The Phong Surface: Efficient 3D Model Fitting using Lifted Optimization

Jingjing Shen, Thomas J. Cashman, Qi Ye, Tim Hutton, Toby Sharp, Federica Bogo, Andrew Fitzgibbon, and Jamie Shotton

Microsoft Mixed Reality & AI Labs, Cambridge, UK
{jinshen, tcashman, yeqi, tihutt, tsharp, febogo, awf, jamiesho}@microsoft.com

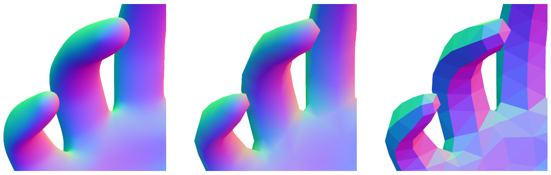
Abstract. Realtime perceptual and interaction capabilities in mixed reality require a range of 3D tracking problems to be solved at low latency on resource-constrained hardware such as head-mounted devices. Indeed, for devices such as HoloLens 2 where the CPU and GPU are left available for applications, multiple tracking subsystems are required to run on a continuous, real-time basis while sharing a single Digital Signal Processor. To solve model-fitting problems for HoloLens 2 hand tracking, where the computational budget is approximately 100 times smaller than an iPhone 7, we introduce a new surface model: the 'Phong surface'. Using ideas from computer graphics, the Phong surface describes the same 3D shape as a triangulated mesh model, but with continuous surface normals which enable the use of lifting-based optimization, providing significant efficiency gains over ICP-based methods. We show that Phong surfaces retain the convergence benefits of smoother surface models, while triangle meshes do not.

Keywords: model-fitting, optimization, hand tracking, pose estimation

1 Introduction

As computer vision systems are increasingly deployed on wearable or mobile computing platforms, they are required to operate with low power and limited computational resources. In this context, the problem of pose estimation (as applied, for example, to tracking hands [15,28], human bodies [2,17,31], or faces [10]) is often tackled with a hybrid architecture that combines discriminative machine-learnt models with generative model fitting to explain the observed data [12, 15, 27–29]. For the purposes of this paper, we define 'model fitting' as the registration of a 3D surface model to a point set observation. Model fitters can benefit from powerful priors learned from data, and recent work even shows the benefits of including model fitting in the training loop [9,30]. An optimizer with fast convergence is critical for building real-time systems that operate with low compute. However, the *correspondences* between the observed data and the model are often unknown, and need to be discovered in the course of the optimization.

Two main optimization alternatives have been proposed for solving this problem. **Iterative Closest Point (ICP)** algorithms [1,7,20,25] solve for model pose via 'block coordinate descent': first finding closest points on the model surface, and then fixing those correspondences while solving for model pose alone.



(a) Subdiv. surface (b) Phong surface (c) Triangular mesh

Surface type	eval.	with $\partial/\partial u$
Subdiv. surface	0.329s	1.241s
Phong surface	0.049s	0.279s
Triangular mesh	0.047s	0.196s

(d) Timings on 10^6 evaluations

Fig. 1: A hand model represented by a Loop subdivision surface [11] (a), a Phong surface (b) and a triangle mesh (c), with surface normals visualized by mapping x, y, z coordinates to red, green and blue components respectively. (d) shows timings in seconds on PC: *eval.* refers to evaluation of 10^6 surface point positions and normals, and *with* $\partial/\partial u$ includes the cost of derivative calculations w.r.t. surface coordinate as well.

The alternative approach is **'lifted' optimization**: to solve for correspondences and model pose *simultaneously*, using a lifted objective function that explicitly parametrizes the unknown correspondences. Taylor et al. [26,27] demonstrate hand tracking systems using this approach, and claim that the smoothness of the model surface is an important prerequisite for a smooth energy landscape that allows lifted optimization to converge efficiently. However, the complex surface representation they propose comes at a cost, entailing as much as 58% of the per-iteration model-fitting time [27, supplementary material].

In this paper, we show that much simpler surface representations are sufficient, thus making it cheaper for vision systems to access the convergence benefits of lifted optimization. In particular, since it is often beneficial to include surface orientation properties in the objective function, one might assume that a normal field with continuous first derivatives (second-order surface smoothness) is a necessary minimum for a gradient-based optimizer. This implies that a simple triangle mesh cannot suffice and we show this is true: a simple triangle mesh is insufficient, but it *is* sufficient to pair a triangle mesh with a normal field that is simply linearly interpolated over each triangle. We call the resulting representation a 'Phong surface', after the Phong shading [19] technique in computer graphics, which evaluates the lighting equation using similarly interpolated normals. Fig. 1 shows the normal field evaluated inside the optimizer for different surface representations, illustrating that the Phong surface (b) leads to surface evaluations that are a close approximation to evaluations on the smooth subdivision surface (a). Informally, our contribution to 3D model fitting is a surface representation which is designed to be "as simple as possible, but no simpler".

We evaluate Phong surfaces in comparison to two other model representations; the smooth subdivision surfaces used by prior work [4,26,27] and simple triangle meshes with a piecewise-constant normal field. Our experiments in rigid pose alignment and in the example application of hand tracking compare these alternative models in the context of different optimizers, and confirm earlier results that lifted optimization converges faster and with a wider basin of convergence than ICP [27].

We have successfully applied lifted optimization with Phong surfaces to implement a fully articulated hand tracker on HoloLens 2 [14], a self-contained head-mounted holographic computer. The tracker needs to run on an embedded Digital Signal Processor (DSP) with a compute budget of 4 GFLOPS, which is 100 times smaller than an iPhone 7, or 1000 times smaller than a high-end laptop. This application requires real-time continuous tracking under tight thermal and power constraints, motivating us to find the simplest possible computation that would allow the optimizer to converge to an accurate hand pose estimate. The Phong surface representation is one of the key innovations that make this hand tracker run in realtime on the HoloLens 2.

In summary, our contributions are that we

- introduce Phong surfaces by transferring the concept of Phong shading from computer graphics to model fitting in computer vision;
- show that Phong surfaces combine the convergence benefits of lifted methods with the computational cost of planar mesh models;
- demonstrate that fitting Phong surface models allows us to implement realtime hand tracking under a computational budget of 4 GFLOPS.

