User-Specific Hand Modeling from Monocular Depth Sequences

Jonathan Taylor[†], Richard Stebbing[†‡], Varun Ramakrishna[†♭], Cem Keskin[†],
Jamie Shotton[†] , Shahram Izadi[†], Aaron Hertzmann[⋆], Andrew Fitzgibbon[†]

Microsoft Research[†] University of Oxford[‡] Carnegie Mellon University[♭] Adobe[⋆]

Abstract

This paper presents a method for acquiring dense nonrigid shape and deformation from a single monocular depth sensor. We focus on modeling the human hand, and assume that a single rough template model is available. We combine and extend existing work on model-based tracking, subdivision surface fitting, and mesh deformation to acquire detailed hand models from as few as 15 frames of depth data. We propose an objective that measures the error of fit between each sampled data point and a continuous model surface defined by a rigged control mesh, and uses as-rigid-as-possible (ARAP) regularizers to cleanly separate the model and template geometries. *A key contribution is our use of a smooth model based on subdivision surfaces that allows simultaneous optimization over both correspondences and model parameters. This avoids the use of iterated closest point (ICP) algorithms which often lead to slow convergence. Automatic initialization is obtained using a regression forest trained to infer approximate correspondences. Experiments show that the resulting meshes model the user's hand shape more accurately than just adapting the shape parameters of the skeleton, and that the retargeted skeleton accurately models the user's articulations. We investigate the effect of various modeling choices, and show the benefits of using subdivision surfaces and ARAP regularization.*

1. Introduction

The acquisition of detailed models of the 3D world has long been a goal of computer vision, both as a scientific goal and for the applications that such models enable. For a rigid scene, techniques are highly advanced, allowing detailed models to be obtained even from impoverished 2D sensors [20]. However, the capture of nonrigid objects remains challenging. Existing techniques commonly use multiple near-synchronized cameras, but this is expensive and impractical, and so the ability to work from a single sensor is highly desirable. With a single sensor, the ability to capture the nonrigid world is considerably inferior to the rigid case, even when using a $2\frac{1}{2}$D depth sensor such as the Kinect.

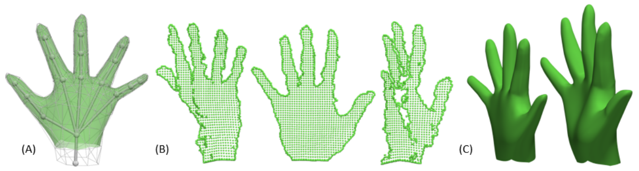


Figure 1: **Hand shape fitting.** (A) Template model with skeleton. (B) Three frames from 15-frame input sequence. (C) Template (left) and refined model (right) fit to this sequence. Note both a large scale change and local deformations (*e.g.* on the palm).

This paper presents a method for nonrigid 3D model acquisition from monocular depth sequences. We consider the important special case where a rigged template mesh of the object class is available. In particular, we focus on reconstruction of the human hand. Such rigged templates are relatively easy to construct: the template we use in our experiments, a 452-vertex mesh with a 21-bone skeleton, was created by two novice users of the 3D modeling package Blender in under a day. Importantly, the model and skeleton *geometry* can be approximate, the key information being the vertex and skeleton connectivity information. Such models are also available online for a variety of object classes. For example, the website TurboSquid lists nearly 4000 rigged 3D models at the time of writing.

The hand is an interesting object class for a number of reasons. First, today's real time sensors are only capable of sparse sampling of a hand due to their size. Second, although an important object class, it appears that no high quality globally-parameterized shape model exists for the hand; the hands on full-body models [3, 10] used in current vision systems tend to be clenched fists. Finally, hands are generally unclothed, particularly in indoor applications, so the benefits of having a user-specific model are perhaps even greater for hands than for the full body. Indeed, Ballan *et al.* [4] demonstrate that extremely robust tracking is possible given a user-specialized hand template, but require manual rigging and a multi-camera capture setup.

The primary contributions of our work are as follows.

(1) We propose a new objective function for specialization of a coarse rigged template to sparse noisy depth images. This objective is parameterized by a global 'core' model, a per-image model pose, and the set of correspondences between model and data. In contrast to previous work using polygon meshes, our model exploits subdivision surfaces to define a smooth continuous model. (2) Because of the smooth surface model, *simultaneous* optimization of both the model parameters and the correspondences is possible using standard nonlinear minimization. In contrast to ICP-like alternation strategies, this can offer rapid quadratic convergence (see Fig. 2b).

1.1. Related work

We consider only work dealing with temporal sequences, as the case of multiple simultaneous captures is equivalent to the rigid case.

Some systems operate in a more general regime than ours, where there is no requirement for a model template. A key component of such systems is nonrigid registration between 3D datasets. Wand *et al.* [29] demonstrate impressive results on noisy range sequences, matching an evolving core model to incoming range images. They use alternating minimization to optimize an energy defined over the core, a motion field from the core to each frame, regularizers encouraging isometry of the motion field, and core-to-frame correspondences. Li *et al.* [16] register pairs of range images by minimizing an energy similar to ours, and like us they simultaneously optimize correspondences and the shape parameters. However, these techniques are only demonstrated on scenes with relatively small deformations [17], or on the 'frontal-only' registration task, while our work allows full orbits of the shape. Liao *et al.* [18] recover general 3D models from single depth image sequences. However, their technique requires a relatively large number of point correspondences to be obtained from SIFT matching on simultaneously acquired RGB data of highly textured surfaces. The method also requires dense clean depth data, meaning that dynamic sequences must be captured using 'stop-motion' techniques.

