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

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

1 Lifting vs. point-to-plane ICP

Here we show that lifted optimization is related to point-to-plane ICP, but is mathematically richer.

We use the rigid alignment of a 2D curve C to a set of data points $\{\mathbf{x}_i\}_{i=1}^N$ to illustrate point-to-plane ICP updates, see Fig. 1. As point-to-plane ICP finds correspondences between the data and the tangent space of the model, in the 2D case we will be performing 'point to tangent line' updates. Let θ parametrize the translation and rotation of the curve, which are the parameters we want to solve for.

Both point-to-plane ICP and lifting first need to select the point-to-curve correspondences between N data points and the posed curve $C(\theta)$. There are many point-matching variants [3], and Fig. 1a depicts simple closest-point correspondences, but our analysis extends to any initial correspondence proposals. Let us denote these correspondences by curve parameter positions $\{t_i\}_{i=1}^N$, and thus data point $\mathbf{x}_i$ is paired with curve point $C(t_i, \theta)$.

The tangent line at model point $C(t_i)$ is $\{C(t_i) + \gamma \dot{C}(t_i) | \gamma \in \mathbb{R}\}$, where $\dot{C}(t_i)$ denotes the unit-length tangent vector at t_i. Then one completely unconstrained point-to-plane ICP update (Fig. 1b) finds θ^* that minimizes the distance of point-to-tangent-line:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_i \min_{\gamma_i} \|\mathbf{x}_i - (C(t_i, \theta) + \gamma_i \dot{C}(t_i, \theta))\|^2, \tag{1}$$

where $(C(t_i, \theta) + \gamma_i \dot{C}(t_i, \theta))$ is the projection of the data point on the tangent line when evaluated at the position γ_i that minimizes the distance to $\mathbf{x}_i$, i.e., the footpoint. Note that this is a 2D version of the original point-to-plane method by Chen and Medioni [1], and it is equivalent to using the distance metrics based on the normal at $C(t_i, \theta)$ described by Low [2]. As shown in Fig. 1b, this could lead to a bad update when the starting point is not close enough.

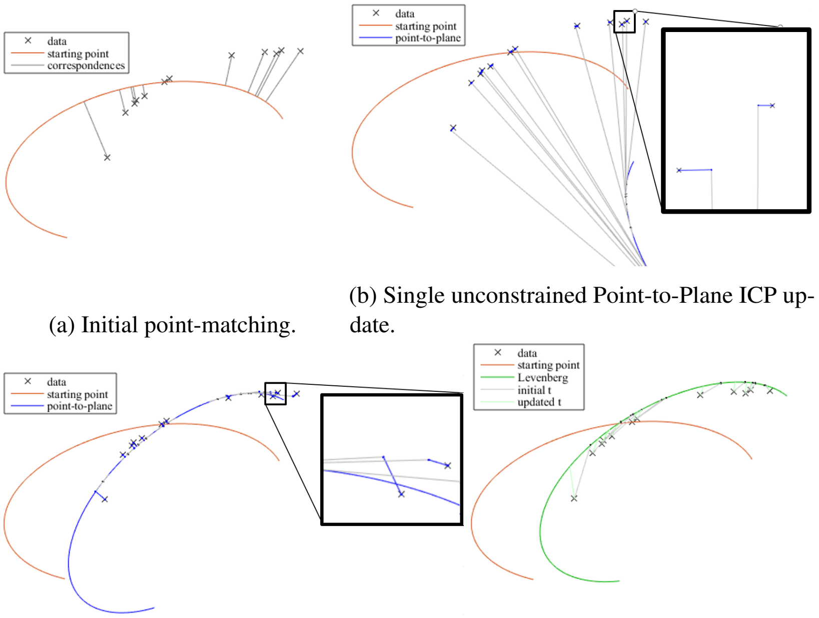
To do better, a regularized version of point-to-plane ICP (Fig. 1c) could be to find θ^* that minimizes the distance of both point-to-point and point-to-tangent-line:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_i \min_{\gamma_i} \|\mathbf{x}_i - (C(t_i, \theta) + \gamma_i \dot{C}(t_i, \theta))\|^2 + \lambda \|\gamma_i\|^2. \tag{2}$$

Here the inner minimization over $\{\gamma_i\}_{i=1}^N$ can be solved separately as a series of least-squares problems, and θ solved afterwards in each iteration.