1.1 Related work

ICP has a long history in surface reconstruction and point set registration, starting from its first descriptions by Besl and McKay [1] and Chen and Medioni [5]. The simplicity of ICP means that it is easy to implement and was broadly adopted, spawning a host of variations on the two fundamental steps of closest point finding and error minimization, as described by Rusinkiewicz and Levoy [22]. Neugebauer [16] and Fitzgibbon [6] formulate ICP as an instance of a non-linear least squares problem, solved by general optimizers such as the Levenberg-Marquardt algorithm [13]. This opens up new possibilities for the objective function, such as allowing a robust kernel to be included directly in the objective as a way to smoothly handle outliers [6].

Meanwhile, ICP was generalized to articulated models by Pellegrini et al. [18], and demonstrated in this form for fitting models of human bodies [3, 32] and hands [25]. Another key advantage of ICP is the broad range of geometric representations that can be fitted: any point set or surface that could support the chosen 'closest point' query is included in a straightforward manner. However, a disadvantage of the alternating coordinate descent is that ICP algorithms *converge slowly*, reducing the objective only *linearly* as optimization proceeds [28]. This is particularly apparent when a model needs to slide relative to a data set, as this requires that the model's pose is updated in harmony with data correspondences. Point-to-plane ICP [5] attempts to address this for common cases, but thereby limits the set of objectives which can be minimized, losing the freedom introduced by Neugebauer and Fitzgibbon. Recently Rusinkiewicz [21] presents a symmetric objective function for ICP which is particularly suited to the original task of point set to point set alignment, but not of point set to parametric surface alignment, which is the focus of this paper. Note that we are explicitly in a non-symmetric scenario. An extension of [21] could be considered by sampling the model, but this would lose the structure that the model provides, and which is available in many real scenarios.

Several authors have attempted to address the shortcomings of ICP by estimating unknown correspondences simultaneously with model parameters. Sullivan and Ponce [24] present one of the first systems to do so in a model-fitting context, and Cashman and

Fitzgibbon [4] elaborate on this idea to also fit smooth surface models to silhouette constraints. Subsequently, Taylor et al. [26, 27] and Khamis et al. [8] demonstrate the benefits of lifted optimization in the area of articulated hand tracking. However, all of these methods required complicated smooth surface constructions, in stark contrast to the freedom available when using ICP. This paper addresses this disadvantage, by showing that lifted optimization is equally applicable to models with extremely simple geometry.

Another line of work performs model fitting without estimating any data correspondences at all. Taylor et al. [28] follow the same approach originally proposed by Fitzgibbon [6] by using articulated distance fields to find correspondences; this is conceptually similar to an ICP closest-point search, but can be implemented extremely efficiently using graphics hardware. Mueller et al. [15] use a discriminative network to perform dense correspondence regression, thus allowing the model-fitting stage to proceed under the assumption that the correspondences are fixed. Neither of these approaches are currently suitable for deployment on low-power devices.

2 Method

We describe model-fitting problems in a common framework with the following notation:

- A 3D surface model $S(\theta) \subset \mathbb{R}^3$ parameterized by a vector θ, which might for example specify rigid pose, shape variations or joint angles,
- A list of sampled data points $\{\mathbf{x}_i\}_{i=1}^D$ with estimated data normals $\{\mathbf{x}_i^\perp\}_{i=1}^D$,
- An objective function $E(\theta)$ that penalizes differences between the parameterized model and the observed data,
- An optimizer that iteratively updates the current hypothesis for θ to locally minimize the objective function.

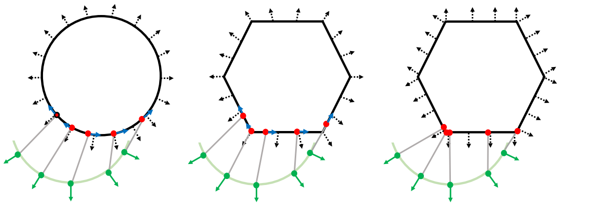
We follow the model fitting work of Taylor et al. [27] which optimizes a differentiable lifted objective function with a Levenberg optimizer. The smoothness of the subdivision surface model used in [27] encourages good convergence properties, but at the cost of expensive function evaluations for the surface positions, normals and their derivatives. We focus on efficiency and investigate the requirements for surface and normal smoothness in a lifted optimizer.

2.1 Phong surface model

The key idea is to generate surface normal vectors by linearly interpolating vertex normals over each planar triangle. This is the same approximation of a smooth surface used in the Phong shading method for computer graphics rendering [19], motivating our use of the ‘Phong surface’ moniker.

The surface model is defined by a triangle control mesh containing N control vertices each with a position and normal vector, $V(\theta) \in \mathbb{R}^{6\times N}$. The model also has a fixed triangulation of the vertices, where each triangle in the mesh corresponds to a parameterized triangular patch of the surface.

As illustrated in Fig. 3a, let $u = \{p, v, w\}$ be a surface coordinate where $p \in \mathbb{N}$ is the index of a triangular patch, and $v \in [0, 1], w \in [0, 1 - v]$ parameterize the unit triangle.



(a) Smooth surface (b) Phong surface (c) Triangular mesh

Fig. 2: Illustration of the cross section of a surface model (black) with normals (dashed black) close to some target data (green). With a continuous normal field, either from a smooth model (a) or from a Phong surface (b), the normal term forces shown in blue drive the red correspondences towards their correct location, improving convergence. These forces do not exist for a triangular mesh (c) where the normals are constant on the facet and so the derivatives are all zero.

$S(u,\theta) \in \mathbb{R}^3$ denotes the surface position and $S^\perp(u,\theta) \in \mathbb{R}^3$ the unit-length normal to the surface, both evaluated at the given coordinate u. Let $\boldsymbol{v}_1(\theta), \boldsymbol{v}_2(\theta), \boldsymbol{v}_3(\theta)$ be the control vertex positions and $\boldsymbol{v}_1^\perp(\theta), \boldsymbol{v}_2^\perp(\theta), \boldsymbol{v}_3^\perp(\theta)$ the control vertex normals of the pth triangular patch as specified by u, where $\boldsymbol{v}_i(\theta)$ and $\boldsymbol{v}_i^\perp(\theta)$ are determined by the pose and/or shape parameter vector θ. Then the Phong surface evaluation is defined simply as a linear interpolation of the control vertices:

$$S(u,\theta) = (1-v-w)\boldsymbol{v}_1(\theta) + v\boldsymbol{v}_2(\theta) + w\boldsymbol{v}_3(\theta), \tag{1}$$

$$\boldsymbol{c}(u,\theta) = (1-v-w)\boldsymbol{v}_1^\perp(\theta) + v\boldsymbol{v}_2^\perp(\theta) + w\boldsymbol{v}_3^\perp(\theta), \tag{2}$$

$$S^\perp(u,\theta) = \frac{\boldsymbol{c}(u,\theta)}{\|\boldsymbol{c}(u,\theta)\|}, \tag{3}$$

where $\boldsymbol{c}(u,\theta)$ is the interpolated normal direction vector.

