Neural Non-Rigid Tracking

We thank all the reviewers for their feedback, and are happy that our work was found to be "quite novel" (R3), "well written" (R4), "easy to follow" (R3), and supported by "promising results" (R1), that achieves "clearly state-of-the-art performance" (R2) while "significantly reducing the computational time [..] by 85 times compared to prior art [4]" (R2).

Relation to [4] and [15] (R1, R2). We propose to learn non-rigid tracking in an end-to-end fashion, where the solution of the non-rigid alignment informs correspondence prediction. Our differentiable formulation allows learning correspondence confidence in a completely self-supervised manner, similar to robust optimization. Such self-supervision of correspondence confidences results in improved tracking performance vs. directly supervising this task. In contrast, DeepDeform [4] learns sparse matches by individual heatmaps, resulting in a high compute cost and inferior performance. [15] has a different focus, as it learns a Preconditioned Conjugate Gradient to speed up solver convergence, using descriptors on nodes in a pixel-aligned graph. Our approach, which works on general graphs, learns to robustify dense correspondence prediction for non-rigid tracking by learning self-supervised correspondence confidences.

Robustness of dense vs. sparse correspondences (R1). We combine the advantages of both by learning a weight for each correspondence. This is inspired by robust optimization, and outperforms sparse correspondence works, e.g., [4].

Learning through introspection (R1). This is an interesting direction; in our approach, we aim to avoid direct reliance upon methods which maintain their own failure modes in very challenging deformation scenarios.

Related self-supervised RGB tracking (R1) and camera parameters (R2). We will discuss [Wang et al. 19], [Li et al. 19] and [Lai et al. 20] in the revised version, as well as clarify that camera parameters are given a priori.

Varying keyframe rates (R1). Tab. 1; performance remains stable even though frame-to-frame deformations increase.

Table 1: Deformation error (mm): more frames means lower sampling rate, i.e., more frame-to-frame motion.

Keyframe density	Deformation error (mm)
DeepDeform (filtering w/ neighboring frames)	31.52
Ours: keyframe every 100 frames	30.70
Ours: keyframe every 75 frames	29.68
Ours: keyframe every 50 frames	**28.72**

Warp loss is superset of graph loss (R2, R3). We agree that this is true in our particular case. We found it to be a more general notation to disentangle them (e.g., for scenarios where graph nodes are not sampled on the RGB-D frame).

ARAP edge re-weighting (R2). We sample nodes uniformly on the surface, thus, all edges have similar length (7.13 ± 1.38 cm). Hence, edge re-weighting changes EPE 3D only marginally (25.70 mm w/o vs. 25.49 mm w/).

How does the graph size affect the method (R2)? We sample graph nodes with 5 cm node coverage, which fits well with our setup of 11 GiB for training. Using coarser graphs, with 10 cm and 15 cm node coverage resulted in poorer performance: 5.49% and 8.33% higher EPE 3D, as well as 7.34% and 28.46% higher Graph Error 3D, respectively. In turn, the memory footprint on the GPU during training (with batch size 4) decreases with node coverage: 10 513 MiB, 6153 MiB and 5931 MiB for 5 cm, 10 cm and 15 cm node coverage, respectively.

Segmentation of foreground object (R2). Our self-supervised learning of correspondence weighting can robustly handle outliers that arise from noisy masks (note that the used annotated masks are not perfect segmentations).

# optimization steps (R2). We empirically found 3 solver iterations to be the best compromise between performance and computational cost. Unrolling 3 solver iterations instead of 1 / 2, results in 24.9% / 0.16% lower EPE 3D and 22.7% / 0.23% lower Graph Error 3D; 4 iterations only improves slightly wrt 3 (0.04% lower EPE 3D, 0.03% lower Graph Error 3D); more than 4 does not change performance notably. We will include this ablation in the revised version.

Nearest-neighbor vs. bilinear depth sampling depth (R2). We found bilinear sampling to perform better, with 5.8% lower EPE 3D and 6.29% lower Graph Error 3D compared to nearest-neighbor sampling.

Optical flow vs. scene flow (R3, R4). Our main contribution is self-supervised correspondence weighting with end-to-end non-rigid tracking, rather than the flow backbone. Thus, we leverage a state-of-the-art optical flow as our backbone (enabling significant pretraining); we believe that scene flow could also provide a useful backbone.

Texture fusion / DoubleFusion results on mocap dataset (R3). We will be happy to include these comparisons.

Ablation on weighting scheme (R4). This is indeed an exciting avenue that we would like to further explore in the future. Nevertheless, we believe our ablation on learning correspondence confidences, i.e., learning those in a completely self-supervised fashion (only possible thanks to our differentiable non-rigid tracker) vs. using supervision, can already be regarded as an exploration of the weighting scheme.