoted as* P_i *and distance value as* D_i **do**

- **1.** deform its location according to Eq. 1: $p_i \rightarrow \tilde{p}_i$;
- **2.** deform its direction P_i according to Eq. 2: $P_i \rightarrow \tilde{P}_i$;
- **3.** record the deformed voxel as a 4-tuple $\langle p_i, \tilde{p}_i, \tilde{P}_i, D_i \rangle$;

foreach *each voxel of* $\mathscr{F}_{ref}$ *at location* q **do**

- **1.** find the set of its neighboring deformed $\mathscr{F}_{trg}$ voxel: $\mathbb{S} = \{\langle p_k, \tilde{p}_k, \tilde{P}_k, D_k \rangle | _{\|\tilde{p}_k - q\| < \varepsilon}\}$;
- **2.** Divide $\mathbb{S}$ in to subgroup $\{\mathbb{G}_i\}$ by clustering on p;
- **3.** Find the subgroup $\mathbb{G}_s$ with smallest averaged D;
- **4.** set the direction and distance value of $\mathscr{F}_{ref}$ at q:

$$P^{ref} = \sum_{k \in \mathbb{G}_s} w_k \tilde{P}_k$$
$$D^{ref} = \sum_{k \in \mathbb{G}_s} w_k D_k$$
$$\text{where } w_k = exp(-(q - \tilde{p}_k)^2$$

3.1 DDF Transformation from Target to Reference

Given the forward deformation parameters $\{\langle A_j, t_j \rangle\}$, it is tempting to set the backward deformation parameters as $\{\langle A_j^{-1}, -t_j \rangle\}$ and the graph node position as $g_j + t_j$; however it does not guarantee a close backward alignment, since the inverse of the linear interpolation of affine matrices does not equal the linear interpolation of the inverse of affine matrices. Instead, we find the point correspondence of the reference and target according to the forward deformation, and estimate the backward deformation parameters via Eq. 3 by formulating the point correspondence into E_{con}.

With the backward deformation estimated, one way of deforming the DDF is first generating the underlying triangulated surface from the DDF, then deforming the surface, and eventually recomputing the new DDF from the deformed surface. This method seems plausible, but the DDF field structure is not preserved dur-

Figure 3: Scanned Surfaces Generated by Our System.

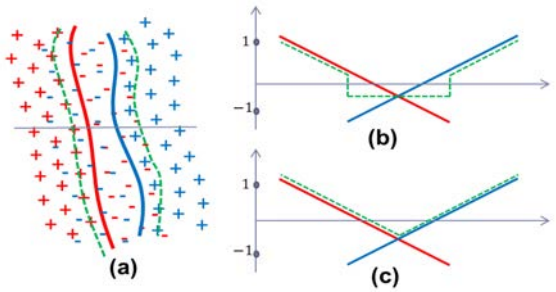


Figure 4: Fusing two 2D truncated signed distance fields. (a) shows two curves and their signed distance fields in red and blue respectively. Directly adding two signed distance fields ends up with expanded curves shown in green. (b) shows the signed two distance functions along one line across two curves in red and blue, and the sum of the two in green. (c) shows the above expansion effect is prevent by taking the one with smaller absolute value.

ing transformation due to downgrading the DDF to a surface. We choose to directly apply deformation on DDF. Although the non-rigid transformation is only defined on the surface, each voxel of the DDF could be transformed according to the deformation parameters of its closest point on the surface. Alg. 2 shows our solution of deforming DDF $\mathscr{F}_{trg}$ to $\mathscr{F}_{ref}$ according to the deformation graph $\mathbb{G}$. Note that we need to handle the situation when the transformed voxels collide with each other. At each voxel position on $\mathscr{F}_{ref}$, we find its nearby transformed $\mathscr{F}_{trg}$ voxels, then group them according to their original grid positions, and find the group with smaller absolute distance value, from which the direction vector and signed distance value for the $\mathscr{F}_{ref}$ voxel are interpolated.

