Markerless Motion Capture with Multi-view Structured Light

Ricardo R. Garcia, *Student Member, IEEE,* and Avideh Zakhor, *Fellow, IEEE*

Abstract—We present a multi-view structured light system for markerless motion capture of human subjects. In contrast to existing approaches that use multiple camera streams, we reconstruct the scene by combining six partial 3D scans generated from three structured light stations surrounding the subject and operating in a round robin fashion. We avoid interference between multiple projectors through time multiplexing and synchronization across all cameras and projectors. Assuming known multi-camera projector calibration parameters, we generate point clouds from each station, convert them to partial surfaces, and merge them into a single coordinate frame. We develop algorithms to reconstruct dynamic geometry using a template generated by the system itself. Specifically, we deform the template to each frame of the captured geometry by iteratively aligning each bone of the skeleton. This is done by searching for correspondences between the source template and the captured geometry, solving for rotations of bones, and enforcing constraints on each rotation to prevent the template from taking on anatomically unnatural poses. Once the sequence of captured geometry is processed, the template is textured using color images from the multi-view structured light systems. We show the effectiveness of our system for a 50-second sequence of a moving human subject.

Index Terms—structured light, motion capture, multi-view reconstruction.

1 Introduction

Human motion capture has been under investigation for many years in different applications [1], [2]. While marker based motion capture has been popular for a while, it is desirable to do away with markers, simplify the capture process, and improve realism by capturing actual subject color and shape. Even though a marker based system with many markers can be used to increase the realism of results, it comes at a significant setup cost where hundreds of markers must be placed directly on the body of the human subject [3].

In many systems, the geometry of the dynamic human subject is captured using multi-camera setups [4], [5], [6], [7], [8], [9]. These systems recover a representation of the human subject by using visual hulls generated from each camera view. Many approaches use a template of the human model to assist in the reconstruction of the dynamic geometry [4], [6], [7], [8], [9], [10], [11], [12]. The methods using templates can either use a skeleton [4], [7], [8], [9], [10], [11] or not [6], [12]. In [8], Vlasic et al. generate a detailed template using a high quality laser scanner. The template is then deformed according to constraints gathered from the observing cameras. De Aguiar et al. similarly present a template based method that starts with a high quality laser scan, but do not use a skeleton to deform the mesh [6]. Rather, mesh based deformation techniques such as those in [13] are used to deform the mesh according to the visual hulls of the observing cameras. The method in [8] fits a template via a combination of coarse movements from skeleton deformation, followed by local mesh deformation to capture the details of the scan. Gall et al. use a similar approach, but do not require manual user intervention [9]. Methods using highly detailed templates are sometimes criticized for "baking in" details that should not be present throughout the reconstructed sequence.

Besides multi-camera methods emphasizing the use of visual hulls, other capture methods have been proposed. In [14], dynamic scans from a photometric stereo system are used for surface reconstruction. While the geometry captured from this system is quite accurate and detailed, the system is complex and expensive. Specifically, it uses a geodesic sphere 8 meters in diameter and with over 1200 individually controllable light sources, making it inaccessible to most users. Unlike methods involving visual hulls from multiple-cameras, in [14] the 3D shape of the human subject's surface is directly captured. Such systems directly reconstruct the subject's surface without any prior information rather than using a template to estimate the pose and shape of a human subject. For instance, the approach in [14], [15] does not use templates to reconstruct water tight meshes. The results from these direct capture approaches often yield errors in the topology of the reconstructed geometry since there is no prior information on the shape of the model. Similar works have focused on the problem of registering and generating consistent geometry from sets of dynamic point clouds [16], [17], [18]. These methods only work with high temporal sampling and limited occluding geometry.

In this paper, we present a markerless motion capture system consisting of multiple structured-light subsystems to capture geometry of a human subject from surrounding views. As with any multi-view system, since the entire shape of the human subject is not always visible from any one capture device, we opt to use a template to represent the shape and pose of the human subject. However, unlike existing approaches which require additional scanning hardware or complex template initialization processes, we use the same motion capture system to generate the template. Specifically, from an initial scan of the human subject in a occlusion free pose, we create a customized template for that subject as shown in Fig. 1b. In contrast to existing methods that use 2D images, we capture 3D geometry using three structured-

• *R. R. Garcia and A. Zakhor are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, 94706.*
E-mail: rrgarcia@eecs.berkeley.edu and avz@eecs.berkeley.edu

Manuscript received January 16, 2015.

(a) (b) (c)

Fig. 1: Using (a) captured partial scans and (b) a template of a human subject, we generate (c) the time varying geometry of a human subject.

light systems. Specifically, with two cameras per structured-light system as shown in Fig. 2, we generate 3D scans from six views as shown in Fig. 1a, deform the template, shown in Fig. 1b, and fit the target scan as shown in Fig. 1c. The problem of fitting a template to the human subject is greatly simplified if constrained by 360-degree geometry from three stations.

There are existing methods for reconstructing dynamic geometry from 3D geometry scans. Dou et al. [12] use eight Kinects surrounding a human subject to build a model of a dynamic figure. Due to sensor noise, they cannot directly create a template in a single frame; rather, they learn it over time. Our system is most similar to [19] which captures two opposing views of the human subject and stitches the views together, leaving significant artifacts around the seams.

The outline of this paper is as follows. In Section 2, we present the specifics of the system architecture and hardware used to capture 3D partial reconstructions of the human subject. Section 3 describes how we generate and deform the template. In Section 4, we explain the sequential deformation method used to fit a template to partial geometry scans. Section 5 describes texture mapping for the template. Sections 6 and 7 present an example of processing real captured human motion and conclusions, respectively.

2 System Setup and Data Capture

Our proposed system consists of three distinct structured-light stations, shown in Fig. 2, surrounding a central capture volume with the devices located on each station pointing towards the center of the capture volume. Each station is equipped with one DLP® projector, two grayscale cameras, and one color camera in a configuration similar to [20]. To prevent interference between the three projectors at each structured-light station, we apply structured-light patterns to one projector at a time in a round-robin fashion. In other words, each station takes turns illuminating the human subject in a time-multiplexed sequence. The system is controlled by two separate desktop PCs, one for driving structured-light patterns to the three projectors, and a second one for streaming in images from all nine cameras. A microcontroller is used to extract timing signals from the projectors to generate trigger signals for all of the cameras. An external PCIe expansion chassis

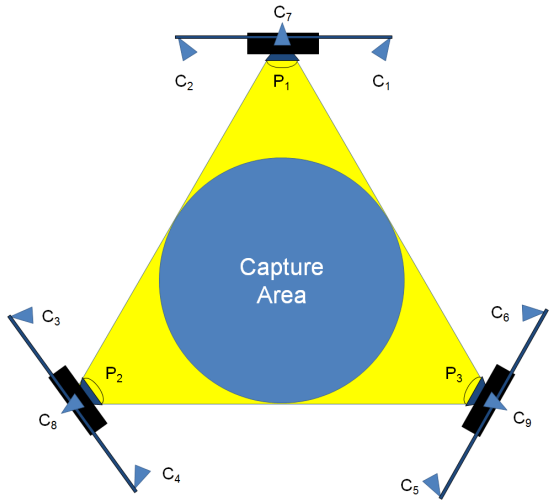


Fig. 2: Top-down view of the three structured-light stations in the system. Geometry cameras are labeled C_1-C_6. Color cameras are C_7-C_9. Projectors are P_1-P_3.

is added to the camera PC to provide enough Firewire cards to connect all nine cameras to the single PC.

The resolution of the projectors is 1024×768, and the resolution of the cameras is 640×480. At each station, the two grayscale cameras are used with the local projector to capture geometry, and the color camera is used to capture texture. As described in [21], the projectors are modified to only project in grayscale by removing the color wheels. DLP® projectors generate color images by sequentially projecting the individual red, green, and blue color channels of a video source. The removal of the color wheel effectively modifies the projector from a 60 Hz color projector to a 180 Hz grayscale projector. Similar to [21], we use three phase-shifted sinusoidal patterns to reconstruct depth. Each pattern is stored into one of the three separate color channels of an image and sent to the projector. The three sinusoidal patterns are individually captured as they illuminate the human subject and the phase of each pixel in the camera is computed [21]. While the phase of a camera pixel determines which points in the projector potentially illuminate it, due to the periodic nature of the sinusoidal patterns, correspondences between the cameras and

projectors are not uniquely known by the phase alone. By using the temporal phase unwrapping method from [22], the phase images can be unwrapped over time and space to ensure spatiotemporally consistent point clouds at each station. Specifically, we combine a quality guided phase unwrapping approach with absolute phase estimates from the stereo cameras to solve for the absolute phase of connected regions. Temporal unwrapping with stereo information significantly improves the consistency of unwrapped results, which in turn leads to consistency of the point clouds [22].

If multiple projectors simultaneously illuminate the same point on the human subject, neither the phases of projectors can be accurately estimated, nor can the correspondence between camera and projector pixels. Since the projectors natively operate at a 60 Hz rate, the time-multiplexing between the three stations lowers the capture rate from each station to 20 frames per second. Clearly faster projectors allow for capturing human subjects with faster motion. To implement this interleaving process, the video signals sent to each projector are all synchronized to each other. Specifically, we use two NVIDIA® Quadro® 5000 graphics cards with an additional G-Sync® daughter card in the display PC to synchronize all video source signals. Triggering signals sent to the capture cameras are derived from the timing of the synchronized projectors. All timing signals are generated using an external microcontroller. Similar to [21], the grayscale cameras capture the individual structured-light patterns at approximately 180 Hz, and the color cameras capture at 60 Hz. The exposure of the color cameras is chosen to last for the entire duration of projection of the three sinusoidal patterns; thus the integration of the three phase-shifted sinusoidal patterns appears as a constant illumination to the color cameras.

Each grayscale camera projector pair is capable of capturing a portion of the human subject's geometry from its own viewpoint. Since our system is designed to capture the full 360-degree view of the human subject, the individual views need to be aligned into a a single coordinate frame, thus requiring an accurate calibration of all cameras and projectors. We use the multi-camera-projector calibration approach in [23] to accomplish this. We use a translucent target placed at various positions within the center of the capture volume as a calibration target. Each projector projects striped binary patterns in the vertical and horizontal directions. The observing cameras in the system can decode the striped binary patterns to determine pixel-level correspondences between all devices. The correspondences along with estimates of the 3D locations of the correspondence points are fed into a bundle adjustment optimization method to calibrate all the devices.

Even though the evolving pose of the human subject can be observed frame by frame after point cloud alignment, the limited number of capture devices and stations surrounding the human subject causes holes during reconstruction. This is especially true for regions that are not illuminated by the projector, such as tops of heads and regions that are occluded by other parts of the body, such as a portion of a torso occluded by an arm. Additionally, regions with low reflectivity and regions with shadows also create holes. Fig. 3 illustrates a single frame captured by each of the six grayscale cameras of the system in Fig. 2. There are several holes in each of the views which need to be filled out in the fused 3D model. This motivates the use of templates in our system.

