Neural Prior for Trajectory Estimation

Chaoyang Wang[1] Xueqian Li[2,3] Jhony Kaesemodel Pontes[3] Simon Lucey[12]
[1]Carnegie Mellon University [2]University of Adelaide [3]Argo AI
chaoyanw@cs.cmu.edu {xueqian.li, simon.lucey}@adelaide.edu.au jpontes@argo.ai

Abstract

Neural priors are a promising direction to capture low-level vision statistics without relying on handcrafted regularizers. Recent works have successfully shown the use of neural architecture biases to implicitly regularize image denoising, super-resolution, inpainting, synthesis, scene flow, among others. They do not rely on large-scale datasets to capture prior statistics and thus generalize well to out-of-the-distribution data. Inspired by such advances, we investigate neural priors for trajectory representation. Traditionally, trajectories have been represented by a set of handcrafted bases that have limited expressibility. Here, we propose a neural trajectory prior to capture continuous spatio-temporal information without the need for offline data. We demonstrate how our proposed objective is optimized during runtime to estimate trajectories for two important tasks: Non-Rigid Structure from Motion (NRSfM) and lidar scene flow integration for self-driving scenes. Our results are competitive to many state-of-the-art methods for both tasks.

1. Introduction

Representing space-time with 3D trajectories provides longer-term information about the dynamics of a 3D scene compared to pairwise representations such as scene flow. It also enables generic priors to solve underconstrained low-level vision tasks, especially for problems that need to be agnostic to different objects and scenes. Here we aim to model a general and dataless prior for estimating 3D trajectories through a neural runtime optimization approach.

Most works studying trajectory priors are from the Non-Rigid Structure from Motion (NRSfM) field, where they are handcrafted to solve the ill-posed inverse problem of lifting dynamic 2D points to 3D. The most straightforward trajectory prior assumes that points move smoothly over time [56, 57]. However, it does not provide enough constraints to disambiguate the camera and point motion. On the other hand, a collection of trajectories observed from a scene contain statistics for stronger priors. The seminal work of Akhter *et al*. [3] proposed to represent trajectories

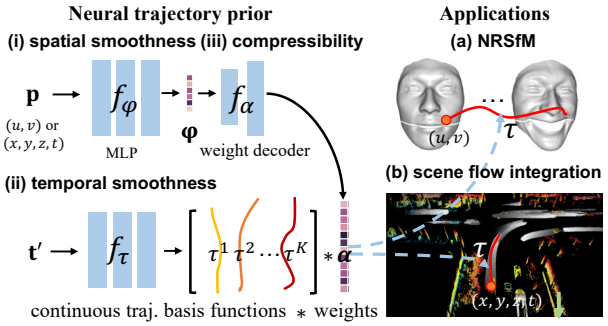


Figure 1. Our neural trajectory prior (NTP) regularizes the motion of a point to be smooth in space by the continuous mapping of a coordinate MLP f_φ. Trajectories are represented as a linear combination of overcomplete continuous basis functions τ^1, τ^2, ..., τ^K. The linear weights $\boldsymbol{\alpha}$ is regularized to be compressible to lower dimensional $\boldsymbol{\varphi}$, and the basis functions are generated by a temporal coordinate MLP f_τ, which implicitly enforces temporal smoothness. We show the results of applying NTP on NRSfM and lidar scene flow integration.

by linearly combining a small set of Discrete Cosine Transform (DCT) bases. However, it is well known that such low-rank linear models are not sufficient to represent complex motions and dense data where the number of trajectories is far greater than the length of the sequence. Recent state-of-the-art NRSfM methods [33, 34, 53] combine both shape and trajectory priors or are tailored towards specific scenes such as deforming surfaces [31, 42, 44]. In this paper, we revisit the idea of a general trajectory prior. Our approach differs from current methods in that we use the architectural regularization properties of neural networks.