Other systems are more specialized than ours and learn a low dimensional parameterization of a shape class. Examples include morphable models and their successors [5, 2, 3, 8], which we shall call parameterized class models (PCMs). Having built a PCM (or having simultaneously discovered alignment and PCM [10, 6]), a wide variety of impressive applications are enabled, such as the fitting of accurate 3D body models to a small number of range scans [30], or accurate hand or body tracking [7, 9]. However, the construction of PCMs is difficult, and indeed we know of no such model for the important class of hands. As an aside, the fact that our model produces aligned meshes may allow it to be used to construct a PCM, although this is future work.

With slightly more general assumptions than ours, Stoll *et al.* [24] require only an approximate template, without rigging information. They use a sparse set of point correspondences and minimize a Laplacian mesh energy to align the template into position on a target point cloud, such as might be obtained from a single range image. An ICP-style iteration then aligns the mesh to the remainder of the data. The system requires rather more correspondences than ours, and does not appear as amenable to automation. Li *et al.* [15] adapt the general-purpose system [16] to template adaptation and detail enhancement to produce impressive shape recovery given reasonably high-resolution and largely-frontal input sequences. Cashman *et al.* [6] also use an unrigged template, and simultaneously discover and fit a subdivision-surface PCM, fitting to single silhouettes of different object instances. However, they do not handle articulated models. One of our contributions is to augment their method with skeleton rigging, and to replace the PCM with ARAP-based regularization.

Some systems do consider a specialization of a *rigged* template. The system of Rhee *et al.* [21], however, requires a single RGB image to be taken in a highly constrained setting and cannot infer geometry for the entire hand. In contrast, Straka *et al.* [25] demonstrate results using dense 3D data from a multi-camera system. Although their system varies the bone lengths of a 'differential LBS' model, akin to the STBS model of [13], no provision for adaptation of the base mesh is considered.

1.2. Preliminaries

Before detailing our method, it will be useful to briefly describe the key ingredients used in this work: meshes, mesh rigging, surfaces and mesh deformation measures.

Meshes. A mesh is defined by a collection of M vertices $\{\mathbf{v}_m\}_{m=1}^{M} \subseteq \mathbb{R}^3$ and a *triangulation* consisting of a set of triples of vertex indices defining the model topology. We will typically refer to the triangulation only through the set of neighbors $\mathcal{N}_m \subset \{1..M\}$ of vertex m. As all of the meshes in this paper share the same triangulation, it will be assumed implicit in the following, so that a mesh is simply expressed by a $3 \times M$ matrix $V = [\mathbf{v}_1, ..., \mathbf{v}_M] \in \mathbb{R}^{3\times M}$. In this work, we seek to recover a rigged user-specific mesh model V_{core} by personalizing a rigged template hand mesh model V_{template}.

Mesh rigging. A rigged mesh $V \in \mathbb{R}^{3\times M}$ requires the definition of a function $\mathcal{P}$ which takes a set of *pose parameters* θ and *shape parameters* κ and transforms the mesh to generate a "posed" mesh $P = \mathcal{P}(V; \theta, \kappa) \in \mathbb{R}^{3\times M}$. We assume that such a function is provided to manipulate the template V_{template}, and thus it can also be applied to V_{core}. In our method's most general form, we only require that the function be differentiable in all of its parameters, but in this work we will use a common linear blend skinning (LBS) approach to define $\mathcal{P}$.

To this end, we use a skeleton of B bones, each of which is associated with a transformation $G_b(\theta,\kappa)$ mapping the bone's local coordinate system into the world coordinate system. The bones are arranged in a tree structure as seen in Figure 1. The transform $G_b(\theta,\kappa)$ applies first a rotation, followed by a translation and finally, if b is not the root, the recursive application of $G_{\pi(b)}(\theta,\kappa)$ where $\pi(b)$ is the parent bone of b. The rotation is a function of θ and parameterizes the orientation of bone b whereas the translation is dependent on κ and specifies the point of attachment for any children bones.

The m^{th} model vertex is skinned to all bones by a set of fixed skinning weights $\alpha_{m1}, ..., \alpha_{mB}$, where typically only a few weights are nonzero. Then $P = [\mathbf{p}_1, ..., \mathbf{p}_M] = \mathcal{P}(V;\theta,\kappa) \in \mathbb{R}^{3\times M}$ is a posed mesh whose m^{th} vertex is

$$\mathbf{p}_m = G_{\text{glob}}(\theta,\kappa) * \sum_{b=1}^{B} \alpha_{mb} G_b(\theta,\kappa) * G_b(\theta^0,\kappa)^{-1} * \mathbf{v}_m \;.$$

Here θ^0 represents the 'base pose' of the model, G_{glob} encodes a global similarity transform including scale and $A * \mathbf{x}$ denotes the transformation of vector $\mathbf{x}$ by the transformation A. Again, the precise structure of $\mathcal{P}$ is not particularly important other than to note that its derivatives are smooth and relatively cheap to compute.