3.2 Fusion of multiple DDFs

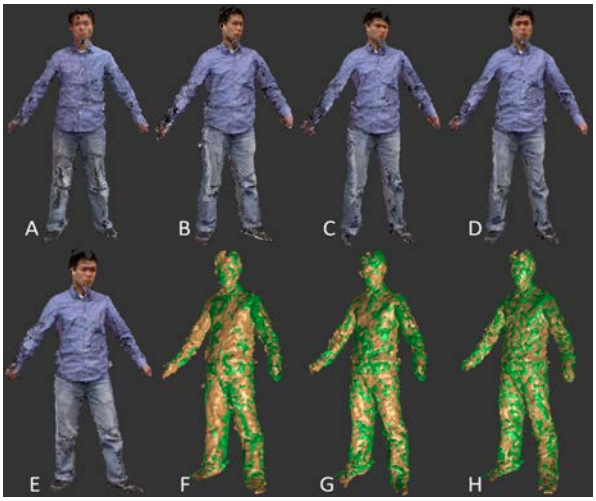
When the noise level of the 3D data is low, summing over multiple aligned DDFs cancels out the noise, as shown in the work of KinectFusion [21]. Then the surface can be recovered by finding the zero-crossing of the fused distance field using algorithms such as Marching Cubes. However, when the noise level is as large as the object dimension, summing over the distance field raises problems as illustrated in Fig. 4(a)&(b). This is because the distance field needs to be truncated so that the distance field of a front surface does not interference with the surface behind. The distance behind the surface where truncation begins, denoted as μ, should be positively relevant to the noise level. Unfortunately, in the case of Fig. 4(a), when a big μ is chosen to suppress the noise, the zero-crossing of the fused distance field does not align with the surface anymore due to the interference between the distance functions of the front and back surface. This is exactly the case when performing dynamic object scanning via commodity depth cameras, since the noise coming from the depth camera, calibration error and non-rigid alignment can easily go beyond the dimension of thinnest part of the object, such as palm or wrist.

Fortunately, in many cases, $\nabla\mathscr{D}$ differentiates which surface a distance value corresponds to. And in our DDF representation, $\nabla\mathscr{D}$ can be easily obtained using Eq. 9. When fusing DDFs, at each voxel, we only sum over the distance value $\mathscr{D}$ with similar $\nabla\mathscr{D}$, preventing the interference between the distance fields corresponding to different surface parts. This results in a new data structure: Multi-Mode Directional Distance Function (MDDF). Each voxel of a MDDF records a set of averaged $\mathscr{D}$'s, $\nabla\mathscr{D}$'s, and the weights on all modes. In our scanning Step 4, a new $\mathscr{F}_{ref}$ can be easily fused to MDDF. First, the mode with most similar $\nabla\mathscr{D}$ is found at each voxel; then its distance value and $\nabla\mathscr{D}$ is incorporated to that mode and the weights are updated.

To recover the underlying surface from a MDDF, it needs to be downgraded to a DDF. One mode is selected among all the modes at a MDDF voxel as the DDF voxel. We choose the one with smaller absolute distance value. As illustrated in Fig. 4(c), this strategy solves the interference problem shown in Fig. 4(b). In practice, the mode with relatively smaller weights (50% of the largest weight in our experiment) are discarded when downgrading.

4 Tracking

After fusing a number of DDFs (a few hundreds in our case), the improvement on a scanned model tends to converge. Thus, after a complete model is achieved, we deform the scanned model to the



Reference and Target Sumner et al. [24] Li et al. [17] Ours

Figure 5: Comparison of various algorithms aligning surface (E) to surface (A). (B)(C)(D) show the deformed surfaces of (E) with three algorithms respectively; (F)(G)(H) show two surfaces after alignment where the reference is in green and the target in orange.

current depth and color observations, i.e., only Steps 1 and 2 in Algorithm 1 are used during this stage—tracking. Note that a fixed Deformation Graph structure is used in Step 2 during tracking.

To track fast moving surfaces, we use a Kalman Filter to predict the translation vector t_j for each deformation graph node of the next frame , and use its prediction as the initial parameter of the non-rigid alignment problem. The affine matrices $\{A_j\}$ are simply initialized using the values of the last frame.

4.1 Tracking Surfaces with Isometric deformations

In many cases, the surface being tracked is roughly under isometric deformations [8], i.e., the geodesic distance of any pair of surface points is preserved during deformation. For example, the deformation of the 3D human body model is near isometric, if not strictly isometric. Thus, for these cases, we add a new term E_{len} to our energy minimization problem in Eq. 7,

$$E_{len} = \sum_{j=1}^{J} \sum_{k \in \mathbb{N}(j)} \left\| |g_j + t_j - g_k - t_k| - |g_j - g_k| \right\|_2^2. \qquad (11)$$

where g_j and t_j are the node location and translation vector of the Deformation Graph respectively, and $\mathbb{N}(j)$ are the neighbors of the j-th node. E_{len} penalizes the changes of the length of the edge connecting the neighboring nodes during deformation. Although E_{len} does not guarantee an exact isometric deformation, in practice it works well to prevent the surface from stretching or shrinking. In our implementation, we use the robust estimation technique [1] on E_{len} to allow for length changes for some parts (outliers).