We are inspired by the recent innovations of using coordinate MLPs [36, 38, 40, 45] and propose a new general neural trajectory prior to model continuous spatio-temporal motions of dynamic scenes (see Fig. 1). We utilize the smoothness bias of coordinate MLPs to enforce trajectories to be temporally smooth and encourage nearby positions to share similar motions. We also introduce a bottleneck layer to the model, thus effectively constraining the output trajectories to be compressible to low dimensions. MLPs with bottleneck has previously been used to regularize shapes for shape-based NRSfM approaches [10, 53, 59, 60], showing

greater expressibility and accuracy compared to low-rank models on large datasets. To our knowledge, we are the first to propose such a strategy for modeling trajectories.

Our neural trajectory prior can be easily integrated to problems outside the NRSfM domain, such as estimating long-term scene flow from lidar point clouds. Despite the remarkable progress on scene flow estimation given a pair of point clouds [21, 30, 37, 47, 62], achieving similar performance over longer sequences has still been challenging. We are interested in integrating sparse 3D lidar points across tens of frames back to a single reference frame, where the scene is dynamic with multiple objects. In addition, working with lidar point clouds has its unique challenges due to the sparsity and unstructured nature of the signals. This makes image-based multi-frame flow methods [16, 26, 27, 50] non-applicable. We demonstrate that our proposed neural trajectory prior is sufficient to regularize scene flow integration both spatially and temporally while outperforming naive Euler-based integration methods proposed by state-of-the-arts scene flow estimators [30, 36].

Contributions. We propose a general neural trajectory prior. It uses a bottleneck architecture to regularize trajectories and a coordinate MLPs to regularize spatio-temporal dependencies when dealing with dense problems. We validate the effectiveness of the bottleneck trajectory prior by achieving competitive results with NRSfM methods that use both shape and trajectory priors. Next, we show that it can be paired with existing neural shape priors to achieve state-of-the-art results across well-known sparse NRSfM benchmarks. Also, we show that our method outperforms a recently proposed neural-based method [53] on dense NRSfM benchmarks while being significantly faster. For lidar scene flow integration, we show that our approach produces better trajectories by modeling spatio-temporal information than if naively integrating pairwise flows from scene flow estimators. We demonstrate the applicability of our scene flow integration through point cloud densification.

2. Related works

Non-rigid structure from motion. We focus our discussion on trajectory-based approaches. Readers are referred to Jensen *et al*. [28] for a comprehensive survey of NRSfM methods. The seminal work of Akhter *et al*. [3] introduced a pure trajectory-based factorization method, which assumes that trajectories of different points can be represented as a linear combination of a small number of DCT bases. Other methods have explored the convolutional structure of trajectories [11, 65] as well as the assumption that different trajectories can be clustered into a small number of linear subspaces [31, 33, 34, 64]. In Sec. 3.1 we give a more detailed discussion of trajectory priors explored in literature.

Lidar scene flow estimation. We review state-of-the-art methods that estimate scene flow directly from point clouds. Most deep methods are based on full supervision from large-scale synthetic data and then fine-tuned on small real-world datasets [21, 37, 47, 61]. The scarcity of large-scale, real-world data with lidar scene flow annotations inspired the proposal of self-supervised methods [30, 39, 55, 62]. However, these methods still rely on large synthetic datasets to get adequate data prior knowledge. Recently, Pontes *et al*. [46] proposed a dataless method to regularize the scene flow using a graph Laplacian and a simple objective function that is optimized during runtime. Of particular interest to our work is the recent Neural Scene Flow Prior work [36] that proposed the use of a coordinate-based network to regularize scene flow implicitly during runtime. Such dataless methods have shown great generalizability and robustness to different data.

3. Background

3.1. Review of rank-based trajectory priors

Although a 3D trajectory is high-dimensional, a collection of trajectories from real-world scenes tends to have simpler structure, thus compressible to lower dimensions. Prior works have explored statistical trajectory priors by reasoning about the rank of trajectories. To motivate our approach, we go through different scene assumptions, and review several well-known rank-based trajectory priors.