Surface Function. A *surface* is a subset of $\mathbb{R}^3$, which we shall assume is defined by a *mapping* $\mathcal{S}$ from an essentially 2D space Ω to $\mathbb{R}^3$, defining the set of 3D points $\mathcal{S}(\Omega) = \{\mathcal{S}(\mathbf{u}) : \mathbf{u} \in \Omega\}$. When the surface is *parameterized* by a set of control vertices V, for example a polyhedron or subdivision surface, we write the dependence explicitly, defining the mapping (or *surface function*)

$$\mathcal{S} : \Omega \times \mathbb{R}^{3\times M} \mapsto \mathbb{R}^3 \;. \tag{1}$$

For triangular meshes, a parameter-space point $\mathbf{u} \in \Omega$ (§2.3) can be concretely expressed with a triplet (u, v, τ) where (u, v) is a coordinate in the triangle τ. The surface function $\mathcal{S}$ can then use this coordinate to linearly interpolate between the three vertices of τ, yielding a piecewise planar surface. We, however, require a smooth surface function $\mathcal{S}(\mathbf{u}; V)$ and an associated smooth surface normal function $\mathcal{S}^{\perp}(\mathbf{u}; V)$ defining the surface normal at $\mathbf{u}$. We employ a Loop subdivision surface as recently employed in modeling from silhouettes [6]. The essential property of this surface is that $\mathcal{S}$ is smooth, and is a simple polynomial function (see supplementary material) of $\mathbf{u}$ (and is linear in V) in each triangle τ. Note that even the linearity in V is not important to express here: the nonlinear optimization (§2.3) will exploit it naturally without the need for explicit special cases. What is important, however, is to have access to the derivatives $\partial\mathcal{S}/\partial\mathbf{u}$ and $\partial\mathcal{S}/\partial V$, which is ensured by the parameterization we use in this work.

Mesh Deformation Measure. As will be shown, our method requires a measure of mesh deformation between two meshes $V = [\mathbf{v}_1, ..., \mathbf{v}_M]$ and $W = [\mathbf{w}_1, ..., \mathbf{w}_M]$. A desirable property of such a measure is that rigid transformations are not penalized, and locally nonrigid transformations (*e.g.*, the bending of a finger) are penalized less than large deformations. The as-rigid-as-possible (ARAP) measure [22, 11] is defined as

$$D(V, W) = \sum_{m=1}^{M} \min_R \sum_{n\in\mathcal{N}_m} \|(\mathbf{v}_m - \mathbf{v}_n) - R(\mathbf{w}_m - \mathbf{w}_n)\|^2$$

where the inner minimizations are over 3D rotations $R \in SO^3$. Note that the definition does not specify how the optimization is to be performed. In most papers using ARAP, direct methods are employed to perform the inner minimization given V and W, but these methods greatly complicate the computation of derivatives such as $\partial D(V, U)/\partial V$. Thus for our purposes an alternative will be required (§2.2). Several related deformations have been defined, including Laplacian-based measures [23] and 'embedded deformation' [26], and our choice of ARAP is to some extent arbitrary, but exploration of the alternatives is left to future work.

2. Method

The input to our system is a collection of F depth frames. In frame f we observe N^f data points $\{\mathbf{x}_n^f\}_{n=1}^{N^f} \subseteq \mathbb{R}^3$, to which we associate estimated normals $\{\mathbf{n}_n^f\}_{n=1}^{N^f} \subseteq \mathbb{R}^3$.

2.1. Objective function

Our algorithm is expressed as energy minimization over all the unknown quantities in the system, divided into two sets: 'primary' and 'latent'. These are illustrated in Figure 2a. The primary unknowns are:

- The **core mesh** V_{core}, a $3 \times M$ matrix.
- The vector of core **shape parameters** κ.
- The per-frame **pose parameters** $\{\theta^f\}_{f=1}^F$, each a vector describing the pose of the hand.
- The **per-view instance meshes** $\{V_{\text{instance}}^f\}_{f=1}^F$, each a $3 \times M$ matrix. Explicitly parameterizing these instance meshes allows shape changes unmodeled by the mesh skinning to be cleanly handled.

The overall energy, in terms of these unknowns, is the sum of a data term and two regularizers:

$$\begin{aligned} E = &\sum_{f=1}^{F}\sum_{n=1}^{N^f} E_{\text{data}}(V_{\text{instance}}^f, \mathbf{x}_n^f, \mathbf{n}_n^f) + \\ &+ \lambda_{\text{core}} D(V_{\text{template}}, V_{\text{core}}) + \\ &+ \lambda_{\text{inst}} \sum_{f=1}^{F} D(\mathcal{P}(V_{\text{core}}; \theta^f, \kappa), V_{\text{instance}}^f) \;. \end{aligned} \tag{2}$$

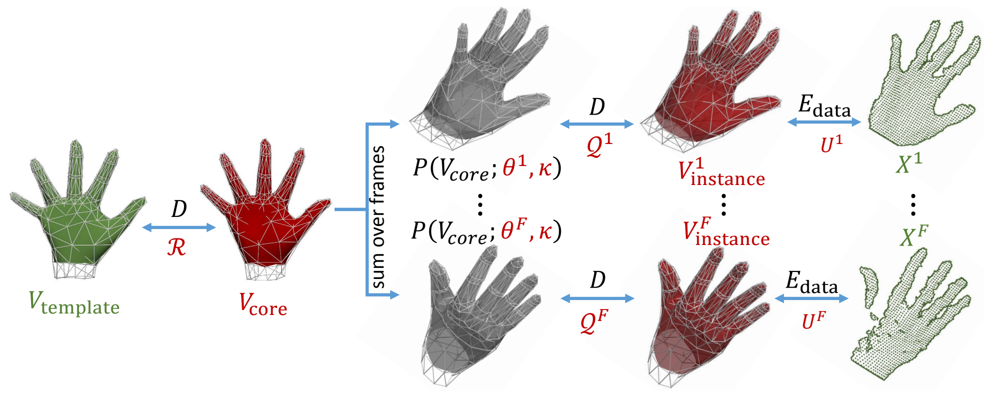


Figure 2a: **Model overview**. This figure depicts all the variables and energy terms in our model-fitting objective. Variables in red are optimized over, values in green are given. The grey meshes are the core warped by the per-frame pose parameters θ^f, and are not explicitly represented, but are computed when the energy and Jacobian are evaluated. The data-to-model correspondences U^f and the ARAP rotations $\mathcal{R}, \{Q^f\}_{f=1}^F$ are explicitly represented in our algorithm, allowing simultaneous optimization over all variables while keeping computational complexity linear in the number of frames.