5 Experimental Results

To test our system, we captured several sequences of people performing various movements using the eight Kinects setup shown in Fig. 2(A). Both depth maps and color images from Kinects are used for nonrigid matching, and both have a resolution of 640×480. Since the depth coming from Kinects deviates from the true depth value [6][20], we correct this depth bias using a linear mapping function for each Kinect separately. The synchronization across Kinects is still an unsolved problem, but the current setup works reasonably well even for fast movement.

The resolution of the lattice of the volumetric representation (DDF) is set to 1cm in our experiment to reduce the processing time and memory, yet it is still enough to output relatively high quality models. The DDF with higher resolution is necessary to achieve the quality of a high-grade commercial laser scanner. In all of our experiments, μ in DDF is set to 4cm; the weights in Eq. 7 are set as follows, $w_{rot} = 30$, $w_{reg} = 5.0$, $w_{con} = 1.0$, $w_{dns\_pts} = 5.0$, $w_{clr} = 5.0$, and $w_{len} = 3.0$. These parameters are chosen experimentally. Both the color intensity and signed distance value have been scaled to $[0,1]$. The Google Ceres solver generally converges after 10 iterations on Eq. 7. The Deformation Graph Model is uniformly sampled on the surface based on the geodesic distance. The minimum distance between neighboring nodes is set to 7cm, and about 350 graph nodes are sampled. A denser graph model does not show visually improved results.

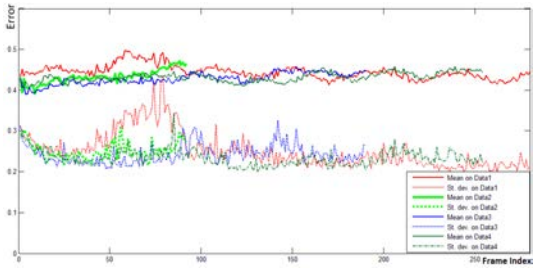


Figure 7: Scanning Errors (in cm). The solid lines indicate the mean deviation of the fused surface to the observed surface at each frame on four data sequence. The dashed line shows the standard deviation of the error.

During scanning or tracking, around 150 pairs (per image pair) of matched corner points from the current and previous frame are found via LK optical flow. The points of the previous frame are transformed to the reference via the backward deformation. Note that E_{dns_pts} and E_{clr} only work when the initialization is reasonably close to the optimal solution. Thus, when the object moves fast and the initial is far off the optimal, E_{con} plays an important role. Otherwise, it is overwhelmed by E_{clr}—dense color matches. In our experiment, we ignore E_{con} in our tracking state to save us from computing the backward deformation and use the Kalman Filter to predict a reasonable initial value.

5.1 Non-rigid Alignment and Comparison

As shown in Fig. 1(b) and (c), our nonrigid alignment algorithm works well to register a sequence of surfaces together using the scanning pipeline shown in Algorithm 1. We compare our algorithm with previous works from Sumner el al. [24] (designed for shape editing and used by others for non-rigid alignment) and Li et al. [17] (the detail synthesis step in Li's work is not included for comparison). Figure 5 shows the results of aligning one frame to the reference. A comparison on the whole sequence is presented in the supplemental video. Sumner et al. only uses matched sparse points for alignment (E_{con} in Eq. 7), so the dense points are not perfectly aligned. Li's algorithm performs better since dense points is employed for alignment; as shown in Figure 5(G), the reference and target surfaces after alignment are well interleaved compared with (F). But the textures of aligned surfaces do not match for both algorithms; in Figure 5, neither (B) nor (C) is close to (A). Since our algorithm includes a color consistency constraint, it performs better than the other algorithms. Not only do the surfaces align more tightly, the textures match as well.