Rigid body. Given a sequence of rigid motions $\mathbf{M}_t \in \mathbb{R}^{3\times4}$ and time index $t \in [1, F]$, the position of i-th point of a rigid body at time t is $\mathbf{s}_{t,i} = \mathbf{M}_t \tilde{\mathbf{s}}_i$, where $\tilde{\mathbf{s}}_i \in \mathbb{R}^4$ denotes the homogeneous coordinate at the canonical frame. The trajectory of i-th point is represented as a vector $\boldsymbol{\tau}_i = [\mathbf{s}_{1,i}^\top, \dots, \mathbf{s}_{T,i}^\top]^\top \in \mathbb{R}^{3F}$. The column-wise concatenation of P trajectories gives a matrix $\mathrm{T} \in \mathbb{R}^{3F\times P}$ whose maximum rank=4, since $\mathrm{T} = \mathbf{M}\tilde{\mathbf{S}}$, where $\mathbf{M} \in \mathbb{R}^{3F\times4}$, $\tilde{\mathbf{S}} \in \mathbb{R}^{4\times P}$ are row-wise and column-wise concatenation of $\mathbf{M}_t$'s and $\tilde{\mathbf{s}}_i$'s. When the rigid object is planar, rank(T) further reduces to 3 due to rank($\tilde{\mathbf{S}}$)=3.

Multi-rigid body. Based on the analysis of single rigid body, the maximum rank of T for K rigid bodies is $4K$ [12].

(Multi) non-rigid body. Motivated by the result from multi-rigid body, Bregler *et al*. [8] propose to assume T being low rank for non-rigid objects. However, the low rank assumption is inadequate when the non-rigid deformation is complex or the scene consists of multiple non-rigid objects [34]. Instead, a more realistic assumption is that points with similar motions are clustered into a low-dimensional subspace, while the whole trajectory space can still be high-dimensional. In other words, trajectories are drawn from a union of multiple low-dimensional linear subspaces [34, 64].

Dense non-rigid surface. Deformation of surfaces usually preserves local differential structure such as isometry and conformality [6, 44]. To provide extra conditions for the motion, piece-wise planarity [58] and infinitesimal pla-

narity [43] are assumed for the local geometry of surfaces. This implies that the trajectory subspaces for neighboring points on surfaces are low-dimensional, and the subspaces in neighboring spatial positions should be similar to each other. Based on this insight, Kumar *et al*. proposed to view the trajectory subspaces as points in a Grassmann manifold [22], and measure the similarity using the projection metric distance [31].

Remark. Prior arts [31, 33] consider the trajectories as discrete variables and requires complex notions to cluster trajectories and enforce spatial dependencies. In this work, we attempt to utilize the notion of neural simplicity biases to model a simpler and continuous trajectory prior.

3.2. Simplicity bias of MLP

Low-rank bias of bottleneck MLP. Studies have shown that deep MLPs are biased towards low rank outputs even if it is overparameterized [25]. This observation has been used as evidence to explain the good generalization ability of neural networks. This low-rank simplicity bias can be further strengthened by introducing bottleneck layers into the MLPs. Hence bottleneck auto-encoders and decoders are the modern choice for learning a compact representation of high-dimensional data [24, 29]. Recent NRSfM works used bottleneck MLP to represent and constrain 3D shapes [53, 60]. They showed competitive results compared to state-of-the-arts methods using complex mathematical notions such as union-of-subspaces. However, modeling 3D shape space is expensive when the data is dense, as shapes become extremely high dimensional which blows up the number of learnable parameters. In this work, we propose to instead model the trajectory space to take advantage of the fact that trajectories are lower dimensional compared to dense 3D shapes, thus more scalable to optimize.