We give the partial derivatives with respect to v, w and θ compactly in terms of the total differentials:

$$\begin{aligned}
\mathrm{d}S(u,\theta) &= \mathrm{d}v(\boldsymbol{v}_2 - \boldsymbol{v}_1) + \mathrm{d}w(\boldsymbol{v}_3 - \boldsymbol{v}_1) + (1-v-w)\mathrm{d}\boldsymbol{v}_1 + v\mathrm{d}\boldsymbol{v}_2 + w\mathrm{d}\boldsymbol{v}_3,\\
\mathrm{d}\boldsymbol{c}(u,\theta) &= \mathrm{d}v(\boldsymbol{v}_2^\perp - \boldsymbol{v}_1^\perp) + \mathrm{d}w(\boldsymbol{v}_3^\perp - \boldsymbol{v}_1^\perp) + (1-v-w)\mathrm{d}\boldsymbol{v}_1^\perp + v\mathrm{d}\boldsymbol{v}_2^\perp + w\mathrm{d}\boldsymbol{v}_3^\perp,\\
\mathrm{d}S^\perp(u,\theta) &= \frac{1}{\|\boldsymbol{c}\|}\left(I_3 - \frac{\boldsymbol{c}\boldsymbol{c}^T}{\|\boldsymbol{c}\|^2}\right)\mathrm{d}\boldsymbol{c}.
\end{aligned}$$

The partials can be read off from this notation as the coefficient of the relevant differential, e.g. $\frac{\partial S}{\partial v} = \boldsymbol{v}_2 - \boldsymbol{v}_1$ by setting $\mathrm{d}v = 1, \mathrm{d}w = 0, \mathrm{d}\boldsymbol{v}_i = \mathbf{0}, \mathrm{d}\boldsymbol{v}_i^\perp = \mathbf{0}$ in $\mathrm{d}S$.

2.2 Lifted optimization with the Phong surface

The objective function of the optimization in model fitting typically includes data terms and problem-dependent prior terms or regularization terms. Here we briefly describe the data terms.

We penalize the distance from each data point $\mathbf{x}_i$ to its corresponding surface point defined by the surface coordinate u_i, and the difference in surface orientation to the associated data normal $\mathbf{x}_i^\perp$, using

$$E(\theta, U) = \frac{1}{D}\sum_{i=1}^{D}\Big(\|S(u_i,\theta) - \mathbf{x}_i\|^2 + \lambda_n\|S^\perp(u_i,\theta) - \mathbf{x}_i^\perp\|^2\Big) \tag{4}$$

where D is the number of data points, λ_n is the contribution weight for normals, and $U = \{u_i\}_{i=1}^{D}$ are the surface coordinates corresponding to each data point.

Note that the surface coordinates U are optimized jointly with θ by the lifted optimizer, whereas the ICP optimizer alternates between updating θ and U. This means that the lifted optimizer can choose to slide these coordinates on the surface to better match the data points and data normals, while simultaneously updating the shape or pose hypothesis. On the other hand, the ICP optimizer operates on only one set of variables each iteration, without access to gradient information in the other variables. U can be viewed as latent variables as we eventually retain only the final θ estimate. As in [27], the lifted optimizer uses Levenberg steps with damping.

Figure 2 illustrates how the continuous surface normals for the subdivision surface and Phong surface cause the data normal term to 'pull' the coordinates in the directions of the blue arrows, leading to faster convergence than for the triangular mesh.

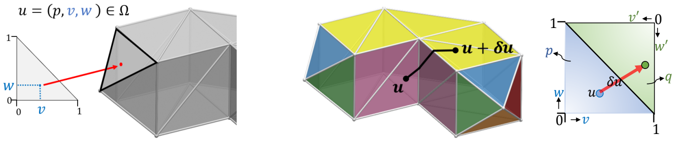
Note that *lifted optimization* has a close relation to *point-to-plane ICP* but is mathematically richer while being more efficient. Both optimizers allow a model to slide against the data in each iteration, improving convergence. However, *point-to-plane ICP* addresses this for only one energy formulation (point-to-model distances) whereas a lifted optimizer generalizes to arbitrary differentiable objectives. For example, we have a normal disparity term in our energy (Eq. 4), which is critical for faster convergence as shown in Fig. 7 and Table 1; point-to-plane ICP cannot minimize this objective. See our supplementary material for more illustrations.

2.3 Correspondence update on triangles

The surface types *Subdiv.*, *Phong* and *Tri. mesh* are all defined by a triangular control mesh. As mentioned in Sec 2.1, we write correspondences as surface coordinates $u = \{p, v, w\}$. The entire surface can be viewed as a collection of triangular patches indexed by p, with each patch parameterized by the unit triangle.

After a Levenberg step in lifted optimization, we apply an update $u := u + \delta u$, which may involve walking across adjacent triangles. We follow the same triangle-walking scheme as in [4, 26], as illustrated in Fig. 3b.

Fig. 3b (right) shows a single transition of a correspondence from one triangle to its neighbor; u and δu are denoted as a 2D point and a 2D vector respectively, in the domain space of the current patch p (colored in light blue). When $u + \delta u$ leaves the domain of triangle p, we calculate the partial update such that $u + r\delta u$ lies on the boundary of that triangle, and then map the remaining update $(1 - r)\delta u$ to the adjacent patch q (colored in light green). For *Subdiv.*, this results in a path that is tangent-continuous on the surface because this surface has C^1 continuity across patches. Note that this C^1 assumption does not hold for *Phong* and *Tri. mesh*; however, we implement the walking in exactly the same way for all surface types and find that it works well in practice.