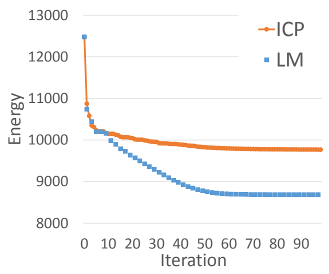


Figure 2b: **Levenberg-Marquardt (LM) vs. ICP.** A standard (ICP) like algorithm that alternates between discrete updates of $\mathcal{U}$ and continuous optimization of all other parameters (ICP) converges extremely slowly, whereas simultaneous optimization of all parameters (LM) converges rapidly.

The data term is a sum over all data points and their associated normals, and is defined independently for each point as follows:

$$E_{\text{data}}(V, \mathbf{x}_n^f, \mathbf{n}_n^f) = \min_{\mathbf{u}\in\Omega} \rho_{\text{data}} \begin{bmatrix} \mathbf{x}_n^f - \mathcal{S}(\mathbf{u}; V) \\ 1 - (\mathbf{n}_n^f)^\top \mathcal{S}^\perp(\mathbf{u}; V) \end{bmatrix}$$

where $\rho_{\text{data}}[\boldsymbol{\delta}_x; \delta_n] = \rho(\|\boldsymbol{\delta}_x\|; \sigma_x) + \lambda_{\text{norm}}\delta_n^2$ is the sum of a robust kernel applied to the point position error and squared normal error. In this work, we use the Cauchy kernel $\rho(t;\sigma) = \sigma^2 \log(1 + t^2/\sigma^2)$. The regularizers in (2) control the deviation of the personalized model V_{core} from the template, and the deviation of the per-frame meshes from the per-frame-posed personalized model. In practice, additional regularizers are added to (2), serving mainly to anneal the optimization, but we will defer their introduction until §2.4 as they do not complicate the overall approach described below.

2.2. Lifting the energy

Ultimately, we will minimize E using a Levenberg-Marquardt (LM) based optimizer [1], with initial estimates provided by a discriminative model [27]. However, as written, E is not in the sum-of-squares form required by LM due to a variety of inner minimizations. Our approach will therefore be to introduce a "lifted" energy $\hat{E}$, of the required form, that is defined over an additional set of *latent parameters* and whose minimum coincides with that of E.

To illustrate this technique, let us first look at a simpler energy

$$H(V) = \sum_{n=1}^{N} \min_{\mathbf{u}} \|\mathbf{x}_n - \mathcal{S}(\mathbf{u}; V)\|^2 + D(V, V_0) \qquad (3)$$

which corresponds to deforming a mesh V to fit a set $\{\mathbf{x}_n\}_{n=1}^N$ of data points while penalizing (in the ARAP sense) deformations from some fixed mesh V_0. We address the data term by introducing a set of latent variables $\mathcal{U} = \{\mathbf{u}_n\}_{n=1}^N$. By replacing $\mathbf{u}$ in the n^{th} minimization with $\mathbf{u}_n$, the minimization can be passed through the sum and we have that

$$\sum_{n=1}^{N} \min_{\mathbf{u}} \|\mathbf{x}_n - \mathcal{S}(\mathbf{u}; V)\|^2 = \min_{\mathcal{U}} \sum_{n=1}^{N} \|\mathbf{x}_n - \mathcal{S}(\mathbf{u}_n; V)\|^2 .$$

The second term can be approached in the same way by introducing the latent variables $\mathcal{R} = \{R_m\}_{m=1}^M \subseteq SO^3$ and noting that $D(V, W) = \min_{\mathcal{R}} \hat{D}(V, W; \mathcal{R})$ where

$$\hat{D}(V, W; \mathcal{R}) = \sum_{m=1}^{M} \sum_{n\in\mathcal{N}_m} \|(\mathbf{v}_m - \mathbf{v}_n) - R_m(\mathbf{w}_m - \mathbf{w}_n)\|^2. \qquad (4)$$

This allows us to rewrite (3) as

$$H(V) = \min_{V,\mathcal{R},\mathcal{U}} \hat{H}(V, \mathcal{R}, \mathcal{U}) \qquad (5)$$

with

$$\hat{H}(V, \mathcal{R}, \mathcal{U}) = \sum_{n=1}^{N} \|\mathbf{x}_n - \mathcal{S}(\mathbf{u}_n; V)\|^2 + \hat{D}(V, V_0; \mathcal{R}) .$$

The lifted energy $\hat{H}$ is now of the desired sum-of-squares form and has a minimum coinciding with that of H. Although we have increased the number of parameters, perhaps greatly, the relative independence of the terms in $\mathcal{R}$ and U mean that the normal equations (7) have large block-diagonal subblocks, and thus may be solved in time linear in the number of variables in $\mathcal{R} \cup U$.