Smoothness biase of coordinate MLP. Many real-world signals can be viewed as a continuous vector field, *e.g*., a 2D image can be represented as a continuous mapping from 2D pixel coordinates to RGB values. Recently, multi-layer perceptron (MLP) has been successfully applied to model continuous vector fields such as signed distance field [45], radiance field [38], and scene flow field [40]. Compared to discrete representations such as voxels, the MLP representation enables (i) cheaper memory cost for higher dimensional data; (ii) better quality for interpolation; (iii) simpler optimization with gradient descent. Moreover, optimizing MLPs by gradient descent biases torwards smooth solutions [7, 23, 52]. This enables its success on tasks that traditionally require handcrafted smoothness priors. In tasks such as scene flow estimation using lidar point clouds, Li *et al*. recently showed that using the inductive bias of MLP outperforms graph Laplacian-based priors [36]. Therefore, we choose to represent the trajectories as a vector field parameterized by MLPs.

4. Neural trajectory prior

The proposed prior is modeled as a continuous trajectory field, which maps a low-dimensional Euclidean coordinate to a continuous trajectory:

$$f_{\text{NT}} : \mathbf{p} \in \mathbb{R}^d \longrightarrow \tau \in \mathcal{F}_\tau, \tag{1}$$

where $\mathcal{F}_\tau$ denotes the set of functions spanned by K trajectory basis functions $\{\tau^k(t) : \mathbb{R} \rightarrow \mathbb{R}^3\}_{k=1}^K$. The dimension d of the input coordinate $\mathbf{p}$ depends on specific tasks. For lidar point cloud integration, $\mathbf{p}$ is 4D space-time position of a physical point. For dense NRSfM, $\mathbf{p}$ is 2D pixel coordinates of the reference image.

As illustrated in Fig. 1, the trajectory field is modeled as,

$$f_{\text{NT}}(\mathbf{p}) = \sum_{k=1}^{K} (f_\alpha \circ f_\varphi)_k(\mathbf{p})\, \tau^k, \tag{2}$$

where $f_\varphi: \mathbf{p} \in \mathbb{R}^d \rightarrow \boldsymbol{\varphi} \in \mathbb{R}^L$ is a vector field which outputs a low dimensional representation of trajectories, and $f_\alpha: \boldsymbol{\varphi} \in \mathbb{R}^L \rightarrow \boldsymbol{\alpha} \in \mathbb{R}^K$ decodes $\boldsymbol{\varphi}$ to a higher-dimensional weight vector $\boldsymbol{\varphi}$. Then the output trajectory is the linear combination of basis τ^k with weights produced by $(f_\alpha \circ f_\varphi)(\mathbf{p})$, where $\circ$ denotes function composition.

Finally, the trajectory basis functions are modeled with a single continuous function $f_\tau : t \in \mathbb{R} \rightarrow \mathbb{R}^{3K}$, which outputs the concatenation of the positions of each τ^k at time t. We explain the details of each functions as follow.

Code field f_φ. We use MLPs with ReLU activation to model f_φ, which results in a piecewise linear and Lipschitz continuous trajectory code field [63]. Recent advance in coordinate-based MLPs suggests encoding the input coordinates with sinusoidal functions (*i.e*., positional encoding) helps improve convergence on high-frequency details [38, 49, 54]. In our experiments, due to the motions being naturally of low frequency, an MLP without positional encoding is enough. Ablation of using positional encoding is provided in the supplementary. We set the MLP to have 4 hidden layers, each with 128 hidden units.

Similar to choosing rank or number of subspaces in previous trajectory reconstruction works, the dimension of the trajectory codes is a hyperparameter to choose based on the complexity of the scene. We find that a fixed dimension of 4 is already expressive enough for all experiments in this work. We follow the practice as in deterministic autoencoders [17] which brings $\|\boldsymbol{\varphi}\|_2^2$ as a regularization. It is shown to be effective to achieve similar accuracy as the code dimension increases.