(a) Surface parameterization (b) Correspondence update across triangles.

Fig. 3: (a) A surface correspondence u that lies on the p-th triangle patch, and its coordinate (v, w) in the unit-triangle domain Ω of this patch. (b) Walking on a triangle mesh to apply an update δu and walking in the domain space (figure from [26]).

3 Experiments

We compare the three surface types mentioned above: Loop subdivision surface (*Subdiv*), Phong surface (*Phong*) and triangular mesh surface (*Tri. mesh*), in both *Lifted* and *ICP* optimization frameworks. First we analyze their efficiency and convergence properties on rigid pose estimation of an ellipsoid. We further extend them to a more challenging scenario: fully articulated hand tracking. Specifically, we implement and evaluate these methods in the hand tracker by Taylor et al. [27] and in the hand tracker on HoloLens 2 [14].

In each experiment, we define the control parameters of each surface as follows: for each control vertex of *Subdiv*, we compute its position and normal on the limit Loop subdivision surface; we use these limit positions as the vertices of *Tri. mesh*, and use both limit positions and normals to define control vertices and normals of *Phong*. Note that this definition is purely to give surface models that are comparable for *Phong*, *Subdiv* and *Tri. mesh*, and we are free to define the Phong Surface model (i.e. control vertex positions and normals) in the way that best represents the target geometry.

We run most experiments on a desktop machine equipped with an Intel® Xeon® W-2155 CPU and 32GB RAM, except for Fig. 12b, which is on a Microsoft HoloLens 2.

3.1 Rigid pose alignment of an ellipsoid

Problem. We parametrize the ellipsoid pose as a 6D vector θ, storing translation and (axis-angle) rotation $[t_x, t_y, t_z, r_x, r_y, r_z]$. It defines a translation vector $\mathbf{t}(\theta) \in \mathbb{R}^3$ and a rotation matrix $R(\theta) \in \mathbb{R}^{3\times3}$. Given a template mesh, each control vertex position $\boldsymbol{v}_i$ is posed according to θ:

$$\boldsymbol{v}_i(\theta) = R(\theta)\boldsymbol{v}_i + \boldsymbol{t}(\theta). \tag{5}$$

For Phong surfaces, we express also control vertex normals as a function of θ (this is more efficient than re-computing the normals from the posed vertices):

$$\boldsymbol{v}_i^{\perp}(\theta) = R(\theta)\boldsymbol{v}_i^{\perp}. \tag{6}$$

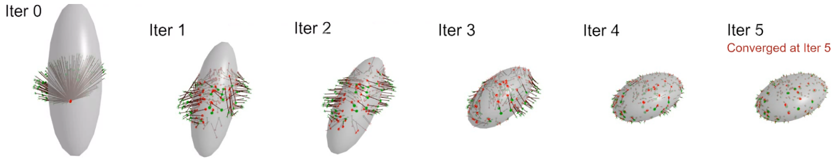


Fig. 4: Lifted optimization using a Phong surface ellipsoid model. Green points are the target data, with black lines joining them to their corresponding surface points in red.

This defines the $\boldsymbol{v}_i(\theta)$, $\boldsymbol{v}_i^{\perp}(\theta)$ introduced in Sec. 2.1. Derivatives $\frac{\partial \boldsymbol{v}_i(\theta)}{\partial \theta}$, $\frac{\partial \boldsymbol{v}_i^{\perp}(\theta)}{\partial \theta}$ can be trivially computed.

The objective function includes only the data term stated in Eq. 4. The surface evaluations are computed as in Sec. 2.1. In our experiments, we set $D = 200$. For fair comparisons among various surfaces and especially *Tri. mesh*, we did parameter sweeping on λ_n in the range $[0.0, 1.0]$ for fitting 400 random rotations. Based on the average fitting error, we find optimal $\lambda_n = 1.0$ for *Subdiv* and *Phong*, and $\lambda_n = 0.05$ for *Tri. mesh*. For the following experiments, for each surface type, we use its optimal λ_n. Note that we have included 0.0 in our λ_n sweeping.

Fig. 4 shows an example fitting result. See supplementary material for more qualitative comparisons.

Input data. Starting with the axis-aligned ellipsoid (with radii 1, 2, 3) centered at the origin (referred to as *neutral pose*), we apply a target rigid transform to obtain a ground-truth (target) pose. We then sample randomly 200 data points and normals on the subdivision surface defined by this mesh. For quantitative convergence analysis, we only sample from the triangle patches facing the positive z-axis: this gives us an incomplete set of data points. We add random noise to the data points and normals by sampling uniform random distributions with range [0.0, 0.1] in each dimension. The initial starting pose for model fitting is always the neutral pose.

Metrics. For the quantitative analysis, we focus on rotations. We compute a pose estimation error by applying the fitted and ground-truth rigid transformations to a fixed vector (e.g., $[1, 0, 0]$), and measuring the angle (in degrees) between the two transformed vectors. To allow for the $180°$ symmetry of an ellipsoid, we define the fitting error as the minimum of two angles, one computed using $[1, 0, 0]$ as the fixed vector and the other computed using $[-1, 0, 0]$ as the fixed vector.

Quantitative analysis. We run 400 trials targeting ground-truth poses $[0, 0, 0, y, y, y]$, where y is uniformly sampled from $(-\pi, \pi)$. Fig. 5 and 6 show the performance obtained for the rigid alignment of an ellipsoid model with 320 facets. Fig. 5 shows that *Phong* performs as well as *Subdiv.* in accuracy, and much better than *Tri. mesh*, with either *ICP* or *lifted* optimization. In Fig. 6 (left), we see that *lifted* optimization converges much faster than *ICP* for both *Phong* and *Subdiv*.