5.2 Scanning

Some scanned 3D models of full human bodies are shown in Fig. 3. Despite the low DDF resolution, our algorithm still recovers the geometry details such as wrinkles on the clothes. Fig. 3 also shows the color on the vertices of the model averaged from all the frames used for fusing (200~300 frames). The sharpness of the color indicates

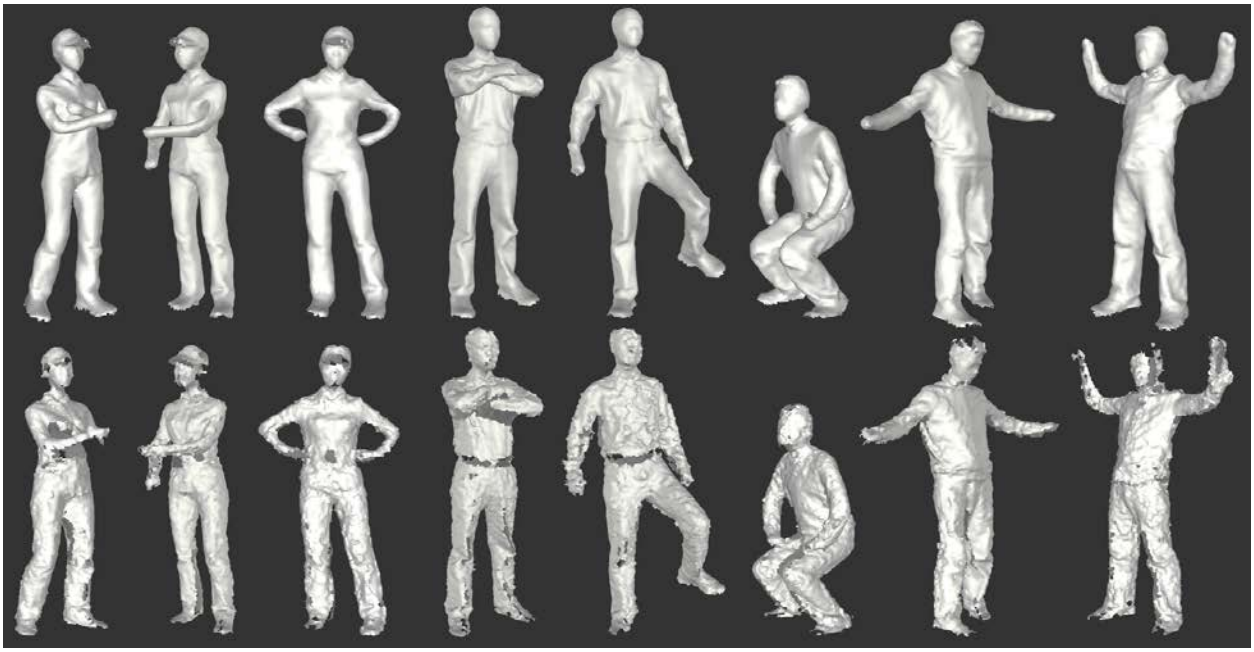


Figure 6: Surface Tracking. The top row shows the deformed human scan; the bottom row shows the input surfaces from Kinects.

that the frames are well aligned. Be aware that the sparseness of the model vertices (due to the low DDF resolution) leads to some blur from the color interpolation during GPU rasterization. A few intermediate models created during scanning are shown in Fig. 8. The noise is gradually filtered out and holes filled up while the geometry details are preserved when accumulating more and more frames.

We evaluated the scanning system quantitatively. The deviation of the scanned model from the observed surfaces at each frame is shown in Fig. 7. To measure the deviation, the observed surfaces are deformed to the reference as shown in Fig. 1(b). The averaged distance between matched points is used as an measurement of the surface deviation. As shown in Fig. 7, the deviation stays around 0.42 cm for all frames of all test data sequences, indicating our nonrigid alignment algorithm works decently and the scanning system handles the error accumulation problem well.

5.3 Tracking

We test our tracking algorithm on multiple sequences with people performing various movements. Fig. 6 shows some results of tracking the scanned surface. The robust nonrigid matching algorithm enables our system to track difficult gestures involving significant topology changes such as arm folding or fast surface deformation such as squatting and stretching. More tracking results and comparisons are presented in the supplemental video.

One application of our system is the novel view generation. With the holes being filled up and noise being filtered out, the rendering result is visually more appealing as shown in Figure 9(B) and (D).

5.4 Limitations of the System

Segmentation. Currently, to segment the foreground dynamic objects from the rest of the scene on the depth map, we first eliminate the floor area via a clipping plane and other background outside a bounding box, then perform connected component labeling to filter out small noisy blobs. The depth-only segmentation works fine on the human body except the feet. The problems on the feet are visible in the scanned models in Fig. 3. An improved segmentation might employ color information to refine the boundary.