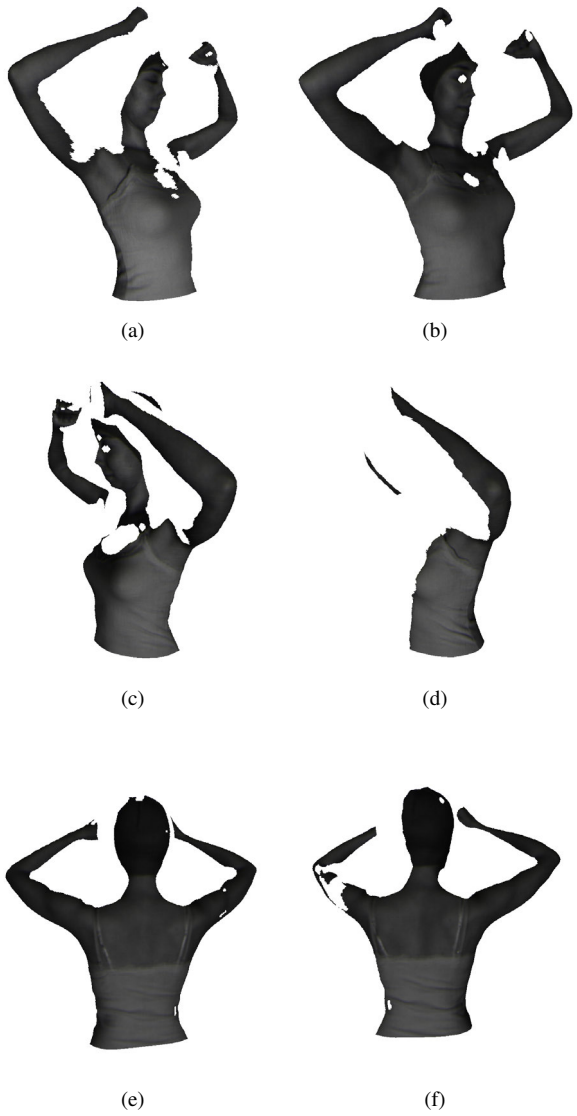
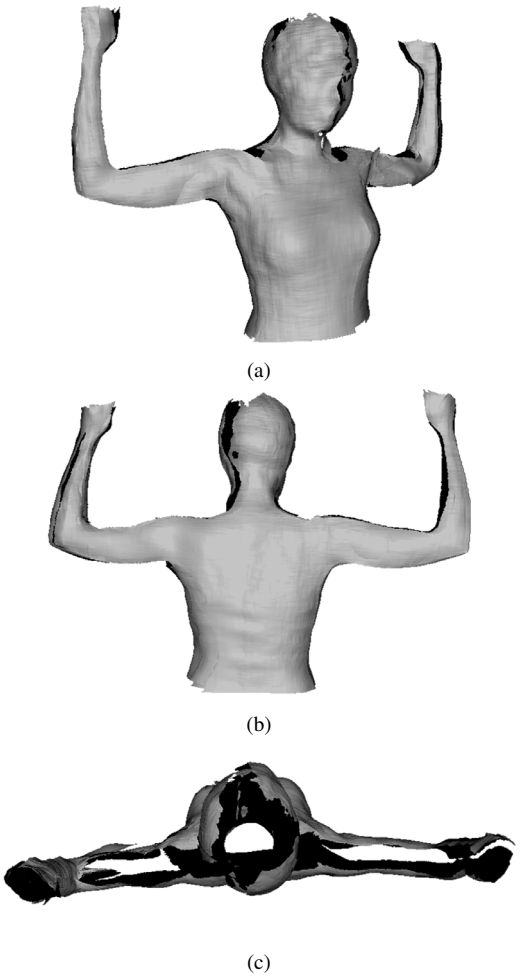


Fig. 3: Partial meshed point clouds of the human subject generated from unwrapped phase images of a single frame. Views captured from each of the six grayscale cameras in the system: (a) C_1, (b) C_2, (c) C_3, (d) C_4, (e) C_5, and (f) C_6 as shown in Fig 2.

3 TEMPLATE

Even though visual hulls can be used to estimate geometry for many poses of the human subject, for scenes with significant self-occlusions, it is not always possible to generate a watertight representation of the entire subject's surface. As such, *a priori* information on the structure of the human subject can help to properly fill in these regions. In this paper, we use a template to create a complete representation of each frame. However, unlike existing methods [6], we use the same system configuration that captures the dynamic geometry to also capture the template, obviating the need for a separate system such as a high-quality laser scanner.

To create a template, the human subject is captured in a pose where the majority of the his or her surface can be observed by the cameras and projectors so as to limit the self-occlusions. Fig. 4 shows the captured geometry with the human subject in the template pose. As seen, the majority of the geometry is captured, except for regions on the tops and bottoms of the arms, the top of the head, and the bottom of the torso, which are not observable by any of the capture devices. These regions are not illuminated



(a) (b) (c)

Fig. 4: Partial geometry used to generate a template. Views from (a) front, (b) back, and (c) top.

by the projector, and therefore their 3D shape cannot be captured. We use Poisson surface reconstruction [24] to fit a smooth meshed point cloud over the existing geometry. As seen in Fig. 5, this generates a hole-free representative template of the human subject.

Even though the template provides an *a priori* reference of the true shape of the human subject, as a mesh alone, it lacks the structure to effectively fit the fused captured geometry from the three stations. In poses with significant occlusions, a reference to the overall structure of the template is useful to prevent unnatural mesh deformations. This is especially true when little geometry for the human subject's arms is captured as they pass in front of the body. In these poses, often only the outside of the arm is visible to the capturing devices. The lack of geometry may cause bending in rigid regions of the arm since there are limited correspondence points to enforce a natural deformation. As is done in traditional motion capture, we choose to fit a skeleton rig to the template to control the way its geometry is deformed over time [1].

The skeleton rigging, or skeleton, is a set of artificial bones set within the template to emulate natural pose deformations of the template mesh. The position and movement of the bones control the movement of the vertices in the template mesh. For our system, which only captures a waist-up view of the human subject, we

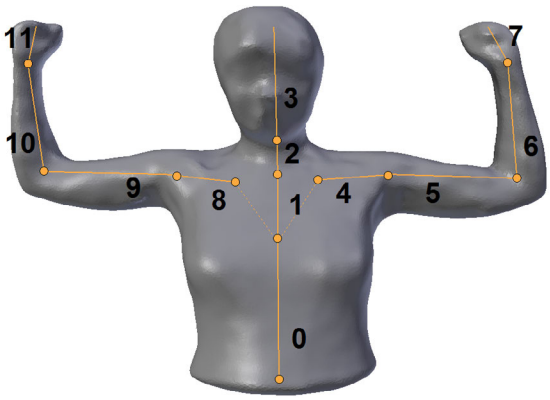


Fig. 5: Rigged template of a human subject generated by Poisson surface reconstruction. The bones and joints are numbered and indicated in orange.

opt to use 12 bones within our skeleton, as shown in Fig. 5. [1] Each bone in the skeleton has a single lead joint that controls its movement and position. The joint-bone pairs are labeled in Fig. 5. The position of each vertex in the template mesh is determined by the positions and orientations of nearby bones. The influence of each bone on each vertex's position is specified by the bone weight map matrix $\mathbf{W}$ with dimensions of $[N \times J]$, where N is the number of vertices in the template mesh and J is the number of bones in the template. Each entry in the matrix $\mathbf{W}(i, j)$ takes on a real value from zero to one, specifying the influence of each bone j on the position of each vertex i by taking into account a vertex's relative distance to each bone. The vertices are influenced by one to three bones. Each row of influence from all bones to a single vertex is normalized to sum to one.

We deform the template by using dual quaternion skinning [25] to modify the skeleton pose so as to fit the fused geometry from three stations. In contrast to the commonly used linear blend skinning, dual quaternion skinning provides more natural deformation, especially under large twisting motions. We solve for the bone positions that deform our mesh to best align the template to the fused surfaces captured in each frame.

As is standard with skeleton-rigged deformation, rotating a branch around a joint not only rotates the connected bone, but also all subsequently connected bones. Specifically, the skeleton of our template is connected in a tree structure with joints serving as the nodes of the tree and the bones as the paths between nodes. We can think of the set of bones that follow from a joint in the skeleton to a terminal bone in the tree as a branch. The largest branch starts from the joint for bone 0 in Fig. 5 and includes all bones in the skeleton. The smallest branches are single bones 3, 7, and 11 as shown in Fig. 5. Anytime a bone is transformed around its joint, the subsequent bones along the branch need to be updated as well. For example, in Fig. 5, if bone 4 is rotated, then bones 5, 6, and 7 are moved to keep their pose relative to bone 4. We refer to the branch that starts at a bone j as branch j. A branch weight map $\bar{\mathbf{W}}$ can be calculated from the bone weight map:

$$\bar{\mathbf{W}}(i,b) = \sum_{j \in B(b)} \mathbf{W}(i,j), \tag{1}$$

where b is the current branch and $B(b)$ is the set of bones in a

1. The limited field of view of each of the cameras and projectors prevents us from capturing the entire height of the human subject.

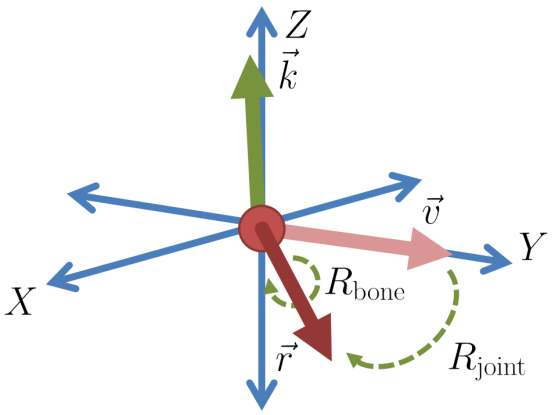


Fig. 6: An example of the decomposition of $\mathbf{R}_{\text{local}}$ into $\mathbf{R}_{\text{joint}}$ and $\mathbf{R}_{\text{bone}}$. The original pose of a bone in P_0 always lies along the y-axis with unit vector $\vec{v}$. The axis of rotation used to rotate between $\vec{v}$ and final bone position $\vec{r}$ is $\vec{k}$.