Fig. 6 (right) further plots accuracy (on the x-axis) against speed (on the y-axis, measured in milliseconds). It shows that *Phong* achieves the same level of accuracy as *Subdiv.* and runs as fast as *Tri. mesh* for both *lifted* (solid lines) and *ICP* (dashed lines). It also shows that lifted optimization converges much faster than ICP, e.g. *Lifted Phong*

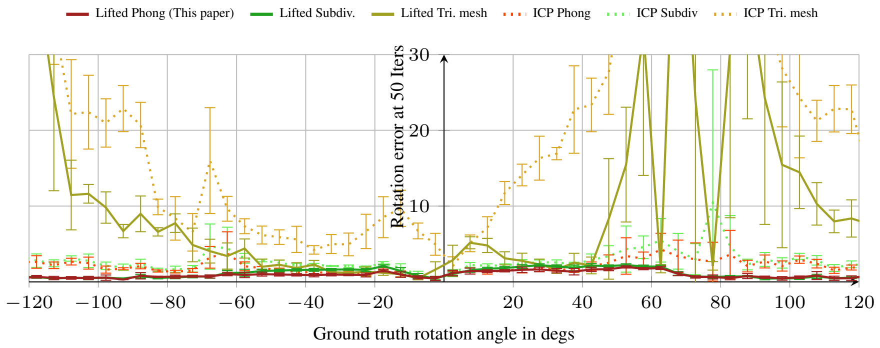


Fig. 5: Accuracy results for the rigid pose alignment of an ellipsoid with *320* facets. Optimizer ran for max. 50 iterations.

Surface type	Avg. rot. err. at 10 iters	Avg. rot. err. at 50 iters
Subdiv.	$9.89°$	$1.21°$
Subdiv. w/o normal	$14.81°$	$2.57°$
Phong	$\mathbf{8.13°}$	$\mathbf{0.99°}$
Phong w/o normal	$23.24°$	$3.54°$
Tri. mesh	$17.47°$	$11.07°$
Tri. mesh w/o normal	$23.24°$	$3.54°$

Table 1: Average rotation error after 10 and 50 lifted optimization iterations, with and without data normal term.

can achieve an average rotation error $< 10°$ within 8 iters, while *ICP Phong* needs 30 iterations.

Note that our runtime for *ICP* is slightly slower than *lifted* here. While a faster ICP might be achievable by further code optimization, we emphasize that the per-iteration cost of lifted is actually comparable to ICP when counting the theoretical FLOP computations. See our supplementary material (part 3) for details.

Data normal term. Here we assess the importance of the data normal term in Eq. 4. In particular, we demonstrate that this term is critical for fast convergence in both lifted and ICP optimizations, and that a continuous normal field improves both the basin of convergence and the accuracy of pose estimation.

As shown in Fig. 7, *Phong* and *Subdiv.*, which have a continuous normal field, converge much faster when the normal term is included, and achieve better accuracy after convergence (see also Table 1). Note that Phong w/o normal (solid bright red) coincides with Tri. mesh w/o normal (dashed black) for both lifted and ICP optimization.

For *Tri. mesh* (dashed lines in Fig. 7 (left)), we observe faster convergence with the data normal term within 10 iterations, but the accuracy reached after convergence is worse. We believe this is because the piecewise-constant surface normals introduce local minima for the correspondences, where reassignment to a different triangle causes

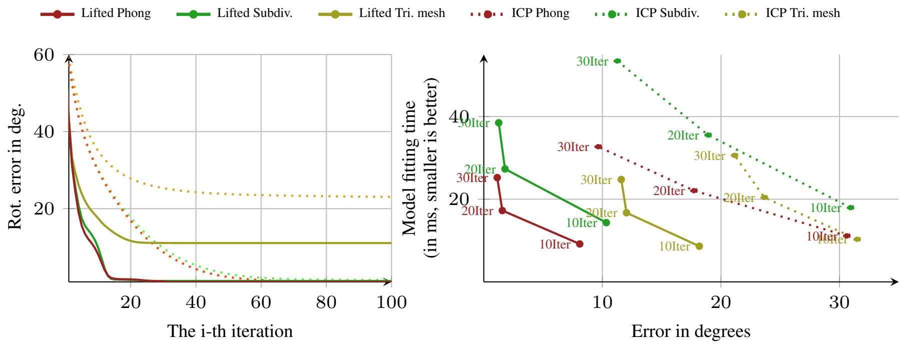


Fig. 6: Left: Convergence results for the rigid pose alignment of an ellipsoid with 320 facets. Right: Speed (model-fitting time in milliseconds) vs. accuracy (avg. error) comparisons on lifted and ICP optimizations, for max.10, 20, and 30 iters. Closer to the origin is better.

discrete jumps in the energy, and it is difficult for the optimizer to find a global minimum for these correspondences as the partial derivatives $\frac{\partial S^{\perp}}{\partial u}$ are zero. Note that the accuracy of *Tri. mesh w/o normal* is still 3 times worse than the *Phong* and *Subdiv.* models when including the normal term (see Table 1, third column).

3.2 Performance on hand tracking

The *lifted* optimizer used in the hand tracker by Taylor et al. [27] uses Loop subdivision surfaces. In this experiment, we simply replace their subdivision surface with our Phong surface. We leave everything else unchanged. Fig. 8 shows an example result obtained fitting our Phong surface with lifted optimization. Fig. 9 compares qualitatively the hand model fitting results at the 2nd and 4th iterations of lifted optimization using various surface types, given the same starting point.

Problem. We optimize a set of hand pose parameters $\theta \in \mathbb{R}^{28}$; θ stores hand orientation and translation, plus the 22 joint angles of the hand skeleton. From θ, the hand surface vertices are computed by Linear Blend Skinning [8].

For Phong surface, the 3D mesh vertex normals are deformed in the same way as vertex positions according to their LBS weights. This computation for the control normals gives a close approximation of the normals that we would obtain by rederiving normals from the posed positions of local vertices, with far greater efficiency.

We refer the reader to [27] for details on the objective function, data and experimental setup and evaluation metrics.

Performance on Dexter. In Fig. 10, 11 and 12a, we compare the accuracy and the computational efficiency of the tracker with different surface types on the Dexter dataset [23]. We adopt the same experimental setup as in [27].