Topology Change. During the scanning stage, we ask the people being scanned to turn around, performing any movement they want except those changing the body topology such as crossing the arms. Our scanning algorithm assumes the body topology does not change. This requirement could be removed by identifying the body parts with topology changes in each frame and only fusing the body part without topology changes. The tracking stage remains free of these restrictions; we make no assumptions on body topology then.

Processing Time. Most of the system implementation is single-threaded and performed on CPU. As a result, it takes approximately one minute per frame to fuse or track one frame containing approximately 30,000 vertices. Most of the computation time is dedicated to solving Eq. 7.

6 Conclusions

Our work shows temporal information of depth and color is helpful for modeling dynamic objects. We demonstrate a complete and accurate surface is fused from a sequence of depth and color data from commodity depth and color cameras. By tracking this fused surface over time, we acquire an improved 3D model for later frames. The key components of our system are a nonrigid alignment algorithm that integrates both depth and color information and a volumetric fusion algorithm that handles large noise on depth observations.

Acknowledgements

We would like to thank the anonymous reviewers for constructive comments and suggestions, and Peter Lincoln for reviewing and editing the paper. This work was supported in part by CISCO Systems and by the BeingThere Centre, a collaboration between UNC Chapel Hill, ETH Zurich, and NTU Singapore, supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the Interactive Digital Media Programme Office.

References

[1] S. Agarwal and K. Mierle. *Ceres Solver: Tutorial & Reference*. Google Inc.

[2] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. Scape: shape completion and animation of people. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 408–416, 2005.

[3] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 2004.

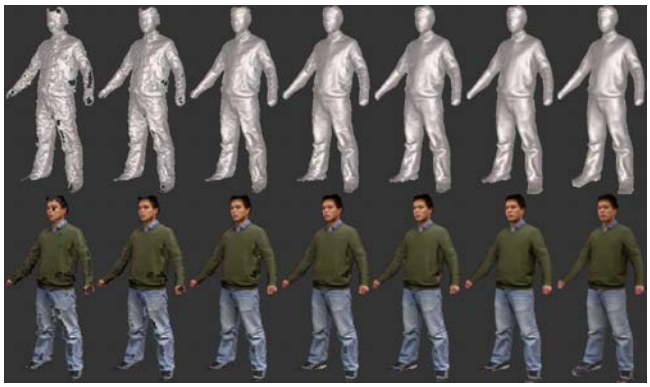


Figure 8: Intermediate Scans at frame 1, 5, 15, 30, 50, 80, and 150.

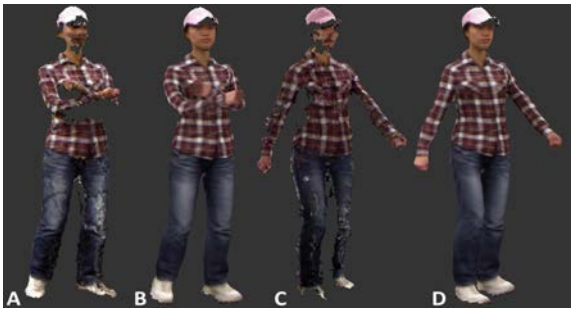


Figure 9: Novel View Generation from the output surfaces of our system (B and D) and original surface (A and C).

[4] L. Ballan and G. M. Cortelazzo. Marker-less motion capture of skinned models in a four camera set-up using optical flow and silhouettes. In *3DPVT*, Atlanta, GA, USA, June 2008.

[5] I. Baran and J. Popović. Automatic rigging and animation of 3d characters. In *ACM Transactions on Graphics (TOG)*, 2007.

[6] S. Beck, A. Kunert, A. Kulik, and B. Froehlich. Immersive group-to-group telepresence. In *IEEE. VR*, 2013.

[7] M. Bojsen-Hansen, H. Li, and C. Wojtan. Tracking surfaces with evolving topology. *ACM Transactions on Graphics (Proceedings SIGGRAPH 2012)*, 31(4), 2011.

[8] A. M. Bronstein, M. M. Bronstein, and R. Kimmel. Generalized multidimensional scaling: a framework for isometry-invariant partial surface matching. *Proc. National Academy of Sciences (PNAS)*, 2006.

[9] J. Chen, S. Izadi, and A. Fitzgibbon. Kinêtre: animating the world with the human body. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, 2012.

[10] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH*, 1996.