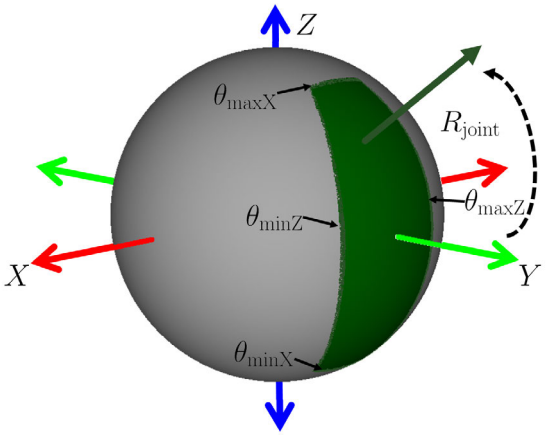
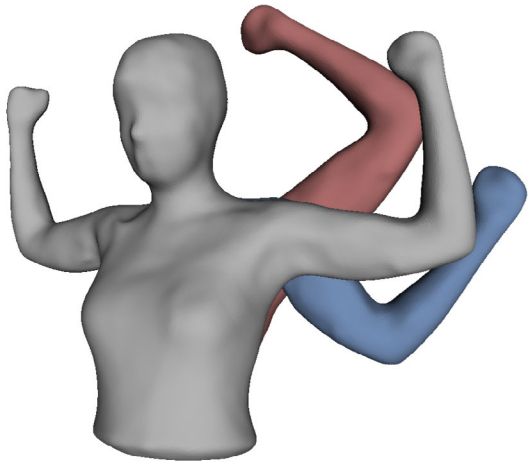
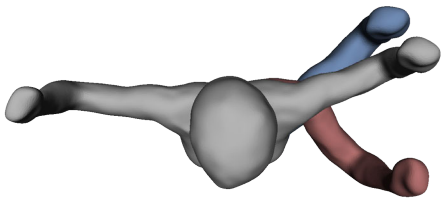


Fig. 7: Illustration of a constrained region for a skeleton joint. The bone out of a joint is always aligned with the y-axis in the original template pose. The green region shows valid new positions into which $\mathbf{R}_{\text{joint}}$ can rotate the bone. The four sides of the valid region are constrained by the rotation limits around the x and z-axes.



(a)



(b)

Fig. 8: An example shoulder joint rotated (a) along the x-axis and (b) along the z-axis of the local coordinate frame. The figure in gray is the original pose. The red model represents a positive rotation around the axis and blue represents a negative rotation.

branch. We use branch weights when calculating how well the vertices in a given branch fit the target frame geometry.

For each bone, we specify a range of motion under which it can be rotated relative to its parent bone. A parent bone p_j is defined as the bone connected to bone j that is higher in the skeleton tree and connected to the lead joint of bone j. For example, in Fig. 5, bone 5 is the parent of bone 6. We define a local right-handed coordinate system for each bone in the original pose P_0 of the template, shown in Fig. 5, that places each joint at the origin and the bone out of that joint on the y-axis as shown in Fig. 6. The constraint set for each joint is generated by inspecting each joint in the template under varying amounts of rotation and selecting a rotation range for the joint relative to its parent that falls within the physical range of motion for a human subject. Specifically, the constraints set for each joint are defined as minimum and maximum limits of rotation on each of the x, y, and z-axes relative to its parent joint. The rotation constraints can be applied directly only when a joint rotates about a single coordinate axis. Since in practice, this is not often the case, we need to compute a different constraint representation based on the values from the individual axis rotation constraints. To do this, we represent the constraint for each joint in two parts. The first component $\mathbf{R}_{\text{joint}}$ is a rotation matrix that represents the portion of the joint rotation that occurs around an axis lying in the x-z plane of the joint's local coordinate system as shown in Fig. 6. This rotation moves the bone from the y-axis to its new position shown in red in Fig. 6. The second component of the joint rotation, represented by the rotation matrix $\mathbf{R}_{\text{bone}}$, rotates around the new axis of the bone shown in red in Fig. 6. Thus, the total transformation $\mathbf{R}_{\text{local}}$ is given by:

$$\mathbf{R}_{\text{local}} = \mathbf{R}_{\text{bone}}\mathbf{R}_{\text{joint}}. \quad (2)$$

We refer to this total transformation as $\mathbf{R}_{\text{local}}$ since it denotes the transformation of the joint with respect to its own local coordinate frame. Since the rotation $\mathbf{R}_{\text{joint}}$ rotates around an axis that lies in the x-z plane, no component of $\mathbf{R}_{\text{joint}}$ rotates around the y-axis. Therefore, the rotation matrix $\mathbf{R}_{\text{joint}}$ is constrained by the individual minimum and maximum constraints for the x-axis and z-axis. As seen in Fig. 7, the valid range in which $\mathbf{R}_{\text{joint}}$ can be defined is constrained by the regions that fall within the minimum and maximum angles of rotation for the x-axis, namely $\theta_{\text{min}X}$ and $\theta_{\text{max}X}$, and the z-axis, namely $\theta_{\text{min}Z}$ and $\theta_{\text{max}Z}$. Once the bone has been put in position by $\mathbf{R}_{\text{joint}}$, we constrain the amount of rotation around the bone by using the angle limits for the y-axis,

Joint	Rotation Limits (radians)					
	$\theta_{\text{min}X}$	$\theta_{\text{max}X}$	$\theta_{\text{min}Z}$	$\theta_{\text{max}Z}$	θ_{minBone}	θ_{maxBone}
0	-3.14	3.14	-3.14	3.14	-3.14	3.14
1	-0.4	0.4	-0.4	0.4	-0.4	0.4
2	-0.6	0.4	-0.4	0.4	-0.8	0.8
3	-0.4	0.4	-0.4	0.4	-1.4	1.4
4	0.0	0.4	-0.2	0.2	-0.4	0.4
5	-1.0	1.2	-0.5	1.4	-2.8	0.4
6	-0.8	0.8	-1.4	1.1	-0.8	1.6
7	-0.4	0.4	-0.4	0.4	0.0	0.0
8	-0.1	0.4	-0.2	0.2	-0.4	0.4
9	-1.0	1.2	-1.4	0.4	-0.4	2.8
10	-1.0	1.5	-1.2	1.0	-0.4	1.2
11	-0.2	0.4	-0.4	0.8	0.0	0.0

TABLE 1: A sample set of constraints used for the template.

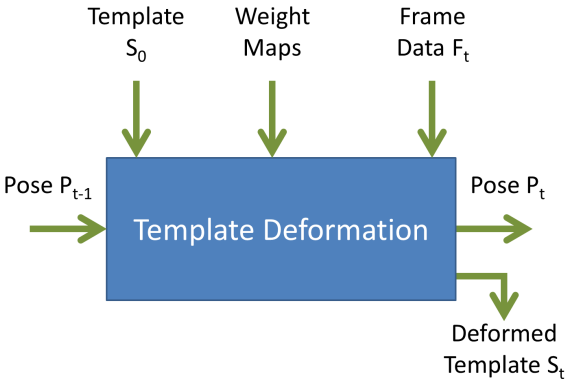


Fig. 9: The basic input and output of the iterative mesh deformation method.

since rotating around the y-axis in the original pose is the same as rotating around the bone axis in the pose after $\mathbf{R}_{\text{joint}}$ is applied. We refer to the minimum and maximum limits of rotation around the bone axis as θ_{minBone} and θ_{maxBone} respectively.

We choose this representation for joint rotations because it is intuitive to constrain. For example, for the joint at the wrist, we expect a limited range of rotation in the x-z plane and zero rotation around the bone axis. Even with correct alignment of the skeleton to existing geometry, joint constraints are important. For regions with significant missing geometry due to occlusion, for example, the constraints keep the skeleton in a reasonable pose until the true geometry can be observed again. Table 1 shows a sample set of motion constraints for the 12 joints in our template. To generate the values in the table, the template is deformed at each joint along each of the x, y, and z-axes, is visualized over a range of deformation angles from $[-\pi,\pi)$, and the subset of this range of angles that appears as a physically plausible limit is selected. The table is populated by iteratively examining each joint over the three possible axes. Fig. 8 presents sample poses of a shoulder joint rotated about the local x-axis and z-axis.

4 Processing a Sequence of Frames

In this section, we describe our method for fitting a template to captured 3D scans. Reconstructing the dynamic shape of a human subject occurs in several nested iterative processes. At the top level, we deform the template to each fused 3D scan resulting from one round of three stations capturing the scene. In doing so, we iteratively update the orientation of each bone in the template to bring it into alignment with the 3D captured geometry. We repeat this process for each consecutive new set of data.

The human subject is captured from only one station at a time at a rate of 60 Hz. At each station, two cameras generate point clouds representing their views of the scene. Each of these two points clouds are meshed into surfaces, which in turn allows for normals to be estimated for each vertex. We merge the six 3D surfaces generated in three consecutive time steps, two from each station, into a partial surface reconstruction of the scene. We refer to the merged partial surfaces captured at time t as a frame F_t. The merged surfaces are not re-meshed to create a single surface; rather, the partial reconstruction is represented as a set of multiple disconnected surfaces. In the process of fitting the template to the partial reconstructions, the surface connectivity of the partial scans is less important than the normal and location of each vertex in the partial surface since these are used for finding correspondences.

To capture the overall movement of the human subject over time, the template is sequentially fit to each frame as shown in Fig. 9. The captured data at time t, denoted by F_t, is processed to compute the pose of the human subject at time t, denoted by P_t. We refer to the template in its original pose, as shown in Fig. 5, as S_0 and the corresponding original skeleton pose itself as P_0. The final skeleton pose that aligns S_0 with frame F_t is referred to as P_t. As seen in Fig. 9, given the template S_0, the captured frame F_t, the weight maps $\mathbf{W}$ and $\bar{\mathbf{W}}$, and the pose P_{t-1} at time $t-1$, we deform the template S_0 to fit the target frame F_t and generate pose P_t and deformed template S_t as output. The steps for template deformation are shown in the flow diagram of Fig. 10. At a high level, the template deformation takes place in three main steps: initialization, processing, and finalizing. During initialization, all data needed for processing is loaded and the template is deformed to the pose from the previous frame. Within the processing step, the template is evaluated to determine whether it closely fits the target frame. If it does not, the branch fitting loop is executed. During branch fitting, the poses of branches within the skeleton are iteratively updated to align to the target geometry. After each pass through branch fitting, the fit between the source template and the target frame is reevaluated. Once a good fit is found, the processing step is concluded and the finalizing step saves the pose and final template for processing the next frame. We now explain each step in more detail.

During initialization, the template S_0, vertex weight map $\mathbf{W}$, branch weight map $\bar{\mathbf{W}}$, skeleton pose from the previous time step P_{t-1}, and frame F_t are all loaded. The pose of each bone is represented as a rotation matrix plus translation vector. The pose of a skeleton P_t specifies the transformation of each bone from the original template pose P_0 to its position and orientation at time t, and it is represented by the set of rotation and translation vectors for each bone. While initializing for time instance t, the template S_0 is deformed according to the skeleton pose P_{t-1} that was aligned to frame F_{t-1}. Since the frames are processed sequentially, pose P_{t-1} is the best initial estimate for target frame F_t. The bone weight map $\mathbf{W}$ assigning the influence of each bone on each template vertex is needed to properly deform the template. The vertices of the template are updated to match the skeleton pose using dual quaternion skinning. We refer to this updated template as S_t^{curr} since it is the current best estimate for aligning with F_t.