The tracking settings for Fig. 10 are 192 data points, 10 starting points and 10 iterations. Fig. 10a shows the max and average error of per-frame fitting, i.e. fitting each frame independently with 10 new starting points. Fig. 10b shows the max and average

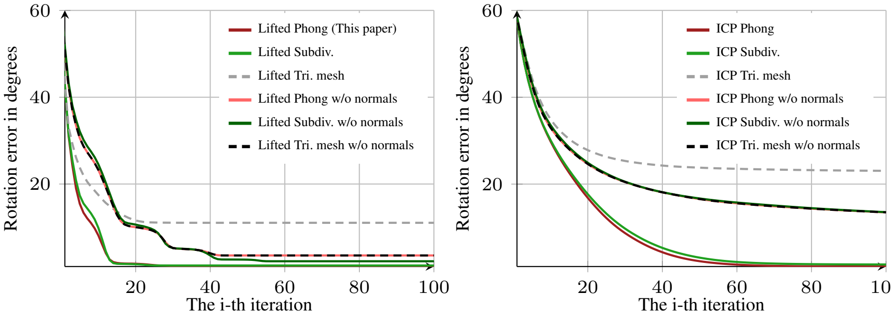


Fig. 7: Impact of the data normal term on optimization convergence, for the rigid pose alignment of an ellipsoid (320 facets). Left: Lifted optimization. Right: ICP optimization.

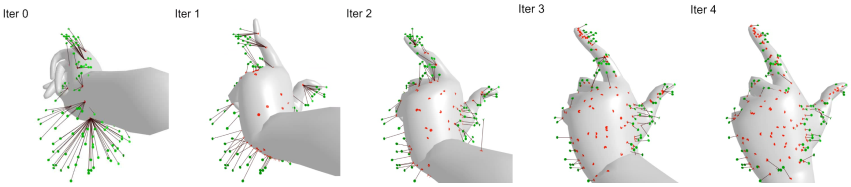


Fig. 8: A starting point and 4 iterations of lifted optimization using a Phong surface hand model. Green points are the target data, with black lines joining them to their corresponding surface points in red.

error of tracking, i.e. using the tracked pose in the previous frame (if available) as one of the 10 starting points for current frame. Both show that on this dataset, *Phong* performs as well as *Subdiv* and achieves higher accuracy than *Tri. mesh*. *Lifted* optimization is slightly better than *ICP*.

Robustness to initialization. Model-fitting methods are often sensitive to initialization, due to the non-convex objectives. This is why the tracking accuracy (Fig. 10b) is better than the per-frame fitting (Fig. 10a), where the optimization starts from scratch each time. The gap between *Phong* and *Tri. mesh* is larger in the per-frame fitting (Fig. 10a), which means that the smooth normal field is the key for faster convergence when the starting point is poorer. We emphasize the importance of fast convergence in live experience, as tracking failures often occur when hands are out of the field of view, or in the presence of self- and object-occlusions.

The computational cost of model fitting is dominated by 3 variables: (i) the number of data points in the data term; (ii) the number of starting points used to initialize the optimizer; and (iii) the number of iterations for each starting point. Fig. 11 shows the impact of these variables on accuracy and convergence. Again, *Phong* behaves similarly to *Subdiv.*, and much better than *Tri. mesh*. Fig. 11b shows how *Lifted* exhibits better convergence than *ICP*, confirming the conclusion in [27].

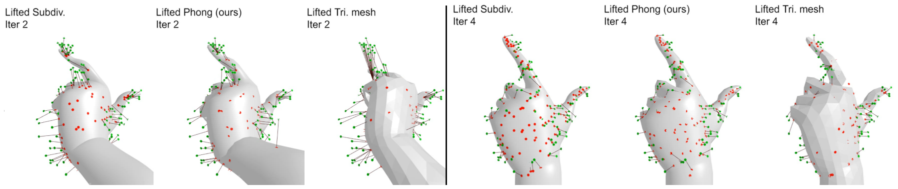
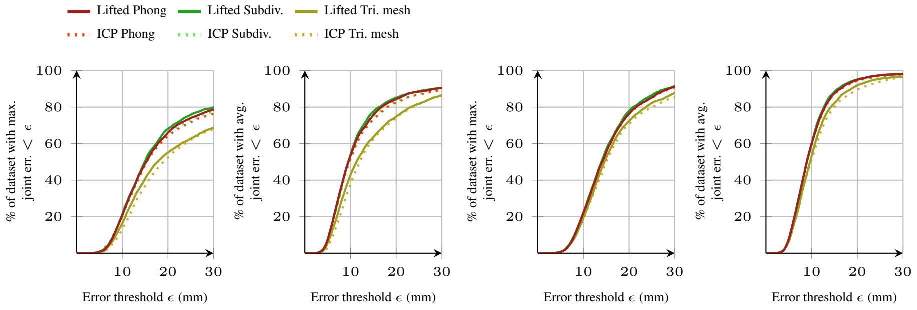


Fig. 9: Fitted hand model result at the 2nd and 4th iterations of lifted optimization using various surface types.



(a) Max (left) and Avg. (right) joint error on *per-frame fitting*. (b) Max (left) and Avg. (right) joint error on *tracking*.

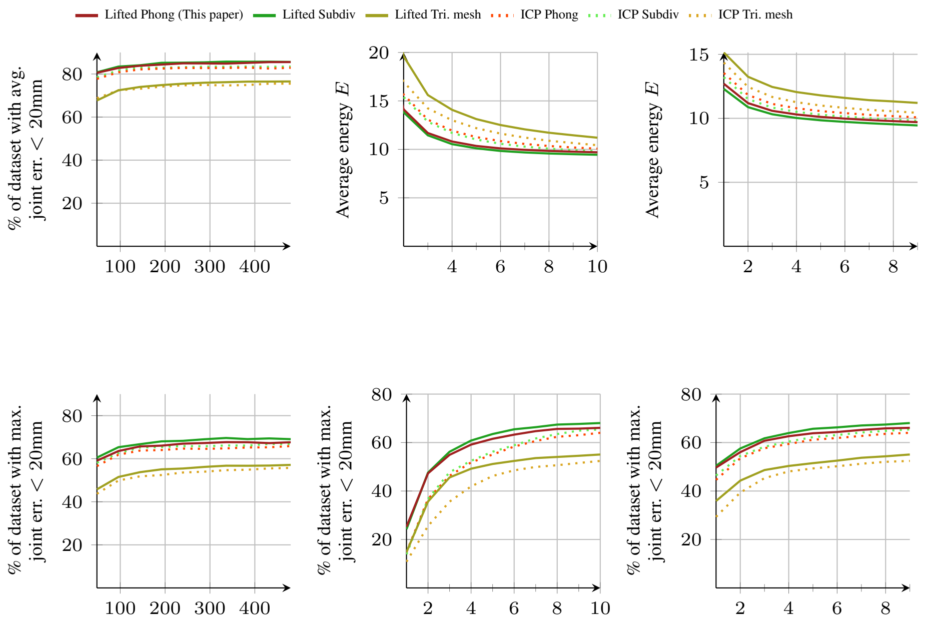
Fig. 10: Accuracy comparison of surface types with lifted optimizer and ICP on Dexter.