Returning to (2), we follow the same approach of explicitly representing the the ARAP rotations $\mathcal{R}$ from template to core and the ARAP rotations $\mathcal{Q}^f$ for the instance mesh in frame f. The full set of correspondences over all data points in all frames is the set $\mathcal{U} = \bigcup_{f=1}^F \mathcal{U}^f$ for $\mathcal{U}^f = \{\mathbf{u}_n^f\}_{n=1}^{N^f}$. Together, these variables comprise the latent parameters, in terms of which the *lifted energy* becomes

$$\begin{aligned}\hat{E} = &\sum_{f=1}^{F}\sum_{n=1}^{N^f} \rho_{\text{data}} \begin{bmatrix} \mathbf{x}_n^f - \mathcal{S}(\mathbf{u}_n^f; V) \\ 1 - (\mathbf{n}_n^f)^\top \mathcal{S}^\perp(\mathbf{u}_n^f; V) \end{bmatrix} + \\ &+ \lambda_{\text{core}} \hat{D}(V_{\text{template}}, V_{\text{core}}; \mathcal{R}) + \\ &+ \lambda_{\text{inst}} \sum_{f=1}^{F} \hat{D}(\mathcal{P}(V_{\text{core}}; \theta^f, \kappa), V_{\text{instance}}^f; \mathcal{Q}^f)\ .\end{aligned} \tag{6}$$

Listing all arguments, we then seek to minimize this lifted energy function $\hat{E}(\mathcal{R}, V_{\text{core}}, \kappa, \{\theta^f, V_{\text{instance}}^f, \mathcal{Q}^f, \mathcal{U}^f\}_{f=1}^F)$.

2.3. Implementation

The lifted function $\hat{E}$ is now representable as a sum of squared residuals in an LM solver[1], and given code to compute $\mathcal{P}$, $\mathcal{S}$, $\mathcal{S}^\perp$, and their derivatives, we need not further know the details of the skinning method or subdivison surface model. However, some implementation details are worth investigating. Taking a high-level view of the issues, let us vectorize the set of correspondences $\mathcal{U}$ into the vector $\mathbf{u}$ and all other parameters into the vector $\mathfrak{s}$. The LM algorithm will proceed by first computing the Jacobian $J = [\partial\mathbf{r}/\partial\mathbf{u} | \partial\mathbf{r}/\partial\mathfrak{s}]$ where $\mathbf{r}$ is the vector of residuals such that $\hat{E} = \|\mathbf{r}([\mathbf{u}; \mathfrak{s}])\|^2$. A step is then computed by solving the system

$$(J^\top J + \mu I) \begin{bmatrix} \delta\mathbf{u} \\ \delta\mathfrak{s} \end{bmatrix} = -J^\top \mathbf{r} \tag{7}$$

where μ is standard adaptation parameter varied by the algorithm. The update $[\mathbf{u} + \delta\mathbf{u}; \mathfrak{s} + \delta\mathfrak{s}]$ is accepted if the resulting energy is reduced.

Two apparent difficulties arise with using subdivison surfaces in this framework. First is that near so-called extraordinary vertices (EVs), the derivatives may vanish, potentially introducing saddle points which could stall optimization. This difficulty is avoided by replacing the surface in a small region around each EV with approximating quartic Bezier triangles. Similar approximations have been performed for Catmull-Clark subdivision surfaces using bicubic B-splines [19]. While the resulting surface is no longer $\mathcal{C}^1$ continuous between extraordinary patches, the discontinuities are minor and neglible in practice.

The second difficulty is that the parameterization of a correspondence $\mathbf{u}$ is unusual, in being a tuple (u, v, τ). Perhaps surprisingly, even with these unusual parameters there is no problem in computing $\delta\mathbf{u}$, the 2D sub-vector of $\mathbf{u}$ corresponding to $\mathbf{u}$. It will always be defined in the current triangle τ, but the difficulty comes in applying the update $\mathbf{u} + \delta\mathbf{u}$. To effect such updates, we follow the scheme of [6], which walks across the triangle mesh applying the update piecewise (see Figure 3). At each triangle boundary the unused update is re-expressed in the adjacent triangle, maintaining tangent continuity on the limit surface, and the remainder of the update is recursively applied. Again, although the bookkeeping is messy, the procedure is well encapsulated once implemented. In terms of computational complexity, we find that we rarely have to traverse more than three triangles per point, and this procedure is not a computational bottleneck.

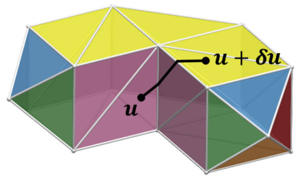


Figure 3: Illustration of mesh walking to apply Levenberg-Marquardt update $\delta\mathbf{u}$.

One more detail is that the scales of the parameters should be chosen to be roughly commensurate, and the solver options should be set to disable attempts to automatically scale parameters by inspecting the columns of J. In our experience, these heuristics significantly hurt performance.

2.4. Additional Regularizers

The regularizers mentioned above are now described. We emphasize that despite the apparently large number of tuning parameters this introduces, most have effect only on convergence properties, rather than on the final shape. The temporal prior on pose is

$$E_{\text{motion}} = \lambda_{\text{motion}} \sum_{f=1}^{F-1} \|\gamma(\theta^{(f+1)}) - \gamma(\theta^f)\|^2$$

where $\gamma(\theta)$ extracts all the joint-local rotations as 3D exponential map parameters and the 3D root bone translation from the pose parameters θ into a single $3(B+1)$-element vector. Similarly the base-pose prior is

$$E_{\text{rigid}} = \lambda_{\text{rigid}} \sum_{f=1}^{F} \|\gamma^-(\theta^f) - \gamma^-(\theta^0)\|^2 \tag{8}$$

where $\gamma^-(\theta)$ contains the $3B$ rotational elements of $\gamma(\theta)$. In order to keep the posed core and instance meshes in the same position, a very low-weighted term $E_{\text{L2}} = \lambda_{\text{L2}} \sum_{f=1}^F \|\mathcal{P}(V_{\text{core}}; \theta^f, \kappa) - V_{\text{instance}}^f\|_F^2$ penalizes squared differences between the vertices in these meshes. Instead of explicitly penalizing intersections with the background, we also find it helpful to simply discourage large scalings of the core geometry using $E_{\text{scale}} = \lambda_{\text{scale}} \zeta(G_{\text{glob}}(\kappa))^2$. where $\zeta(G_{\text{glob}})$ extracts the global scale parameter from transformation G_{glob}.