Finally, lifted optimization (Fig. 1d) simultaneously finds θ^* and $T = \{t_i\}_{i=1}^N$ that minimizes

$$\theta^*, T^* = \underset{\theta,T}{\operatorname{argmin}} \sum_i \|\mathbf{x}_i - C(t_i, \theta)\|^2 \tag{3}$$



(a) Initial point-matching.

(b) Single unconstrained Point-to-Plane ICP update.

(c) Single regularized Point-to-Plane ICP update.

(d) Single lifted update.

Fig. 1: Illustration of point-to-plane ICP and lifted optimization.

We can see that point-to-plane ICP shares similarities with a lifted update; both allow a model to slide against the data in each iteration, improving convergence. While lifting achieves the sliding by simultaneously optimizing θ and T in the update, point-to-plane ICP achieves the sliding by projection onto the model tangent plane. 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. Recall that we include a normal disparity term in our energy (Eq. 4 in main paper), which is critical for faster convergence as shown in the experiment described in Sec. 3.1 and Fig. 7 of the main paper; point-to-plane ICP cannot minimize this objective.

2 Efficiency of lifting vs. ICP

As shown in Fig. 6 (right) and Fig. 12 in main paper, our actual runtime for ICP is *slower* than for lifting in terms of wall-clock timing on PC. While we believe our implementations for both are reasonable, this efficiency comparison could still be unfair in many ways.

Instead, let us use FLOP (Floating Point Operation) counts to theoretically compare the two in the context of hand tracking. Note that in the extreme low-power case we discuss (1% of an iPhone 7), we assume the programmer is already optimizing cache hit rates etc. In our code for hand tracking on HoloLens 2, for example, our tightest loop is running at 75% of theoretical peak FLOPS.

Several of the steps of ICP are comparable to lifting:

1. A major component of the cost is determining the point-to-mesh correspondences between N data points (less than 200 for our hand tracking examples), and M sampled mesh points. For lifting, M can be relatively small (e.g. 300), as most correspondences accept their lifted update. We might hope that a spatial index over mesh samples would accelerate this per-data-point query, but as the mesh changes at every iteration, a KD-tree or other spatial data structure would need to be rebuilt at every iteration, and for $M \approx 300$ this cost is never recouped. The cost is therefore $O(MN)$ FLOPs, although the constant factor can be small with efficient SIMD usage.
   For ICP, as the correspondences are held fixed, convergence depends on finding good matches at every iteration. The mesh samples therefore either need to be numerous, say $M = 3000$, where a KD-tree can achieve asymptotic complexity $O(N \log M)$ but with a constant overhead that makes the incurred cost greater when amortized per query. Or else ICP can use smaller M but needs at least one Newton step per closest-point query, involving the solution of N 2×2 linear systems adding to the $O(MN)$ cost above.
   Parallelization is trivial for this step, but does not reduce the number of operations, and therefore cannot help to address the power constraints of a mobile device. The impact on latency is important but it does not change the FLOP count analysis here.
2. The second major component is the cost of solving for the θ update (P params), and the N correspondence updates in the case of lifting. For ICP this involves a $P \times P$ linear system solve per sub-iteration at a cost of $O(P^3)$. In lifting, the system is more complex but still quite sparse, adding $O(PN)$ operations.
   Taylor et al. [4] also point out in their Section 3.3.1 that lifted optimization scales linearly with N; in fact, the extra floating-point multiplications required for lifting to solve $\Delta\theta$ (their Eq. 22) and ΔU are about $(18 + 4P)N$, which with $N = 128$ and $P = 28$ as in our HoloLens hand tracking example gives only $\approx$ 16K additional floating-point multiplications per iteration (<2MFLOPS overall).