This deformed template is used as input to the "processing"

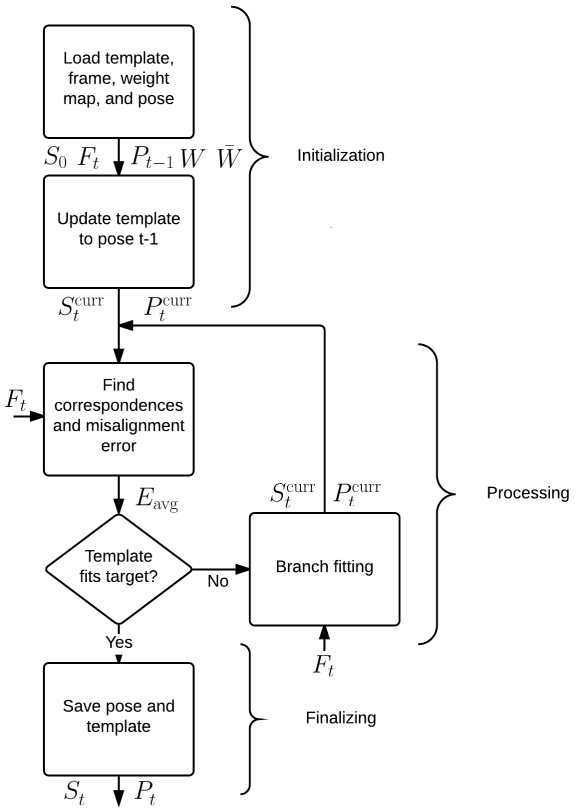


Fig. 10: Processing steps for the template deformation block shown in Fig. 9.

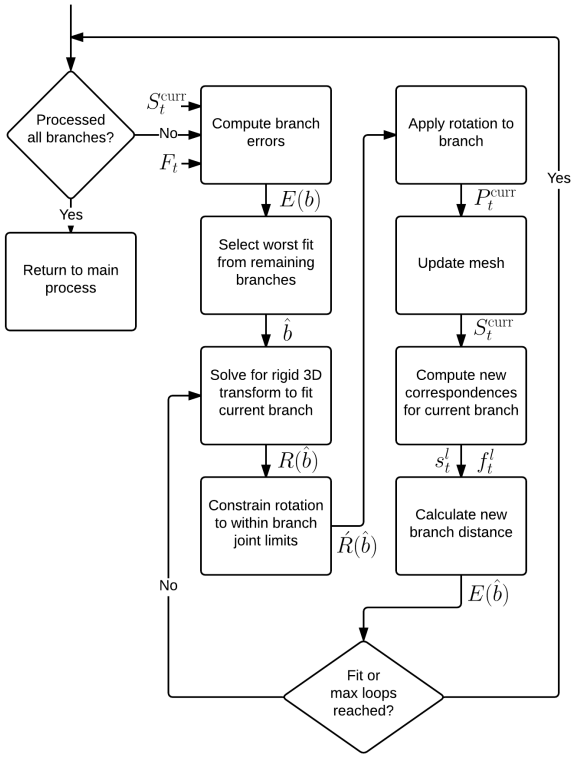


Fig. 11: Processing steps for the branch fitting block showed in 10.

portion of our algorithm shown in Fig. 10. Correspondences from template S_t^{curr} to the frame F_t are generated, and the average distance between all correspondence points is found. Specifically, a correspondence for each vertex in frame F_t is found by locating the closest vertex in template S_t^{curr} that lies in the direction of the frame vertex's normal. To prevent erroneous correspondences, the maximum distance is kept below a preset threshold. The average misalignment distance over all correspondence pairs is calculated and represented as E_{avg}. If this average error is small, then template S_t^{curr} is well aligned with the frame F_t and the finalizing step can start. If S_t^{curr} it not yet well aligned with F_t, then the branch fitting process must be executed. In the branch fitting process to be described in Section 4.1, the pose of each branch of the skeleton P_t^{curr} is updated to bring template S_t^{curr} into alignment with the frame F_t. For each frame, the branch fitting process is executed multiple times until a reasonable fit is found.

In the finalizing step, the pose of the skeleton P_t^{curr} that transforms the source template S_0 into alignment with the frame F_t is stored along with the deformed template S_t^{curr}. The deformed template itself is not used during subsequent processing, but its pose is used during the initialization step of the following frame. The deformed template is the main output of our system corresponding to the dynamic geometry of the human subject. The heart of our approach for template deformation lies within the branch fitting process to be described next.

4.1 Branch Fitting

The branch fitting process is illustrated in Fig. 11. At a high level, the pose of each branch in the template skeleton is updated to bring the source template into alignment with the target frame. This process is run until each branch in the skeleton has been updated. The order in which the branches are processed is determined by the misalignment error for each branch, starting with the worst fit branch and continuing in order of decreasing error.

The branch fitting process begins by checking whether all the branches have been processed. If they have not, the error for each branch is calculated. To do this, correspondences between the template S_t^{curr} and captured frame F_t are found as described earlier. We denote the correspondence i as a vertex s_t^i in the template S_t^{curr} and a vertex f_t^i in frame F_t. The Euclidean distances between each corresponding vertex pair in the source and target $d(s_t^i, f_t^i)$ are computed. Using these values, the average misalignment of correspondences of each branch b is

$$E(b) = \frac{\sum_{i \in N} d(s_t^i, f_t^i)\bar{\mathbf{W}}(i,b)}{\sum_{i \in N} \bar{\mathbf{W}}(i,b)}. \quad (3)$$

Each distance term is weighted by the branch weight map $\bar{\mathbf{W}}(i,b)$, which represents the total influence of branch b on the vertex position s_t^i. We select the most misaligned branch $\hat{b}$ and correct it before any other branch:

$$\hat{b} = \arg\min_{b \in \mathbb{B}} E(b), \quad (4)$$

where $\mathbb{B}$ is the set of possible branches.

Once the branch is selected, the transformation to align the correspondences between the template S_t^{curr} and frame F_t^{curr} in the branch is estimated by solving for a rotation and translation. Specifically, we use a standard 3D rigid transform to align the correspondences for vertices in branch $\hat{b}$ [26]. The 3D correspondence pair for vertex l that is fed into the transform is weighted

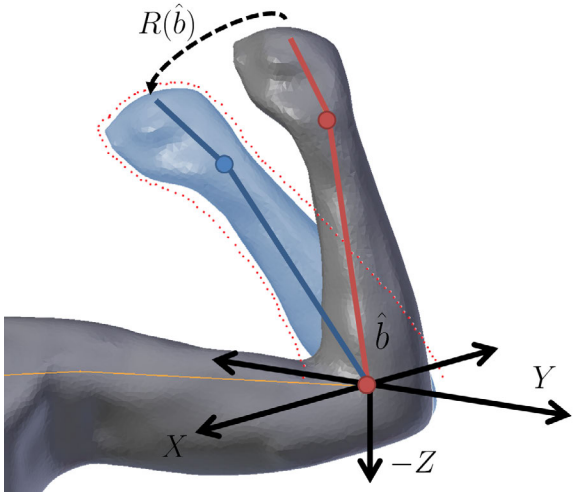


Fig. 12: An example of branch fitting. The forearm branch of a template is rotated by $\mathbf{R}(\hat{b})$ to align with points in the target geometry shown as red dots. The new template pose after applying $\mathbf{R}(\hat{b})$ is shown in blue. The rotation occurs in the world coordinate frame with the origin positioned at the lead joint of the rotating branch.

as

$$w_{t,l} = \begin{cases} d(s_t^l, f_t^l) n_t^{s,l\mathsf{T}} n_t^{f,l} & \text{if } n_t^{s,l\mathsf{T}} n_t^{f,l} > 0, \\ 0 & \text{if } n_t^{s,l\mathsf{T}} n_t^{f,l} \leq 0, \end{cases} \quad (5)$$

where $n_t^{s,l}$ and $n_t^{f,l}$ are the normals of the vertices in the target and source, respectively. A correspondence weight is calculated for each vertex l in the set of vertices in branch $\hat{b}$. The correspondence weights are proportional to the distance between s_t^l and f_t^l in a correspondence pair. Correspondences with a large distance between the source and target vertices occur in regions where the source template is most misaligned with respect to the target frame, i.e. the regions that are most important to fit. To ensure correct correspondences, we penalize those without similar normals. Specifically, if the normals are more than 90 degrees apart, then they are likely matching the incorrect side of a surface and are thus ignored.

Before computing the rotation to align the correspondences, the vertices of the source and target are translated such that the origin of the world coordinate frame is centered at the lead joint of branch $\hat{b}$. The resulting rotation matrix $\mathbf{R}(\hat{b})$ and translation vector $\vec{T}(\hat{b})$ computed from the 3D transformation represent the update transformation of the branch in the world coordinate system centered at the lead joint of branch $\hat{b}$ [26]. Fig. 12 illustrates the forearm of a human template, shown in gray, being fit to target geometry, shown as red dots in the figure. By applying the rotation $\mathbf{R}(\hat{b})$, the forearm of the template is aligned with the target geometry. The rotation $\mathbf{R}(\hat{b})$ is applied in the world coordinate frame centered at the lead joint of branch $\hat{b}$. When fitting all branches other than the root branch, we discard the translation vector $\vec{T}(\hat{b})$ since the position of a branch's lead joint is determined by the parent bone connected to it. For example, in Fig. 5, the position of the lead joint of bone 7 is determined by the position and orientation of bone 6.

Once the update rotation $\mathbf{R}(\hat{b})$ is estimated, the next step is to ensure that the rotation is within the constrained limit for the branch. This is described in detail in Section 4.2. Constraining the

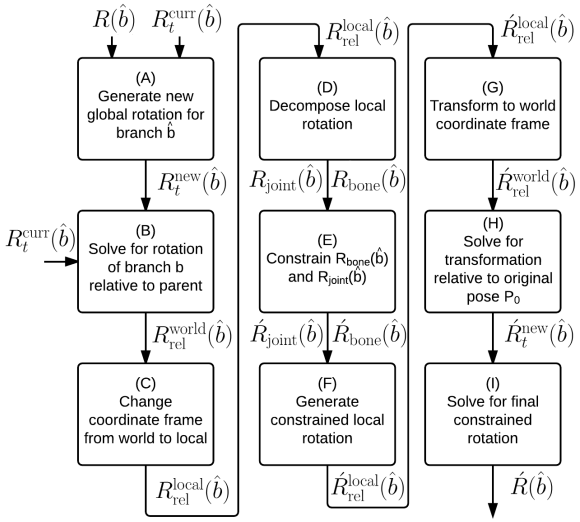


Fig. 13: Processing steps for the constraining a branch rotation block shown in 11.

update rotation of a branch results in a new rotation matrix $\acute{\mathbf{R}}(\hat{b})$. With the new rotation, the position of branch $\hat{b}$ is updated in P_t^{curr} and the template S_t^{curr} is updated according to this new pose. As done in the initialization, the template S_t^{curr} is deformed using dual quaternion skinning.

Given the new pose for the template S_t^{curr}, we recompute the correspondences for the updated branch $\hat{b}$, the one just rotated, and compute the average misalignment error for the branch using Equation 3. The process is repeated until the average misalignment error for $\hat{b}$ drops below a certain threshold for a preset number of iterations. Once this happens, we return to the beginning of the branch fitting process and check to see whether all branches have been fit. If they have, we exit the branch fitting process, otherwise, the next branch is processed. After all branches are processed, we exit the branch fitting process.