To motivate a final regularizer, consider the function $l_b(\kappa) = G(\theta^0, \kappa) * [0, 0, 0]^T$ that extracts the position of

[1]Note that the robust kernel can be dealt with by taking the square root or employing a robust LM-variant [28, 1].

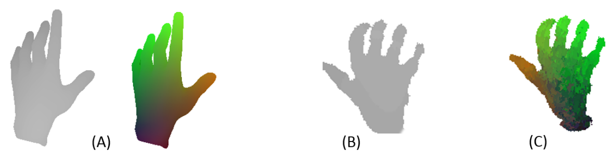


Figure 4: *Vitruvian Manifold correspondences*. (A) A (depth, correspondence) pair from the training set. (B) Test depth image. (C) Predicted correspondence labels.

bone b in the base pose θ^0 under shape κ. We would like to encourage these positions in the core to remain consistent relative to a set of nearby vertices (typically a vertex ring) in the template. We do this by denoting the indices of these vertices with the set $C_b \subseteq \{1, ..., M\}$ and employ an ARAP deformation penalty

$$D_b^\dagger(\kappa, V_{\text{core}}) = \sum_{m \in C_b} \min_R \|(l_b(\kappa^0) - \mathbf{v}_m) - R((l_b(\kappa) - \mathbf{w}_m)\|^2$$

where $\mathbf{v}_m$ and $\mathbf{w}_m$ denote the m^{th} vertex in the template and core mesh respectively and κ^0 is the template shape parameters. We then add the term

$$E_{\text{skeleton}} = \lambda_{\text{skeleton}} \sum_{b=1}^{B} D_b^\dagger(\kappa, V_{\text{core}}) \tag{9}$$

to the energy and again lift the rotations out of the inner minimization using latent variables for optimization.

2.5. Initialization

As with any nonlinear optimization (ICP or LM), a sensible initialization is important to ensure convergence to a good optimum. In our energy, the important quantities to initialize well are the correspondences $\mathcal{U}$. If a reasonable proportion of these are roughly on the correct hand part, convergence of the remaining parameters is typically reliable. Note that this is despite the apparent nonlinearity of the energy: lifting the parameters to $\hat{E}$ appears to greatly simplify the optimization surface.

To achieve this, we adapt the 'Vitruvian Manifold' method of [27] to directly predict correspondences from the input image. We train a decision forest classifier on a training set of synthetically rendered (depth image, correspondence map) pairs, and then at runtime the decision forest is applied independently at every pixel $\mathbf{x}_f^n$ to generate an initial estimate for $\mathbf{u}_f^n$. To cope with front-back ambiguities we train a two-stage model [14], where the first stage performs binary classification on the entire image to detect whether the hand is front or back facing, and then the standard method is applied, using trees trained with front or back data as appropriate. Figure 4 illustrates the process.

2.6. Optimization

We optimize the energy by interleaving LM optimization on *all* parameters and global closest-point like updates for $\mathcal{U}$. Note that this is *not* the same as alternating between global optimization on all non-correspondence parameters and closest-point like updates. Indeed, we initially tried that strategy (see Figure 2b) but it is was prohibitively slow compared to performing simultaneous optimization. With simultaneous optimization the optimizer, for example, has the opporunity to take much larger steps in pose space to achieve lower energies as the correspondences are free to slide along the model surface. This approach is made possible by the smoothness of the subdivision surface model, emphasizing the benefit of incorporating such a model.

The global optimization follows a simple annealing schedule, where the various regularizations are initially set high (especially temporal smoothness of pose and global rotation and translation), and 20 LM iterations are performed, with global $\mathcal{U}$ updates every 5 iterations. Then the regularizations are set to their desired values, and up to 50 more LM iterations are performed with global updates every 10 iterations. Finally, up to 50 more LM iterations are performed. For a 200K-variable problem, each LM iteration takes a few tens of seconds and the entire procedure takes roughly 10 minutes.

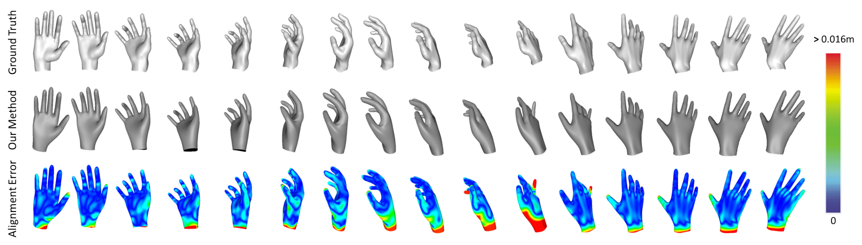
3. Evaluation

We performed quantitative and qualitative evaluation of our approach on several sequences. We use synthetic renders and 3D printed models to quantify the method. Results on Kinect depth images further validate the approach on real, noisy input data. We also explore various parameters of our model.

Synthetic input sequences. We created three synthetic hand sequences using Poser.[2] The hand shape was manually edited to be different in each sequence, resulting in 'small', 'medium' and 'large' hand sequences. Poser interpolated smoothly between a set of manually created key poses, again different for each sequence, though a roughly similar global rotation is applied across the sequences so that the hand rotates around the vertical axis. Note that the Poser hand model has a quite different mesh topology and rigging to the template we use for fitting. Fig. 5 shows qualitative and quantitative results of our technique, showing an overall accuracy of about 3mm RMS distance from the ground truth. The peaks in error around frame 10 are due to occlusion in side views as the hands pass through 90° in their global rotation.