So iteration timings are very comparable, and from our convergence figures, we emphasize that the second-order convergence of lifted optimization is much faster than the linear convergence observed for ICP. For example, Fig. 6 (left) in the main paper shows that Lifted Phong/Subdiv (red/green solid line in (a)) converges within ≈ 13 iterations, while the ICP Phong/Subdiv (red/green dashed line) within ≈ 50 iterations, i.e. **3.8 times more**. For the hand tracking experiment in the main paper (Fig. 12(a)), to achieve an accuracy level where 79% of dataset has average joint error $<$ 20mm, Lifted Phong (red solid line) requires ≈ 4 iterations, while ICP Phong (red dashed line)

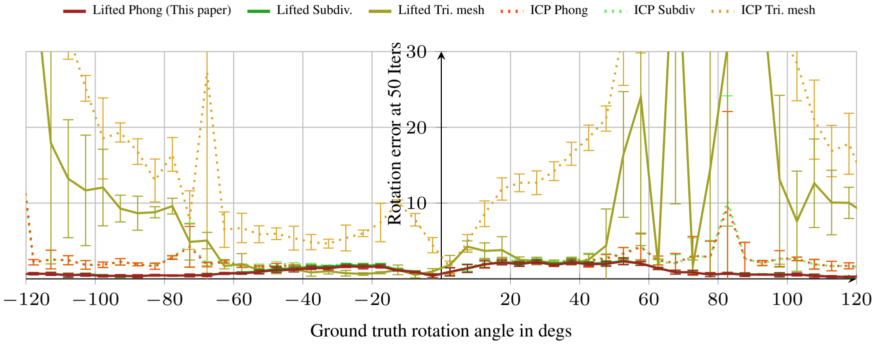


Fig. 2: Rigid alignment accuracy for an ellipsoid with 1280 facets. The optimization runs for max. 50 iterations.

requires ≈ 7 iterations, i.e. **1.75 times more**. Recall that from the analysis above, the per-iteration cost of lifting for $N \leq 200$ is comparable to ICP.

In summary, depending on several factors ICP can range from marginally cheaper than lifted optimization per iteration to considerably more expensive, but also comes with the cost of increases in iteration count (and decrease in basin of convergence, requiring more compute for any initial estimate).

3 More results on rigid pose estimation of an ellipsoid

Here we show more results on convergence comparison of various surface types and optimization frameworks on rigid pose estimation of an ellipsoid (Sec 3.1 in main paper).
Qualitative comparisons. Fig. 4 visualizes the optimization iterations to reach the ground-truth rigid pose $[0.1, 0.3, 2.0, 1, 1, 1]$. In the *lifted* case, the *Phong* and *Subdiv* surfaces both converge to the correct pose within 5 iterations; *Tri. mesh* needs 15 iterations. With *ICP*, the *Phong* and *Subdiv.* surfaces both converge to the correct pose within 24 iterations; *Tri. mesh* needs 35 iterations.
Varying mesh resolution. As for the *Phong* interpolated normals exhibit less variation with higher-resolution meshes, we run a similar experiment considering an ellipsoid with 4x denser triangles. Fig. 2 shows that, in this case, *Tri. mesh* get slightly better accuracy than with lower resolution. However, the overall performance seems comparable to that reported in Fig. 5 in the main paper.
Data point positional errors. In addition to the rotation error analysed in the main paper Fig. 6 (left), here we show the data point positional error and data point normal error in Fig. 3. It's clear that the *Tri. mesh* fits well to data point positions but quite bad to data normals.
Noise in data normals. As our proposed model-fitting method relies on data normal term for good convergence, one possible failure case will be when there is too much noise in the input data normals, due to either data too sparse or normal estimation too noisy. To confirm this, we tried to bump up the noise level in data normals in the ellipsoid