These tests show that the smooth normal field of the model is important for faster convergence in lifted optimization; it is not actually relevant whether the mesh geometry is smooth or not.

Fig. 12a shows the speed vs. accuracy plot in the per-frame fitting case (192 data points, 10 starting points). For each surface type, the number of iterations varies from 2 to 10. The x-axis reports the accuracy, measured as the percentage of dataset frames that have average joint error $< 20mm$. The y-axis reports the speed in FPS (per starting point) of the model-fitting stage, i.e. not including the preprocessing time. For example, in the *lifted* case (solid lines), if we require to run the fitting at 50fps, we can perform 6 iterations for *Phong* and *Tri. mesh*, but only 4 iterations for *Subdiv.*, and *Phong* provides the highest accuracy at this speed. Alternatively, if we require the fitter to reach near 80% accuracy, we can run 4 iterations with *Phong* and *Subdiv.*, but *Phong* is 20% faster. So *Phong* achieves almost the same level of accuracy as *Subdiv*, while being as cheap as *Tri. mesh* in terms of efficiency.

Similar conclusions can be drawn for ICP optimizations (dashed lines in Fig. 12a). As pointed out earlier at the end of Sec. 3.1, our runtime for *ICP* is slightly slower than *lifted*, but we show that per-iteration cost of lifted is actually comparable with ICP in our supplementary material, part 3.

Note that the accuracy achieved by *Lifted Subdiv.* after 8 iterations coincides with that achieved by *Lifted Phong* after 10 iterations. This is because *Lifted Phong* already



(a) Effect of the number of data points N on accuracy.

(b) Effect of the number of iterations on energy and accuracy.

(c) Effect of the number of starting points on energy and accuracy.

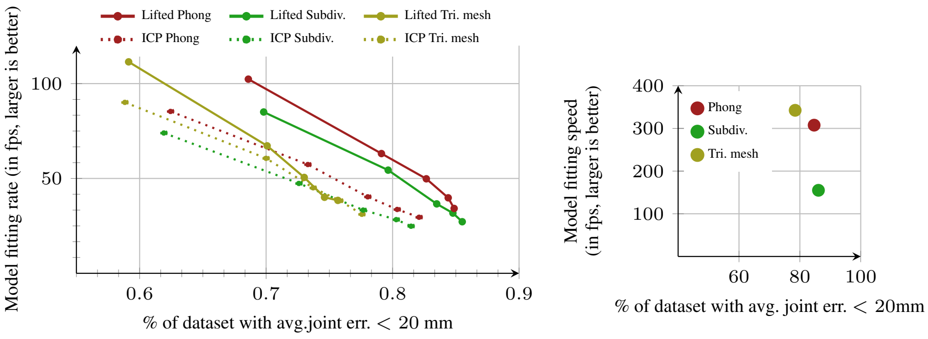
Fig. 11: Effect of alternative optimization configurations when fitting to Dexter. The results shown are for per-frame model fitting.

converged after 8 iterations, and further iterations do not improve accuracy further (see also Fig. 11b).

Performance on HoloLens 2. Starting from the work of [27], we made many improvements to enable us to run a hand tracker in real time on the HoloLens 2 [14], a mobile device with very limited computational and power resources. The Phong surface model presented here was one of the key efficiency improvements that was required. Fig. 12b shows the speed vs. accuracy of various surface types with lifted optimization in this hand tracker on HoloLens 2, evaluated on a captured depth dataset. The *Phong* surface (red dot) can be evaluated twice as fast as the subdivision surface (green dot), and gives the same level of accuracy.

4 Conclusions

In this paper we demonstrated that the convergence benefits of lifted optimization are available to a wider range of surface models than was previously thought. We introduced Phong surfaces, and showed that they provide sufficient information about the local model geometry to allow a model-fitting optimizer to converge fast, whilst requiring a



(a) Application on the hand tracker by Taylor et al. [27].

(b) Our hand tracker on HoloLens 2.

Fig. 12: (a) The speed (model-fitting speed in fps for one starting point) vs. accuracy (percentage of frames with avg. joint error less than 20 mm) for per-frame fitting using the hand tracker by Taylor et al. [27] on Dexter. Upwards and to the right is better. The dots from top to bottom on each line denote the number of iterations being 2, 4, 6, 8, 10. (b) The speed vs. accuracy for our hand tracker on HoloLens 2.

fraction of the compute of expensive smooth surface models. Beside rigid pose alignment and hand tracking, the proposed method can be applied to various 3D surface model-fitting applications, for example, 3D pose and shape estimation of SMPL body [2], face [10] and animals [33, 34], and is particularly valuable when computational budget is limited.

Given the generalization we show in this paper, a natural question is 'what are the set of requirements in order for lifted optimization to work effectively?' We hypothesize that the only requirement is for a model to provide sufficient approximations to the energy tangent space $\partial E/\partial\theta$ for a gradient-based optimizer to take efficient steps in each iteration, with an implied freedom on global topology and connectivity, as well as the form taken by those approximations. We intend to explore this hypothesis more fully in future work.