4.2 Constraining a branch rotation

At the time the update rotation $\mathbf{R}(\hat{b})$ of branch $\hat{b}$ is calculated in Fig. 11, the skeleton is in pose P_t^{curr} and the template S_t^{curr} reflects this pose. We need to apply $\mathbf{R}(\hat{b})$ to rotate branch $\hat{b}$ to better align the template mesh S_t^{curr} with the target frame F_t. The pose P_t^{curr} is represented by a set of rotation matrices $\mathbf{R}_t^{\text{curr}}(b)$ for each branch b that specify the amount of rotation the branch undergoes in the process of moving from the original template pose P_0 to the current pose P_t^{curr}. The rotations $\mathbf{R}_t^{\text{curr}}(b)$ are all represented in the world coordinate frame since this is the coordinate frame of the template S_t^{curr} and target F_t. On the other hand, to test whether a branch $\hat{b}$ satisfies its specified constraints, its rotation must be represented in the local coordinate frame of branch $\hat{b}$ in pose P_0 since the constraints of each branch are defined in this local coordinate frame. Therefore, to apply any constraint in the rotation of bones, we must reconcile these two coordinate systems.

The steps for constraining a branch rotation are illustrated in Fig. 13 and can be summarized as follows. In step A, we concatenate incremental rotation $\mathbf{R}(\hat{b})$ from Fig. 11 with the current rotation $\mathbf{R}_t^{\text{curr}}$ of the skeleton to determine the new rotation $\mathbf{R}_t^{\text{new}}(\hat{b})$ of branch $\hat{b}$ in the world coordinate system with respect to its pose in P_0. Then, in step B, we solve for the rotation of branch $\hat{b}$ relative to its parent branch $\hat{p}$ and represent it as $\mathbf{R}_{\text{rel}}^{\text{world}}(\hat{b})$. This

is because the constraints of each branch are defined relative to the orientation of their parent branch. Since the relative rotation must be represented in the local coordinate frame where the constraints are defined, in step C, the coordinate frame of the relative transformation $\mathbf{R}_{\text{rel}}^{\text{world}}(\hat{b})$ is converted from the world coordinate system to the local coordinate frame of branch $\hat{b}$ in pose P_0 denoted by $\mathbf{R}_{\text{rel}}^{\text{local}}(\hat{b})$. In step D, the local rotation is decomposed into the two rotation components $\mathbf{R}_{\text{bone}}(\hat{b})$ and $\mathbf{R}_{\text{joint}}(\hat{b})$ presented in Section 3. In step E, we constrain the two rotation components to meet the branch constraints resulting in $\acute{\mathbf{R}}_{\text{bone}}(\hat{b})$ and $\acute{\mathbf{R}}_{\text{joint}}(\hat{b})$. From here, the inverse of steps A–D are visited in reverse order. First in step F, the constrained local relative rotation $\acute{\mathbf{R}}_{\text{rel}}^{\text{local}}(\hat{b})$ is generated by composing $\acute{\mathbf{R}}_{\text{bone}}(\hat{b})$ and $\acute{\mathbf{R}}_{\text{joint}}$ and then, in step G, it is transformed to the world coordinate system $\acute{\mathbf{R}}_{\text{rel}}^{\text{world}}(\hat{b})$. In step H, the transformation is computed relative to the original pose P_0 rather than to the parent branch to generate $\acute{\mathbf{R}}_t^{\text{new}}(\hat{b})$, and, in step I, we remove the $\mathbf{R}_t^{\text{curr}}$ from $\acute{\mathbf{R}}_t^{\text{new}}(\hat{b})$ to obtain the final constrained rotation $\acute{\mathbf{R}}(\hat{b})$. We now explain each step in more detail.

The rotation $\mathbf{R}_t^{\text{curr}}(\hat{b})$ represents the rotation that branch $\hat{b}$ has undergone in moving from pose P_0 to pose P_t^{curr} in the world coordinate system. The update rotation $\mathbf{R}(\hat{b})$ in Fig. 11 is the incremental rotation we apply in addition to $\mathbf{R}_t^{\text{curr}}(\hat{b})$ to better align the vertices of branch $\hat{b}$ in S_t^{curr} with F_t. In step A, the total rotation $\mathbf{R}_t^{\text{new}}(\hat{b})$ for branch $\hat{b}$ is the concatenation of these two rotations as follows:

$$\mathbf{R}_t^{\text{new}}(\hat{b}) = \mathbf{R}(\hat{b})\mathbf{R}_t^{\text{curr}}(\hat{b}). \tag{6}$$

The rotation $\mathbf{R}_t^{\text{new}}(\hat{b})$ for branch $\hat{b}$ is represented in the world coordinate system. It describes the rotation of the branch in the same coordinate system as the template S_t^{curr} and target frame F_t. The rotation $\mathbf{R}_t^{\text{new}}(\hat{b})$ specifies how much branch $\hat{b}$ is being rotated from its orientation in pose P_0. Rather than represent the rotation of the branch with respect to its orientation in pose P_0, we would like to capture the rotation of branch $\hat{b}$ relative to its parent branch $\hat{p}$. This is because the constraints defined for each branch are represented as the amount of rotation a branch can undergo relative to its parent branch. The need for this representation is illustrated in Fig. 14. For a human posed with the upper arm extending out to the side, there is a set of rotations in the world coordinate system that describes the possible movement of the forearm, as shown in Fig. 14a. If the upper arm is extended to the front of the body as shown in Fig. 14b, the range of possible rotations for the forearm in the world coordinate system would now be different. However, the relative rotation of the forearm with respect to the upper arm would be the same in these two situations. We are interested in calculating this relative rotation since our constraints for a branch are defined in this relative manner. As such, in step B, the relative rotation $\mathbf{R}_{\text{rel}}^{\text{world}}(\hat{b})$ of branch $\hat{b}$ to the parent branch $\hat{p}$ is computed as

$$\mathbf{R}_{\text{rel}}^{\text{world}}(\hat{b}) = \mathbf{R}_t^{\text{new}}(\hat{b})\mathbf{R}_t^{\text{curr}}(\hat{p})^{-1}. \tag{7}$$

To illustrate the method of relative rotations, we present an example where branch $\hat{b} = 6$ and $\hat{p} = 5$ as labeled in Fig. 5. If a template takes on pose P_t the corresponding rotation matrices to generate this pose from P_0 are represented as $\mathbf{R}_t(b)$, $\forall\, b \in \mathbb{B}$. When the template is in pose P_0 as shown in Fig. 15a, $\mathbf{R}_{t=0}(\hat{b}) = \mathbf{R}_{t=0}(\hat{p}) = \mathbf{I}$, the identity matrix. In Fig. 15b, we depict a rotation about the branch $\hat{p}$ at time $t = 1$. In this pose, both branches 5 and 6 are rotated from their orientation in pose P_0 by a rotation of $\mathbf{R}_1$, that is $\mathbf{R}_{t=0}(\hat{p}) \neq \mathbf{R}_{t=1}(\hat{p})$, but $\mathbf{R}_{t=1}(\hat{p}) = \mathbf{R}_{t=1}(\hat{b}) = \mathbf{R}_1$ since both branches have undergone the same rotation. If now the

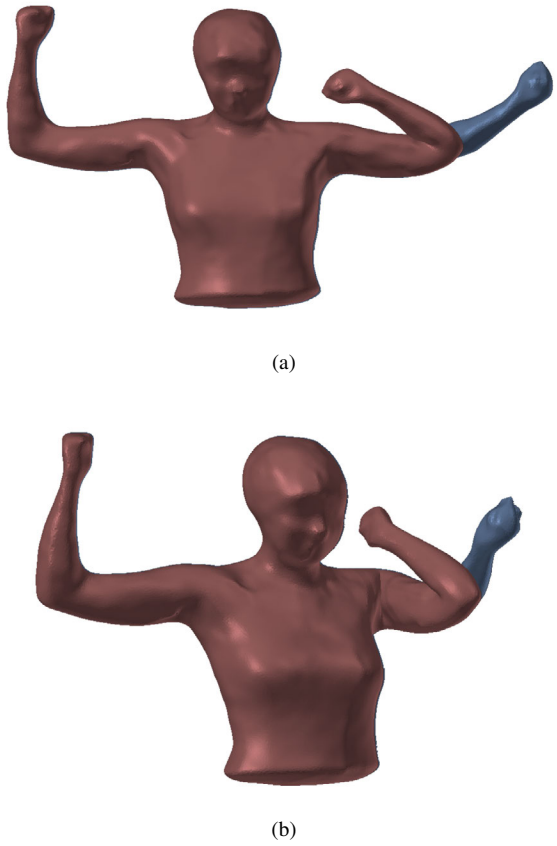


Fig. 14: An example range of rotation for an elbow joint is shown with the upper arm positioned (a) towards the side of the body and (b) towards the front of the body. Note that range of motion for the elbow is the same regardless of the position of the upper arm.

branch $\hat{b}$ is rotated at time $t = 2$ by a rotation of $\mathbf{R_2}$, as shown in Fig. 15c, then $\mathbf{R}_{t=1}(\hat{p}) = \mathbf{R}_{t=2}(\hat{p}) = \mathbf{R}_1$, but $\mathbf{R}_{t=2}(\hat{b}) \neq \mathbf{R}_{t=2}(\hat{p})$. Obviously $\mathbf{R}_{t=2}(\hat{b}) = \mathbf{R_2}\mathbf{R}_1$ and $\mathbf{R}_{t=2}(\hat{p}) = \mathbf{R}_1$. The rotation of branch $\hat{b}$ relative to $\hat{p}$ for pose P_2 can be determined by using Equation 7:

$$\mathbf{R}_{\text{rel}}^{\text{world}}(\hat{b}) = \mathbf{R}_{t=2}(\hat{b})\mathbf{R}_{t=2}(\hat{p})^{-1} = \mathbf{R_2}\mathbf{R}_1\mathbf{R}_1^{-1} = \mathbf{R_2}. \tag{8}$$

Thus, we are able to isolate the rotation of branch $\hat{b}$ relative to its parent branch $\hat{p}$ by using Equation 7. Additionally, since all rotations $\mathbf{R}_t(\hat{b})$ are defined in the world coordinate system, the relative rotation $\mathbf{R}_{\text{rel}}^{\text{world}}$ is also represented in the world coordinate system.