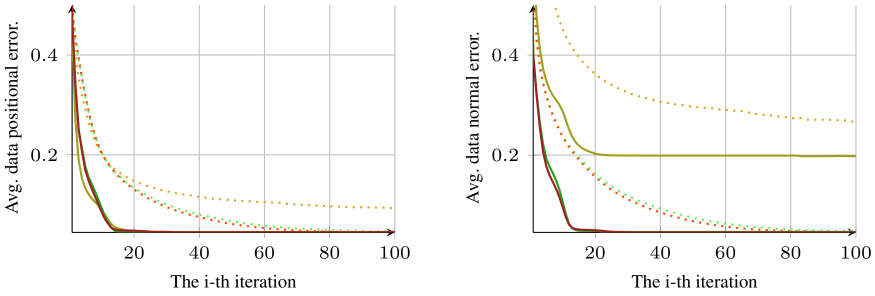


Fig. 3: Averaged data point positional error and normal error during fitting for an ellipsoid with 320 facets. The error is averaged across 400 trials. The optimization runs for max. 100 iterations.

example (Sec 3.1 and Fig. 5 in the main paper), from default random range [0.0, 0.1], up to [0.0, 0.25] and [0.0, 0.5]. We found that on average the *Phong* does get slightly worse at 0.25, but still much better than *Tri. mesh*; but the advantage gets much smaller at 0.5.

References

1. Chen, Y., Medioni, G.: Object modeling by registration of multiple range images. In: IEEE International Conference on Robotics and Automation. pp. 2724–2729 (1991) 1
2. Low, K.L.: Linear least-squares optimization for point-to-plane icp surface registration. Tech. rep. (2004) 1
3. Rusinkiewicz, S., Levoy, M.: Efficient variants of the icp algorithm. In: International Conference on 3D Digital Imaging and Modeling. pp. 145–152 (2001) 1
4. 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) 3

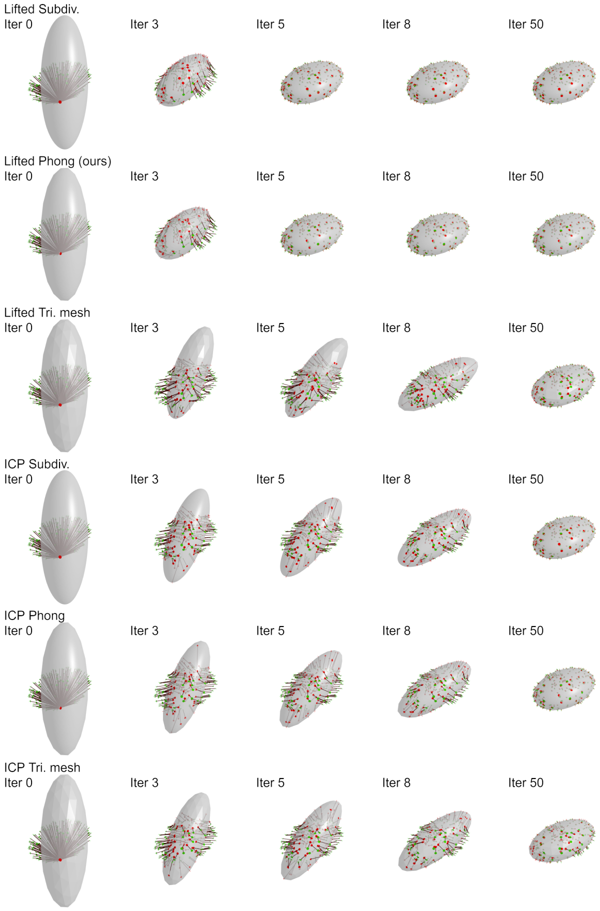


Fig. 4: Rigid pose alignment of an ellipsoid with 3 surface types in lifted optimization and ICP optimization. The green and red dots represent data points and surface points, respectively; black lines denote correspondences between the two. Only the first two methods converge within 5 iterations.