3D printed model. To further validate our method quantitatively, we 3D printed a hand model. For speed of printing

[2] http://poser.smithmicro.com/.



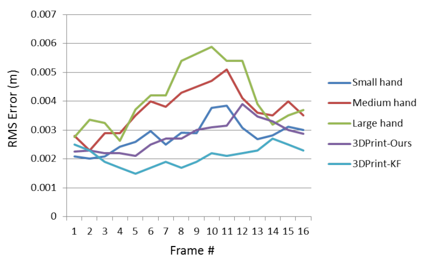


Figure 5: **(Left)** Ground truth comparison to one synthetic sequence. While occasional frames may misplace a joint, the simultaneous global optimization keeps such failures local. **(Right)** Ground truth comparisons: RMS *vs*. orientation. RMS error is plotted against frame number for both the synthetic and 3D printed sequences.

this was a small hand, roughly 70% of an average adult hand. We then captured a depth sequence using Kinect, and reconstructed the (rigid) hand from the sequence using both the rigid KinectFusion algorithm [12] and our nonrigid approach. Fig. 5 compares the result of the two methods across the sequence. Overall, KinectFusion achieved an RMS error of 2.0mm, while we were able to achieve a very comparable 2.5mm, despite our method allowing nonrigid deformations. We also note that the accuracy on the nonrigid small hand synthetic sequence is comparable to that on the rigid 3D printed sequence that contains a similar sized hand. Fig. 6 gives a qualitative comparisons between the systems. Note the low quality of the input data, given the small size of the hand and the low resolution of the sensor. Both methods thus struggle: KinectFusion tends to merge fingers together, while our approach appears to be stuck in a local minimum where certain correspondences from the middle finger have been assigned to the ring finger. We believe this to be a problem of the optimization rather than the energy.

Real data. As a final experiment, we captured real sequences of three very different hand shapes including adult male, child, and bulky winter glove. The hands articulate while rotating roughly 180° about the vertical axis from the palm to the backside. These varied hand shapes challenge our method to properly adapt the core mesh given the weak constraints provided by the noisy Kinect data. Despite, this we are able to recover models representative of this shape variation, although two of the fingers of the instance geometry become stuck in a few side-on frames of the child sequence due to missing data.

We do not have ground truth for these sequences, but present qualitative results. Fig. 7 shows the resulting core and instance geometries for the three sequences. Despite large variations in shape, our model is able to infer an accurate per-user template (core) for each sequence, and fit this reliably to noisy depth data.

In Fig. 8, we investigate varying the weights in the model to extreme settings to simulate the removal of aspects of the model. Setting λ_{core} to a large value has a strong effect on

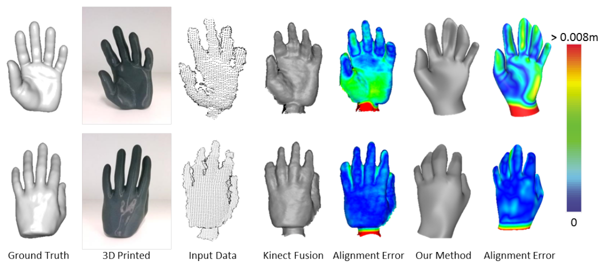


Figure 6: **Ground truth comparison to 3D-printed sequence**, compared with KinectFusion.

the resulting core user mesh: it can no longer deform away from the template, and as a result is no longer able to explain the data well. The effect of setting λ_{inst} to a large value is less pronounced.

4. Conclusions

We have presented a new technique for the acquisition of nonrigid scenes given only a rough model template. Our new optimization framework can reliably infer a user-specific hand mesh from a coarse rigged template using as few as 15 noisy frames from a commodity depth sensor as input. The use of subdivision surfaces to provide a smooth continuous surface model allows the *simultaneous* optimization of both the model parameters and the correspondences using standard nonlinear minimization. Results demonstrate both robust fitting to the observed noisy depth data, as well as support for pose variation across frames.

References

[1] S. Agarwal, K. Mierle, and Others. Ceres solver. `https://code.google.com/p/ceres-solver/`.

[2] B. Allen, B. Curless, and Z. Popović. The space of human body shapes: reconstruction and parameterization from range scans. *ACM Trans. Graphics*, 22(3), 2003.

[3] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. SCAPE: Shape completion and animation of people. *ACM Trans. Graphics*, 24(3), 2005.

[4] L. Ballan, A. Taneja, J. Gall, L. V. Gool, and M. Pollefeys. Motion capture of hands in action using discriminative salient points. In *Proc. ECCV*, 2012.

[5] V. Blanz and T. Vetter. A morphable model for the synthesis of 3D faces. In *Proc. SIGGRAPH*, 1999.

[6] T. Cashman and A. Fitzgibbon. What shape are dolphins? building 3D morphable models from 2D images. *IEEE Trans. PAMI*, 2013.

[7] M. de La Gorce, N. Paragios, and D. J. Fleet. Model-based hand tracking with texture, shading and self-occlusions. In *Proc. CVPR*, 2008.

[8] N. Hasler, C. Stoll, M. Sunkel, B. Rosenhahn, and H.-P. Seidel. A statistical model of human pose and body shape. *CGF*, 2009.

[9] T. Helten, A. Baak, G. Bharaj, M. Muller, H. Seidel, and C. Theobalt. Personalization and evaluation of a real-time depth-based full body tracker. In *Proc. 3DTV*, 2013.

[10] D. A. Hirshberg, M. Loper, E. Rachlin, and M. J. Black. Coregistration: Simultaneous alignment and modeling of articulated 3D shape. In *Proc. ECCV*. 2012.

[11] T. Igarashi, T. Moscovich, and J. F. Hughes. As-rigid-as-possible shape manipulation. *ACM Trans. Graphics*, 24(3), 2005.