In step C, we convert the relative rotation $\mathbf{R}_{\text{rel}}^{\text{world}}(\hat{b})$ from the world coordinate frame to the local coordinate frame of branch $\hat{b}$ in pose P_0. The coordinate transformation between a branch $\hat{b}$ in the local coordinate frame of pose P_0 and the world coordinate frame is represented as $\mathbf{R}_{l\to w}(\hat{b})$. We use this coordinate transformation to convert the relative rotation from world coordinates to local coordinates:

$$\mathbf{R}_{\text{rel}}^{\text{local}}(\hat{b}) = \mathbf{R}_{l\to w}(\hat{b})^{-1}\mathbf{R}_{\text{rel}}^{\text{world}}(\hat{b})\mathbf{R}_{l\to w}(\hat{b}). \tag{9}$$

The relative rotation is now represented in the correct local coordinate frame, but it needs to be decomposed into a different representation before the constraints can be tested. In step D, we decompose $\mathbf{R}_{\text{rel}}^{\text{local}}(\hat{b})$ into $\mathbf{R}_{\text{joint}}(\hat{b})$ and $\mathbf{R}_{\text{bone}}(\hat{b})$, as in Section

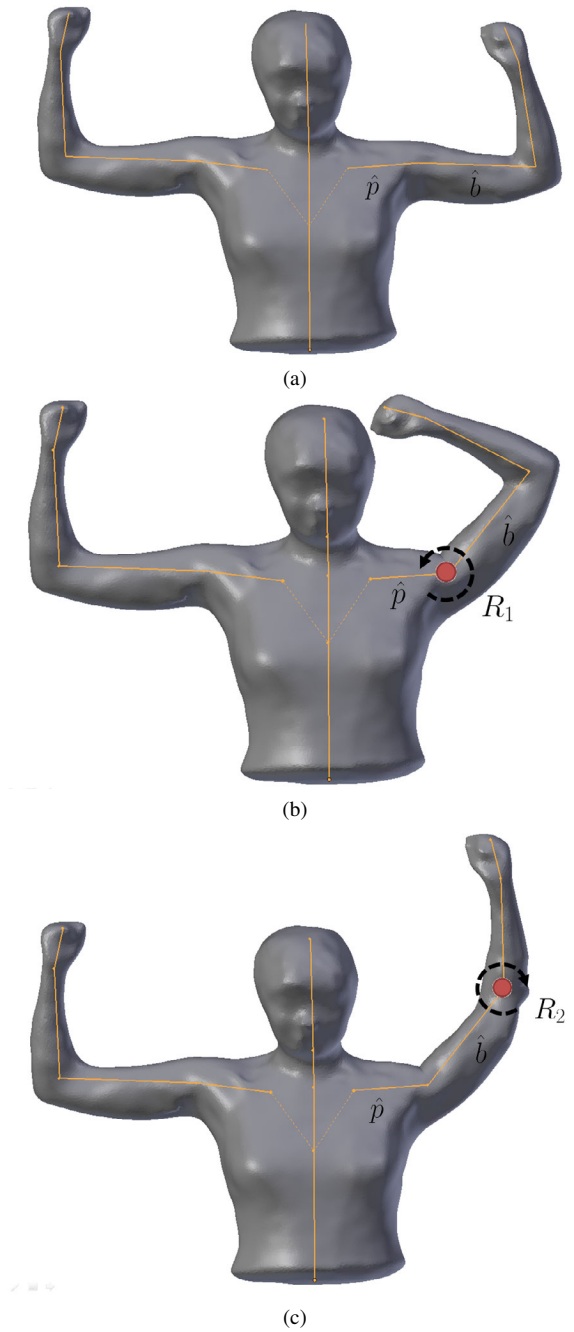


Fig. 15: Illustration of the template in three consecutive poses: (a) the template fit to pose P_0, (b) pose P_1 where branch $\hat{p} = 5$ is rotated, and (c) pose P_2 where branch $\hat{b} = 6$ is rotated.

3, to match the same form in which the branch constraints are defined. The constraints for $\mathbf{R}_{\text{joint}}(\hat{b})$ are specified as allowable ranges of rotation around the x-axis and z-axis as shown in Fig. 7. The constraint for $\mathbf{R}_{\text{bone}}(\hat{b})$ is specified as an allowable range of rotation around the axis of the bone, as shown in Fig. 6. The rotation $\mathbf{R}_{\text{joint}}(\hat{b})$ captures the new position to which a bone would be moved to if rotated by $\mathbf{R}^{\text{local}}_{\text{rel}}(\hat{b})$. The rotation $\mathbf{R}_{\text{bone}}(\hat{b})$ accounts for any rotations around this bone caused by $\mathbf{R}^{\text{local}}_{\text{rel}}(\hat{b})$. Deriving $\mathbf{R}_{\text{joint}}(\hat{b})$ and $\mathbf{R}_{\text{bone}}(\hat{b})$ from $\mathbf{R}^{\text{local}}_{\text{rel}}(\hat{b})$ is performed in two steps. First, we solve for $\mathbf{R}_{\text{joint}}(\hat{b})$ and then generate $\mathbf{R}_{\text{bone}}(\hat{b})$ by using $\mathbf{R}_{\text{joint}}(\hat{b})$ and $\mathbf{R}^{\text{local}}_{\text{rel}}(\hat{b})$.

The rotation $\mathbf{R}_{\text{joint}}(\hat{b})$ captures the movement of the bone to a new location caused by $\mathbf{R}^{\text{local}}_{\text{rel}}(\hat{b})$. In the local coordinate frame of each branch in original pose P_0, the bone out of each joint lies along the y-axis as shown in Fig. 6. We can determine the new position of the bone by transforming a y-direction unit vector by $\mathbf{R}^{\text{local}}_{\text{rel}}(\hat{b})$ as follows:

$$\vec{r} = \left[\mathbf{R}^{\text{local}}_{\text{rel}}(\hat{b}).\right] \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{10}$$

The vector $\vec{r}$ accounts for the new location of the bone, but it does not represent the amount of rotation around the axis of the bone. We wish to solve for the rotation needed to transform the original pose of the bone in P_0, a unit-length vector on the y-axis, to the new position $\vec{r}$ without any rotation around the y-axis. The rotation matrix $\mathbf{R}_{\text{joint}}(\hat{b})$ can be represented in an axis-angle representation, where $\vec{k}$ is the axis of rotation as shown in Fig. 6 and θ_{joint} is the angle of rotation about the axis. The axis of rotation for $\mathbf{R}_{\text{joint}}(\hat{b})$ must lie in the x-z plane of the joint's local coordinate system to effectively separate the rotation about the bone axis in $\mathbf{R}_{\text{bone}}(\hat{b})$. To transform a y-axis unit vector to align with vector $\vec{r} = [r_x r_y r_z]^\intercal$, the axis of rotation $\vec{k}$ is defined as $[r_z\ 0\ -r_x]^\intercal / |[r_z\ 0\ -r_x]|$. The vector $\vec{k}$ is perpendicular to $\vec{r}$ and lies in the x-z plane as shown in Fig. 6. By rotating a y-axis unit vector around $\vec{k}$, the y-axis unit vector can be aligned with $\vec{r}$. We set the original direction of the bone vector to $\vec{v} = [0\ 1\ 0]^\intercal$, and we use the Rodrigues' rotation formula to solve for θ_{joint} [27]:

$$\vec{r} = \vec{v}\cos\theta_{\text{joint}} + (\vec{k} \times \vec{v})\sin\theta_{\text{joint}} + \vec{k}(\vec{k} \cdot \vec{v})(1 - \cos\theta_{\text{joint}}). \tag{11}$$

Given the rotation vector is $\vec{k}$ and the angle of rotation is θ_{joint}, we can solve for the rotation matrix $\mathbf{R}_{\text{joint}}(\hat{b})$ using the matrix representation of the Rodrigues' formula [27]:

$$\mathbf{R}_{\text{joint}}(\hat{b}) = \mathbf{I} + (sin\theta_{\text{joint}})\mathbf{K} + (1 - cos\theta_{\text{joint}})\mathbf{K}^2, \tag{12}$$

where $\mathbf{K}$ is the cross-product matrix for $\vec{k}$ and $\mathbf{I}$ is the 3×3 identity matrix. Given $\mathbf{R}^{\text{local}}_{\text{rel}}(\hat{b})$ and $\mathbf{R}_{\text{joint}}(\hat{b})$, we can solve for $\mathbf{R}_{\text{bone}}(\hat{b})$ using Equation 2:

$$\mathbf{R}_{\text{bone}}(\hat{b}) = \mathbf{R}^{\text{local}}_{\text{rel}}(\hat{b})\mathbf{R}^{-1}_{\text{joint}}(\hat{b}) \tag{13}$$

At this point we have converted the local rotation $\mathbf{R}^{\text{local}}_{\text{rel}}(\hat{b})$ into the proper form of our constraints $\mathbf{R}_{\text{bone}}(\hat{b})$ and $\mathbf{R}_{\text{joint}}(\hat{b})$. In step E, the individual rotations are compared against the branch constraints in Table 1 and modified to fit as necessary. First, we check whether the position of $\vec{r}$ falls within a valid rotation range for the x and z axes. The angle between the y-axis and vector $\vec{r}$ is used to calculate θ_X and θ_Z, the rotation of $\vec{R}$ around the x-axis and z-axis, respectively. As shown in Fig. 7, there is a range of valid positions into which the branch can be rotated, shown in green in the figure. Specifically, we must ensure that $\theta_{\text{min}X} < \theta_X < \theta_{\text{max}X}$ and $\theta_{\text{min}Z} < \theta_Z < \theta_{\text{max}Z}$. If $\vec{r}$ falls outside of the constrained region, it is moved into a valid position. Using Rodrigues' formula in Equation 11, the value of θ_{joint} is updated to a new angle $\acute{\theta}_{\text{joint}}$ to move $\vec{r}$ into a valid new position $\vec{\acute{r}}$. Applying Equation 12, the new angle $\acute{\theta}_{\text{joint}}$ and $\vec{k}$ are used to generate $\acute{\mathbf{R}}_{\text{joint}}(\hat{b})$.

Next, we convert $\mathbf{R}_{\text{bone}}(\hat{b})$ into its axis-angle representation. By definition, $\mathbf{R}_{\text{bone}}(\hat{b})$ must rotate around the axis of the bone. We determine the angle of rotation θ_{bone} for $\mathbf{R}_{\text{bone}}(\hat{b})$ and ensure it meets the constraints $\theta_{\text{minBone}} < \theta_{\text{bone}} < \theta_{\text{maxBone}}$. The rotation θ_{bone} is found using the matrix to axis-angle conversion for

rotation matrices [27]:

$$\theta_{\text{bone}} = \arccos\left(\frac{\text{trace}(\mathbf{R}_{\text{bone}}(\hat{b})) - 1}{2}\right). \quad (14)$$

If θ_{bone} does not meet the set constraints, it is modified to the closer value of θ_{minBone} and $\theta_{\text{max}Bone}$. The constrained rotation is updated as $\acute{\theta}_{\text{bone}}$. Using $\acute{\theta}_{\text{bone}}$ and $\vec{r}$, the new rotation around the bone, $\acute{\mathbf{R}}_{bone}(\hat{b})$, is computed using Equation 12.