[11] E. De Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun. Performance capture from sparse multi-view video. In *ACM Transactions on Graphics (TOG)*, volume 27, page 98. ACM, 2008.

[12] A. W. Fitzgibbon. Robust registration of 2d and 3d point sets. *Image and Vision Computing*, 31, 2003.

[13] J. Gall, C. Stoll, E. De Aguiar, C. Theobalt, B. Rosenhahn, and H.-P. Seidel. Motion capture using joint skeleton tracking and surface estimation. In *IEEE Conf. CVPR, 2009*.

[14] E. Herbst, X. Ren, and D. Fox. Rgb-d flow: Dense 3-d motion estimation using color and depth. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[15] D. Hirshberg, M. Loper, E. Rachlin, and M. Black. Coregistration: Simultaneous alignment and modeling of articulated 3D shape. In A. F. et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, LNCS 7577, Part IV, pages 242–255. Springer-Verlag, Oct. 2012.

[16] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, 2011.

[17] H. Li, B. Adams, L. J. Guibas, and M. Pauly. Robust single-view geometry and motion reconstruction. *ACM Transactions on Graphics (Proceedings SIGGRAPH Asia 2009)*, 28(5), December 2009.

[18] H. Li, L. Luo, D. Vlasic, P. Peers, J. Popović, M. Pauly, and S. Rusinkiewicz. Temporally coherent completion of dynamic shapes. *ACM Transactions on Graphics*, 31(1), January 2012.

[19] K. Madsen, H. Nielsen, and O. Tingleff. Methods for non-linear least squares problems. *Lecture Note*, 2004.

[20] A. Maimone and H. Fuchs. Encumbrance-free telepresence system with real-time 3d capture and display using commodity depth cameras. In *ISMAR*, 2011.

[21] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. *IEEE ISMAR*, 2011.

[22] J. Pokrass, A. M. Bronstein, M. M. Bronstein, P. Sprechmann, and G. Sapiro. Sparse modeling of intrinsic correspondences. In *Eurographics Computer Graphics Forum (EUROGRAPHICS)*, 2013.

[23] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proceedings of the EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing*, pages 179–188. ACM Press, 2004.

[24] R. W. Sumner, J. Schmid, and M. Pauly. Embedded deformation for shape manipulation. In *SIGGRAPH*, 2007.

[25] J. Tong, J. Zhou, L. Liu, Z. Pan, and H. Yan. Scanning 3d full human bodies using kinects. *IEEE Trans. on Visualization and Computer Graphics*, 18(4), 2012.

[26] D. Vlasic, I. Baran, W. Matusik, and J. Popović. Articulated mesh animation from multi-view silhouettes. In *ACM Transactions on Graphics (TOG)*, volume 27, page 97. ACM, 2008.

[27] A. Weiss, D. A. Hirshberg, and M. J. Black. Home 3d body scans from noisy image and range data. In *Int. Conf. on Computer Vision (ICCV)*, 2011.

A DDF Calculation pseudo-code

Input: the surface points $\{v_i\}_{i=0}^{M}$; the DDF grid dimension $N_x \times N_y \times N_z$; the DDF thickness μ.

```
1  d_xyz ← ∞; //signed distance field
2  Ind_xyz ← −1; //index to closest surface point
3  p⃗_xyz ← 0⃗ //directional field
4  //loop over M surface points
5  for(unsigned int i=0; i<M; i++) {
6    //loop over grid points within the bounding
7    //box (x1−x2, y1−y2, z1−z2) centered at v_i
8    //with radius of μ
9    for(unsigned int x=x1; x<=x2; x++)
10     for(unsigned int y=y1; y<=y2; y++)
11       for(unsigned int z=z1; z<=z2; z++){
12         d_new = distance to v_i
13         if( d_new < d_xyz ){
14           d_xyz = d_new;
15           Ind_xyz = i;
16         }
17       }
18 }
19 for(unsigned int x=0; x<N_x; x++)
20   for(unsigned int y=0; y<N_y; y++)
21     for(unsigned int z=0; z<N_z; z++){
22       if(Ind_xyz is surface boundary vertex)
23         continue;
24       //v⃗ is the closest surface point
25       //and n⃗ is its outward normal
26       //g⃗ is the current grid point
27       p⃗_xyz = v⃗ − g⃗;
28       d_xyz = (p⃗_xyz · n⃗ < 0) ? d_xyz : −d_xyz;
29       d_xyz = max(−1.0, min(1.0, d_xyz/μ));
30     }
```