[12] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proc. UIST*, 2011.

[13] A. Jacobson and O. Sorkine. Stretchable and twistable bones for skeletal shape deformation. *ACM Trans. Graphics*, 30(6), 2011.

[14] C. Keskin, F. Kiraç, Y. E. Kara, and L. Akarun. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *Proc. ECCV*, 2012.

[15] H. Li, B. Adams, L. J. Guibas, and M. Pauly. Robust single-view geometry and motion reconstruction. *ACM Trans. Graphics*, 2009.

[16] H. Li, R. W. Sumner, and M. Pauly. Global correspondence optimization for non-rigid registration of depth scans. *Proc. SGP*, 2008.

[17] H. Li, E. Vouga, A. Gudym, L.Luo, J. Barron, and G. Gusev. 3D self-portraits. *ACM Trans. Graphics*, 32(6), 2013.

[18] M. Liao, Q. Zhang, H. Wang, R. Yang, and M. Gong. Modeling deformable objects from a single depth camera. In *Proc. ICCV*, 2009.

[19] C. Loop and S. Schaefer. Approximating catmull-clark subdivision surfaces with bicubic patches. *ACM Trans. Graph.*, 27(1), 2008.

[20] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *Proc. CVPR*, 2011.

[21] T. Rhee, U. Neumann, and J. P. Lewis. Human hand modeling from surface anatomy. In *Proc. I3D*, 2006.

[22] O. Sorkine and M. Alexa. As-rigid-as-possible surface modeling. In *Proc. SGP*, 2007.

[23] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian Surface Editing. In *Proc. SGP*, 2004.

[24] C. Stoll, Z. Karni, C. Rössl, H. Yamauchi, and H.-P. Seidel. Template deformation for point cloud fitting. In *Proc. Eurographics*, 2006.

[25] M. Straka, S. Hauswiesner, M. Ruether, and H. Bischof. Simultaneous shape and pose adaption of articulated models using linear optimization. In *Proc. ECCV*, 2012.

[26] R. W. Sumner, J. Schmid, and M. Pauly. Embedded deformation for shape manipulation. *ACM Trans. Graphics*, 26(3), 2007.

[27] J. Taylor, J. Shotton, T. Sharp, and A. Fitzgibbon. The Vitruvian Manifold: Inferring dense correspondences for one-shot human pose estimation. In *Proc. CVPR*, 2012.

[28] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment—a modern synthesis. In *Vision algorithms: theory and practice*, pages 298–372. 2000.

[29] M. Wand, B. Adams, M. Ovsjanikov, A. Berner, M. Bokeloh, P. Jenke, L. Guibas, H.-P. Seidel, and A. Schilling. Efficient reconstruction of nonrigid shape and motion from real-time 3d scanner data. *ACM Trans. Graphics*, 2009.

[30] A. Weiss, D. Hirshberg, and M. J. Black. Home 3D body scans from noisy image and range data. In *Proc. ICCV*, 2011.

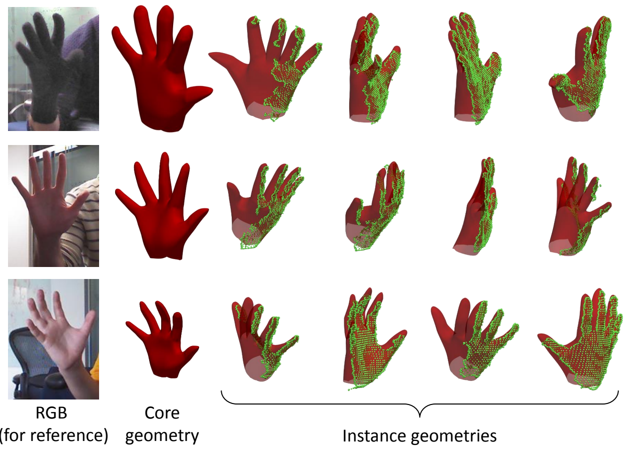


Figure 7: Results on three real image sequences with different hand geometries. **(Top)** A hand in a bulky winter glove. **(Middle)** An adult male hand. **(Bottom)** A child's hand. From left to right we show an RGB image (only for visual reference – not used by our method), the core geometry adapted to the subject's hand (with correct relative scale), and four side-on views of the instance surface (red) fitting the noisy data (green). Note an accurate shape estimate in the core geometry, and the ability to closely fit the observations in the instance geometries, despite noisy input data and nonrigid pose deformations.

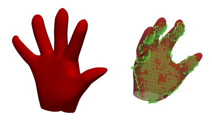
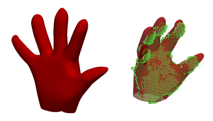
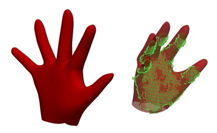
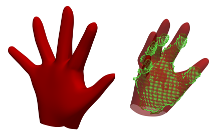
λ_{inst} \ λ_{core}	Normal	High
Normal		
High		

Figure 8: **Effect of terms in our energy.** We vary two parameters to simulate the removal of parts of the model. In each, the left shows the inferred core user mesh, and the right overlays the input data (green) on the inferred instance mesh in red. **(Top left)** Normal system. The bent-back index finger is part of the input – this sequence is of a gloved hand (see supplementary material). **(Bottom left)** the core is forced to match the template, so effectively fitting a scaled LBS skeleton with an ARAP data term. **(Top right)** the ARAP data term is highly weighted, so data points must effectively match directly to the posed mesh. For low-articulation frames this makes little difference, but at extreme articulations it may improve convergence. **(Bottom right)** simulating mesh adaptation [25].