The final four steps of Fig. 13, steps F–I, are the inverse of steps A–D in reverse order. They are executed to convert the constrained rotation back to the proper coordinate frame and representation. In step F, we compose $\acute{\mathbf{R}}_{\text{bone}}(\hat{b})$ and $\acute{\mathbf{R}}_{\text{joint}}(\hat{b})$ into one rotation to generate local rotation:

$$\acute{\mathbf{R}}^{\text{local}}_{\text{rel}}(\hat{b}) = \acute{\mathbf{R}}_{\text{bone}}(\hat{b})\acute{\mathbf{R}}_{\text{joint}}(\hat{b}). \quad (15)$$

In step G, $\acute{\mathbf{R}}^{\text{local}}_{\text{rel}}(\hat{b})$ which is in the constrained relative rotation in the local coordinate system is transformed to the world coordinate system:

$$\acute{\mathbf{R}}^{\text{world}}_{\text{rel}}(\hat{b}) = \mathbf{R}_{l\to w}(\hat{b})\acute{\mathbf{R}}^{\text{local}}_{\text{rel}}(\hat{b})\mathbf{R}_{l\to w}(\hat{b})^{-1}. \quad (16)$$

$\acute{\mathbf{R}}^{\text{world}}_{\text{rel}}(\hat{b})$ represents the constrained relative rotation of branch $\hat{b}$ with respect to its parent. In step H, $\acute{\mathbf{R}}^{\text{world}}_{\text{rel}}(\hat{b})$ is modified so that it defines the rotation of $\hat{b}$ with respect to the original pose of the branch in pose P_0:

$$\acute{\mathbf{R}}^{\text{new}}_{t}(\hat{b}) = \acute{\mathbf{R}}^{\text{world}}_{\text{rel}}(\hat{b})\mathbf{R}^{\text{curr}}_{t}(\hat{p}). \quad (17)$$

Finally, in step I, the rotation of the branch due to the pose $\mathbf{R}^{\text{curr}}_{t}(\hat{b})$ is removed to isolate $\acute{\mathbf{R}}(\hat{\mathbf{b}})$:

$$\acute{\mathbf{R}}(\hat{b}) = \acute{\mathbf{R}}^{\text{new}}_{t}(\hat{b})\mathbf{R}^{\text{curr}}_{t}(\hat{b})^{-1}. \quad (18)$$

Once $\acute{\mathbf{R}}(\hat{b})$ is calculated, the constrained rotation is returned to the calling branch fitting process in Fig 11.

5 Texturing the Template

In addition to capturing the realistic movement of the human subjects, we are also interested in capturing the colors of the human subject's clothes and skin. This is done by using the three color cameras located on the three system stations. Since the connectivity and number of vertices in the template does not change as it is deformed in each frame, each vertex should correspond to the same location on the human subject throughout the captured sequence. For example, the vertex located on the nose of the human subject should correspond to the nose in all frames of data.

We opt to texture the human subject primarily from three color images captured at the same time as the frame used to generate the template, specifically at time $t = 0$. These three images are aligned with the template in its original pose and expose a maximum amount of the subject's surface to the color cameras. With the three color cameras located on each structured-light station, the majority of the surface of the template can be textured as shown in Fig. 16a. To texture, we project the vertices of the template S_0 onto the color images and each vertex is assigned the color of the pixel it is projected onto.

Template regions that are not visible to the texture cameras at time $t = 0$ are shown in black in Fig. 16a. They make up regions such as the top of the head and top and bottom of the arms. We provide an estimate of the colors for these vertices by using the remaining color images captured during each target frame. Specifically, the deformed template generated for each target frame is textured using the color images captured at the same time as the target frame. Each deformed mesh is textured in the same manner as the original template. However, we do not expect as many of the vertices to be textured since most frames have more occlusions than the original template pose.

Fig. 16: The template is shown textured: (a) using only color images from the template frame, (b) using median colors mapped to each template vertex, and (c) using a blend of textures (a) and (b).

Once the deformed templates for all frames are textured, we analyze the color vector assigned to each vertex in each frame of the sequence. In particular, we focus on the vertices that are not directly assigned colors from color images of time $t = 0$. Although some colors may be incorrect in individual frames, we can use the entire color vector for a vertex to estimate a representative color. To do this, the vector of colors for a single vertex is sorted in order of increasing intensity in the HSV color space. We use this color

space rather than RGB so as to sort according to a single value per vertex. By selecting the color with the median intensity for each vertex, we obtain a reasonable estimate of the color for all vertices in the mesh. Fig. 16b shows the median color for each vertex in the template. It does not have as much detail as the direct projection of texture from images as in Fig. 16a, but it still represents the human subject's texture well.

The final texture for the template is generated by texturing the majority of the template with the color images from the template frame and by filling the empty regions with the median selected colors. The seams between the two regions are manually blended to improve the overall texture quality, as shown in Fig. 16c. In the event that a vertex of the mesh is never viewed by a color camera, the vertex will remain black. To correct these cases, the vertex color is copied from the nearest neighboring pixels with a computed texture. However, in the capture sequence presented in Section 6, each vertex is viewable by the observing cameras in at least a few frames, so we do not need to perform this texture copy process.

We have empirically found that directly applying the texture for frame t using the corresponding color images at time t results in unnatural, displeasing artifacts compared to the proposed texturing method. This is because the geometry in each frame is from a deformed version of the template; as such, any slight misalignment in the estimate of the pose for the template can result in misaligned texture. This can especially be true in regions where local surface deformations are not captured by the deformed template.

6 Results

We test our deformation method on a 50-second animation sequence consisting of 992 frames. Starting from the grayscale images that capture the human subject under structured light, a sequence of phase images are calculated for each camera. These phase images are unwrapped as in [22] and a sequence of point clouds are generated from each camera. The point clouds from each of the six geometry capturing cameras are filtered, meshed, and transformed into a single world coordinate frame. The result is a sequence of partial scans that we use to deform our template. During processing, we set parameters to iterate through all branches four times per frame. Additionally, the rigid transformation of each branch is estimated at most four times before proceeding to the other branches.

Fig. 17 shows the transformation of each branch over time in the local coordinate frame. The branches are labeled as in Fig. 5 and the rotation angles θ_X, θ_Z, and θ_{bone} of each branch are relative to the branch's orientation in pose P_0. Specifically, the angles for a bone b are derived from the rotation matrices $\hat{\mathbf{R}}_{\text{joint}}(b)$ and $\hat{\mathbf{R}}_{\text{bone}}(b)$ computed from the final pose P_t of each frame shown in Fig. 9. The sequence of captured geometry starts with the human subject in pose P_0, which is reflected in Fig. 17 as the θ_X, θ_Z, and θ_{bone} angles for each branch set to zero. Over time, the angles drift from this pose. The angles for the wrists, i.e. branches 7 and 11, are not illustrated because we did not allow for rotation around these joints in the final reconstruction. Limited geometry captured for the hands in most frames makes it difficult to constrain their movement consistently. As seen, there is a significant variation in the amount of rotation for each branch. The branches in the arms, specifically branches 5, 6, 9, and 10, exhibit the greatest variation in orientation. Sample poses shown in Fig. 18 illustrate the large range of motion that the arms exhibit during the the captured sequence.

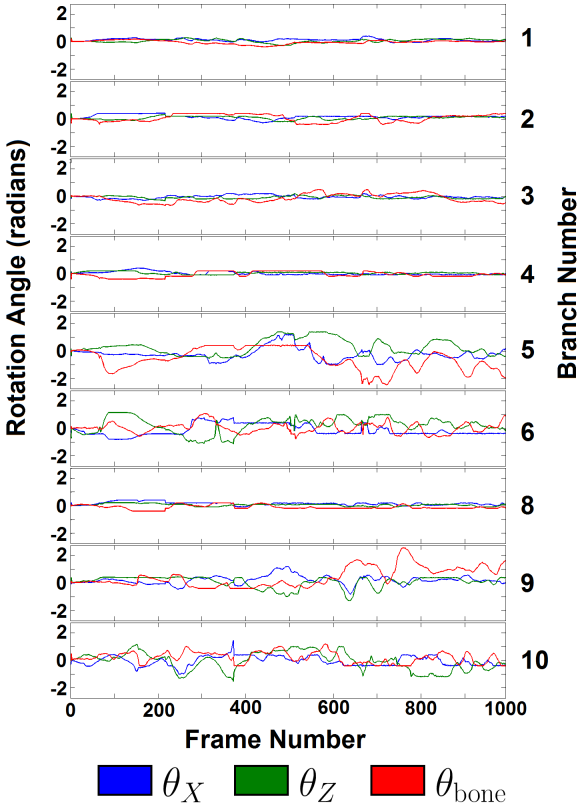


Fig. 17: Rotation angles for branches in Fig. 5. For each branch, the sequence of values for θ_X, θ_Z, and θ_{bone} are plotted over the frame indices.

Fig. 18: The template is deformed to fit the partial scans in a variety of poses.

The rotation of the root branch 0 around the bone axis θ_X is plotted in Fig. 19. Since the root branch does not have a parent branch, the rotation around the axis of the root bone is plotted relative to the world coordinate system. We show the pose of the template at approximate rotation angles of (a) 0, (b) $-\pi/2$, (c) π, and (d) $\pi/2$ in Fig. 20. As seen, the orientation of this angle controls the rotation of the template pose.

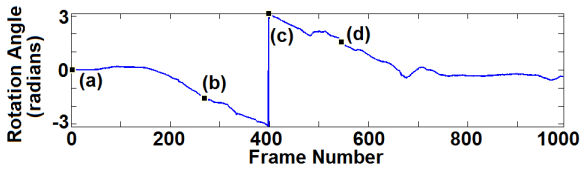


Fig. 19: Rotation of root branch relative to world coordinate system. Rotation angles (a) 0, (b) θ_X, (c) θ_Z, and (d) θ_{bone} are illustrated in Fig. 20.

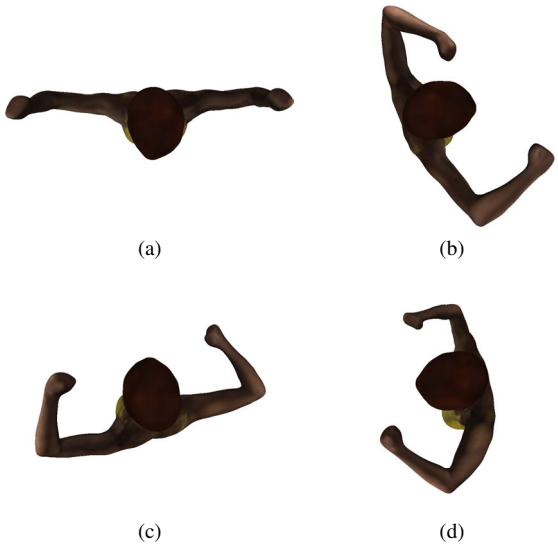


Fig. 20: Top view of deformed template for frames labeled in Fig. 19:(a) 0, (b) $-\pi/2$, (c) π, and (d) $\pi/2$.

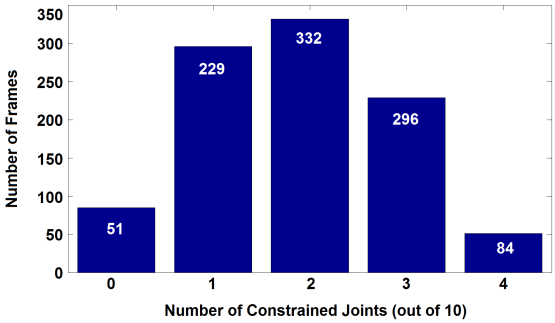


Fig. 21: Histogram of the number of constrained joints for all 992 frames.