References

1. Besl, P.J., McKay, N.D.: A method for registration of 3-D shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence **14**(2), 239–256 (1992) 1, 3
2. Bogo, F., Kanazawa, A., Lassner, C., Gehler, P., Romero, J., Black, M.J.: Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. In: European Conference on Computer Vision. pp. 561–578 (2016) 1, 14
3. Bogo, F., Romero, J., Loper, M., Black, M.J.: Faust: Dataset and evaluation for 3d mesh registration. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 3794–3801 (2014) 3
4. Cashman, T.J., Fitzgibbon, A.W.: What shape are dolphins? Building 3D morphable models from 2D images. IEEE Transactions on Pattern Analysis and Machine Intelligence **35**(1), 232–244 (2013) 2, 4, 6
5. Chen, Y., Medioni, G.: Object modeling by registration of multiple range images. In: IEEE International Conference on Robotics and Automation. pp. 2724–2729 (1991) 3
6. Fitzgibbon, A.: Robust registration of 2D and 3D point sets. In: Proceedings of the British Machine Vision Conference. pp. 411–420 (2001) 3, 4
7. Hirshberg, D., Loper, M., Rachlin, E., Black, M.: Coregistration: Simultaneous alignment and modeling of articulated 3D shape. In: European Conference on Computer Vision. pp. 242–255 (2012) 1
8. Khamis, S., Taylor, J., Shotton, J., Keskin, C., Izadi, S., Fitzgibbon, A.: Learning an efficient model of hand shape variation from depth images. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 2540–2548 (2015) 4, 10
9. Kolotouros, N., Pavlakos, G., Black, M., Daniilidis, K.: Learning to reconstruct 3D human pose and shape via model-fitting in the loop. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 2252–2261 (2019) 1
10. Li, T., Bolkart, T., Black, M.J., Li, H., Romero, J.: Learning a model of facial shape and expression from 4D scans. ACM Transactions on Graphics **36**(6), 194:1–194:17 (2017) 1, 14
11. Loop, C.T.: Smooth Subdivision Surfaces Based on Triangles. Master's thesis, University of Utah (1987) 2
12. Magic Leap Inc: Perception at Magic Leap (2019), https://sites.google.com/view/perceptionatmagicleap/ 1
13. Marquardt, D.W.: An algorithm for least-squares estimation of nonlinear parameters. Journal of the Society for Industrial and Applied Mathematics **11**(2), 431–441 (1963) 3
14. Microsoft: HoloLens 2 (2019), https://blogs.microsoft.com/blog/2019/02/24/microsoft-at-mwc-barcelona-introducing-microsoft-hololens-2 2, 7, 13
15. Mueller, F., Davis, M., Bernard, F., Sotnychenko, O., Verschoor, M., Otaduy, M.A., Casas, D., Theobalt, C.: Real-time pose and shape reconstruction of two interacting hands with a single depth camera. ACM Transactions on Graphics **38**(4), 49:1–49:13 (2019) 1, 4
16. Neugebauer, P.J.: Geometrical cloning of 3D objects via simultaneous registration of multiple range images. In: International Conference on Shape Modeling and Applications. pp. 130–139 (1997) 3
17. Pavlakos, G., Choutas, V., Ghorbani, N., Bolkart, T., Osman, A.A.A., Tzionas, D., Black, M.J.: Expressive body capture: 3D hands, face, and body from a single image. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 10975–10985 (2019) 1
18. Pellegrini, S., Schindler, K., Nardi, D.: A generalisation of the ICP algorithm for articulated bodies. In: Proceedings of the British Machine Vision Conference. pp. 87.1–87.10 (2008) 3
19. Phong, B.T.: Illumination for computer generated pictures. Communications of the ACM **18**(6), 311–317 (1975) 2, 4

20. Qian, C., Sun, X., Wei, Y., Tang, X., Sun, J.: Realtime and robust hand tracking from depth. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 1106–1113 (2014) 1
21. Rusinkiewicz, S.: A symmetric objective function for ICP. ACM Transactions on Graphics **38**(4), 85:1–85:7 (2019) 3
22. Rusinkiewicz, S., Levoy, M.: Efficient variants of the icp algorithm. In: International Conference on 3D Digital Imaging and Modeling. pp. 145–152 (2001) 3
23. Sridhar, S., Oulasvirta, A., Theobalt, C.: Interactive markerless articulated hand motion tracking using RGB and depth data. In: International Conference on Computer Vision. pp. 2456–2463 (2013) 10
24. Sullivan, S., Ponce, J.: Automatic model construction and pose estimation from photographs using triangular splines. IEEE Transactions on Pattern Analysis and Machine Intelligence **20**(10), 1091–1097 (1998) 3
25. Tagliasacchi, A., Schröder, M., Tkach, A., Bouaziz, S., Botsch, M., Pauly, M.: Robust articulated-ICP for real-time hand tracking. Computer Graphics Forum **34**(5), 101–114 (2015) 1, 3
26. Taylor, J., Stebbing, R., Ramakrishna, V., Keskin, C., Shotton, J., Izadi, S., Hertzmann, A., Fitzgibbon, A.: User-specific hand modeling from monocular depth sequences. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 644–651 (2014) 2, 4, 6, 7
27. Taylor, J., Bordeaux, L., Cashman, T., Corish, B., Keskin, C., Sharp, T., Soto, E., Sweeney, D., Valentin, J., Luff, B., Topalian, A., Wood, E., Khamis, S., Kohli, P., Izadi, S., Banks, R., Fitzgibbon, A., Shotton, J.: Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. ACM Transactions on Graphics **35**(4), 143:1–143:12 (2016) 1, 2, 4, 6, 7, 10, 11, 13, 14
28. Taylor, J., Tankovich, V., Tang, D., Keskin, C., Kim, D., Davidson, P., Kowdle, A., Izadi, S.: Articulated distance fields for ultra-fast tracking of hands interacting. ACM Transactions on Graphics **36**(6), 244:1–244:12 (2017) 1, 3, 4
29. Tkach, A., Pauly, M., Tagliasacchi, A.: Sphere-meshes for real-time hand modeling and tracking. ACM Transactions on Graphics **35**(6), 222:1–222:11 (2016) 1
30. Wan, C., Probst, T., Gool, L.V., Yao, A.: Self-supervised 3D hand pose estimation through training by fitting. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 10845–10854 (2019) 1
31. Xiang, D., Joo, H., Sheikh, Y.: Monocular total capture: Posing face, body, and hands in the wild. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 10957–10966 (2018) 1
32. Zheng, J., Zeng, M., Cheng, X., Liu, X.: SCAPE-based human performance reconstruction. Computers & Graphics **38**, 191–198 (2014) 3
33. Zuffi, S., Kanazawa, A., Black, M.J.: Lions and tigers and bears: Capturing non-rigid, 3d, articulated shape from images. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 3955–3963 (2018) 14
34. Zuffi, S., Kanazawa, A., Jacobs, D.W., Black, M.J.: 3d menagerie: Modeling the 3d shape and pose of animals. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 5524–5532 (2017) 14