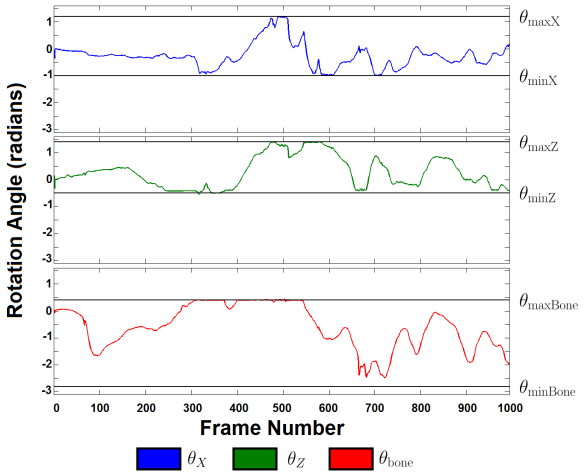


Fig. 22: Variation of θ_X, θ_Z, and θ_{bone} for branch 5 over captured sequence with angle limits plotted for each axis of rotation.

The effect of the joint constraints for the shoulder of the left arm, i.e. branch 5, is visualized in Fig. 22. As shown, the angles of each axis of rotation obey the specified constraints throughout the entire captured sequence. Although an angle may be constrained over multiple consecutive frames, the rotation of the angle resumes normally once the angle is within the constrained range. We find constraints to be most useful for branches not visible in a given pose. Additionally, some constraints are useful in ensuring that rotations occur at the correct joint. For example, the collar bones 4 and 8 in Fig. 5, are constrained in many frames to ensure that rotation of the arms are handled by the shoulder joints rather than the collarbones. This matches the anatomy of actual human skeletons which have a limited range of motion in collar bones compared to shoulders. We have empirically observed the majority of 992 to frames have at least one constrained joint. Fig. 21 shows the histogram of the number of constrained joints for all 992 frames. As seen, the majority of frames have only one or two joints constrained. This implies the iterative updates to the skeleton pose in successive frames are tracking to reasonable positions. Some of the constrained joints can be attributed to intentional restriction of some joints, as in the example with the collar bone above. We only consider the ten joints that were not constrained, excluding the two wrist joints.

As shown in Fig. 18, our deformation method is able to successfully capture the human subject in a variety of poses. Since we capture geometry from multiple sides of the human subject, the human subject is able to rotate and change the direction her body faces while still having the template successfully deform to match the geometry. The human subject must keep her fists closed during capture due to the limited resolution of the cameras in the system. The deformation processing of each frame takes less than two minutes with non-optimized prototype code. We expect this to be significantly improved by using the GPU.

The rendered view of the reconstructed sequence is shown in [28] and [29]. Specifically, [28] shows a side-by-side comparison of a video of the human subject and the same sequence reconstructed using our proposed approach. Also, [29] shows a 360-degree virtual view of the subject. As seen, our method is able to both accurately capture the geometry of the human subject and apply realistic texture. In [28], we can clearly observe our reconstructed geometry matching the motion of the human subject. In [29], the video shows that the motion is accurately captured on all sides of the human subject.

7 Conclusions

We have presented a multi-view structured-light system capable of performing markerless motion capture for human subjects. With rich captured 3D data rather than 2D video streams, we are able to easily deform a template to match partial scans of the human subject. Unlike existing methods, we can easily create templates directly from our system without having to rely on a special scanning process or needing high quality laser scans.

In future work, we plan to explore improve the quality and level of details of geometry resulting from the system. This could be done by testing other configurations of structured-light patterns or changing projector and camera hardware for models with higher speed and improved resolutions. This would allow us to also integrate local deformation methods to capture the subtle details of the human subject's geometry [30]. Additionally, we would like to decrease processing time by offloading some computing onto the GPU and by streamlining the 3D geometry generation and deformation steps.

REFERENCES

[1] T. B. Moeslund and E. Granum, "A survey of computer vision-based human motion capture," *Computer Vision and Image Understanding*, vol. 81, no. 3, pp. 231–268, 2001.

[2] T. B. Moeslund, A. Hilton, and V. Krger, "A survey of advances in vision-based human motion capture and analysis," *Computer Vision and Image Understanding*, vol. 104, no. 23, pp. 90–126, 2006, special Issue on Modeling People: Vision-based understanding of a persons shape, appearance, movement and behaviour.

[3] S. I. Park and J. K. Hodgins, "Capturing and animating skin deformation in human motion," in *ACM SIGGRAPH 2006 Papers*, ser. SIGGRAPH '06. ACM, 2006, pp. 881–889.

[4] S. Corazza, L. Mndermann, A. Chaudhari, T. Demattio, C. Cobelli, and T. P. Andriacchi, "A markerless motion capture system to study musculoskeletal biomechanics: Visual hull and simulated annealing approach," *Annals of Biomedical Engineering*, vol. 34, no. 6, pp. 1019–1029, 2006.

[5] C. Theobalt, N. Ahmed, H. Lensch, M. Magnor, and H.-P. Seidel, "Seeing people in different light-joint shape, motion, and reflectance capture," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 13, no. 4, pp. 663–674, 7 2007.

[6] E. de Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun, "Performance capture from sparse multi-view video," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 98:1–98:10, 8 2008.

[7] L. Ballan and G. M. Cortelazzo, "Marker-less motion capture of skinned models in a four camera set-up using optical flow and silhouettes," *3D Data Procesing, Visualization, and Transmission (3DPVT), Atlanta, GA, USA*, vol. 37, 2008.

[8] D. Vlasic, I. Baran, W. Matusik, and J. Popović, "Articulated mesh animation from multi-view silhouettes," in *ACM SIGGRAPH 2008 Papers*, ser. SIGGRAPH '08. ACM, 2008, pp. 97:1–97:9.

[9] 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 *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 6 2009, pp. 1746–1753.

[10] L. Mundermann, S. Corazza, and T. Andriacchi, "Accurately measuring human movement using articulated icp with soft-joint constraints and a repository of articulated models," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, 6 2007, pp. 1–6.

[11] S. Corazza, L. Mndermann, E. Gambaretto, G. Ferrigno, and T. Andriacchi, "Markerless motion capture through visual hull, articulated icp and subject specific model generation," *International Journal of Computer Vision*, vol. 87, no. 1-2, pp. 156–169, 2010.

[12] M. Dou, H. Fuchs, and J.-M. Frahm, "Scanning and tracking dynamic objects with commodity depth cameras," in *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, 10 2013, pp. 99–106.

[13] R. W. Sumner, J. Schmid, and M. Pauly, "Embedded deformation for shape manipulation," in *ACM SIGGRAPH 2007 Papers*, ser. SIGGRAPH '07. ACM, 2007.

[14] D. Vlasic, P. Peers, I. Baran, P. Debevec, J. Popović, S. Rusinkiewicz, and W. Matusik, "Dynamic shape capture using multi-view photometric stereo," in *ACM SIGGRAPH Asia 2009 Papers*, ser. SIGGRAPH Asia '09. ACM, 2009, pp. 174:1–174:11.

[15] H. Li, L. Luo, D. Vlasic, P. Peers, J. Popović, M. Pauly, and S. Rusinkiewicz, "Temporally coherent completion of dynamic shapes," *ACM Trans. Graph.*, vol. 31, no. 1, pp. 2:1–2:11, 2 2012.

[16] N. J. Mitra, S. Flöry, M. Ovsjanikov, N. Gelfand, L. Guibas, and H. Pottmann, "Dynamic geometry registration," in *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, ser. SGP '07. Eurographics Association, 2007, pp. 173–182.

[17] J. Süßmuth, M. Winter, and G. Greiner, "Reconstructing animated meshes from time-varying point clouds," *Computer Graphics Forum*, vol. 27, no. 5, pp. 1469–1476, 2008.

[18] T. Popa, I. South-Dickinson, D. Bradley, A. Sheffer, and W. Heidrich, "Globally consistent space-time reconstruction," *Computer Graphics Forum*, vol. 29, no. 5, pp. 1633–1642, 2010.

[19] A. Griesser, T. Koninckx, and L. Van Gool, "Adaptive real-time 3D acquisition and contour tracking within a multiple structured light system," in *Computer Graphics and Applications. Proceedings. 12th Pacific Conference on*, 10 2004, pp. 361–370.

[20] T. Weise, B. Leibe, and L. Van Gool, "Fast 3d scanning with automatic motion compensation," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, 6 2007, pp. 1–8.

[21] S. Zhang and P. Huang, "High-resolution, real-time 3d shape acquisition," in *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW '04. Conference on*, 6 2004, pp. 28–28.

[22] R. R. Garcia and A. Zakhor, "Consistent stereo-assisted absolute phase unwrapping methods for structured light systems," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 6, no. 5, pp. 411–424, 9 2012.

[23] ——, "Geometric calibration for a multi-camera-projector system," in *Applications of Computer Vision (WACV), 2013 IEEE Workshop on*, 1 2013, pp. 467–474.

[24] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the Fourth Eurographics Symposium on Geometry processing*, 2006.

[25] L. Kavan, , S. Collins, J. Žára, and C. O'Sullivan, "Skinning with dual quaternions," in *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, ser. I3D '07. ACM, 2007, pp. 39–46.

[26] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Proc. SPIE*, vol. 1611, 1992, pp. 586–606.

[27] D. Koks, *Explorations in mathematical physics*. Springer, 2006.

[28] http://www-video.eecs.berkeley.edu/research/4D_SL/reference.mp4.

[29] http://www-video.eecs.berkeley.edu/research/4D_SL/rotate.mp4.

[30] 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)*, vol. 28, no. 5, December 2009.

Ricardo R. Garcia received his B.S. in electrical engineering from The University of Texas at Austin in 2007. He went on to complete his M.S in electrical engineering at the University of California, Berkeley in 2009 and his Ph.D. in 2014. His research interests are in computer vision with an emphasis in structured light systems and dynamic geometry capture.

Avideh Zakhor joined the faculty at UC Berkeley in 1988 where she is currently a professor and holds Qualcomm chair in Electrical Engineering and Computer Sciences. Her areas of interest include theories and applications of signal, image and video processing, 3D computer vision, and multimedia networking. She has won a number of best paper awards, including the IEEE Signal Processing Society in 1997 and 2009, IEEE Circuits and Systems Society in 1997 and 1999, international conference on image processing in 1999, Packet Video Workshop in 2002, and IEEE Workshop on Multimodal Sentient Computing in 2007. She holds 6 U.S. patents, and is the co-author of three books with her students. Prof. Zakhor received her B.S. degree from California Institute of Technology, Pasadena, and her S.M. and Ph. D. degrees from Massachusetts Institute of Technology, Cambridge, all in electrical engineering, in 1983, 1985, and 1987 respectively. She was a General Motors scholar from 1982 to 1983, was a Hertz fellow from 1984 to 1988, received the Presidential Young Investigators (PYI) award, and Office of Naval Research (ONR) young investigator award in 1992. In 2001, she was elected as IEEE fellow and received the Okawa Prize in 2004. She co-founded OPC technology in 1996, which was later by Mentor Graphics (Nasdaq: MENT) in 1998, Truvideo in 2000, and UrbanScan Inc. in 2005 which was acquired by Google in 2007.