Trajectory basis functions. The overcomplete trajectory basis functions $\{\tau^k(t) : \mathbb{R} \rightarrow \mathbb{R}^3\}_{k=1}^K$ are generated by a single temporal coordinate MLP f_τ. The time input t to the MLP is embedded by cosine positional encoding. The

frequencies of the encoding is sampled logarithmically between $[1, \pi F]$, so as to cover the complete spectrum of a trajectory if the number of input frames is F. We pick $K = 256$, which is greater than the rank of trajectories in most scenarios.

We also note that the trajectory basis τ^k produced by f_τ is not defined in the world coordinate as it was in Sec. 3.1, but shifted so that they start from the origin. Though the statistical properties analyzed previously still hold. In practise, we fill the starting positions of trajectories as zero and do not evaluate f_τ when $t = 0$.

Trajectory weight decoder f_α. We use a 128-width MLP with 4 layers to decode weights from the trajectory codes. Details are provided in the supplementary. The role of using a decoder here is different from that in generative modeling. We utilize the expressiveness of the non-linear representation of the decoder as well as the constraints brought by the bottleneck dimension. However, sampling the code space is not required in our task. Thus enforcing a prior distribution on the latent code [16, 51] does not affect our performance.

In the next sections, we validate the NT prior in different applications, *i.e.*, non-rigid structure motion in Sec. 5 and lidar scene flow integration in Sec. 6.

5. Non-rigid structure from motion

Task. NRSfM takes input from a sequence of 2D keypoint positions of a deforming object and outputs the 3D keypoint positions. Formally, denote the input 2D keypoint positions as a 2D measurement matrix $\mathbf{W} \in \mathbb{R}^{2F \times P}$, which is a row-wise concatenation of F number of matrices $\mathbf{w}_t \in \mathbb{R}^{2 \times P}$, each storing 2D keypoint positions of the target object at the t-th view. NRSfM algorithms solve for the 3D keypoint positions $\mathbf{s}_t \in \mathbb{R}^{3 \times P}$ and camera projection matrices $\mathbf{M}_t$'s.

The first clue to solve NRSfM is minimizing the 2D reconstruction cost. Assuming camera projection is weak perspective, the 2D reconstruction cost is formulated as

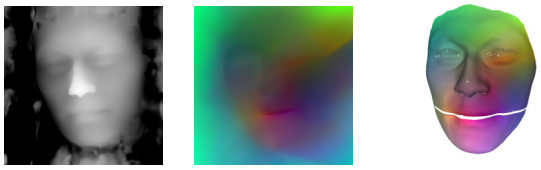
$$\mathcal{C}_{\text{2D recon.}} = \|\mathbf{W} - \mathbf{M}\mathbf{S}\|_F, \tag{3}$$

where $\mathbf{S} \in \mathbb{R}^{3F \times P}$ is the row-wise concatenation of unknown $\mathbf{s}_t$'s, and $\mathbf{W} = \text{blockdiag}(\mathbf{M}_1, \dots, \mathbf{M}_F)$. $\mathcal{C}_{\text{2D recon.}}$ alone provides fewer number of constraints than the number of unknowns (*i.e.*, $\mathbf{S}$, $\mathbf{W}$), thus designing extra constraints is required to solve NRSfM. Finally, we note that NRSfM uses no object specific knowledge (*e.g.*, kinematic constraints, shape templates) or supervised training. Thus it is different to learning-based tasks, *e.g.*, 2D-3D body pose estimation, which uses offline training on separate data.

5.1. Method

We keep a unified approach to solve both sparse and dense NRSfM by minimizing the cost

$$\mathcal{C}_{\text{2D recon.}} + \lambda_1 \mathcal{C}_{\text{smooth traj.}} + \lambda_2 \|\mathbf{\Phi}\|_2^2, \tag{4}$$



$\{ f_z$: depth field, f_φ: traj. code field. $\} \rightarrow$ canonical recon.

Figure 2. **Illustration of using NT prior for dense NRSfM.** We use coordinate MLPs to model the 2D depth map f_z and trajectory code field f_φ at the canonical frame. In all our dense NRSfM experiments, we choose the code dimension to be 3, thus we can directly visualize it in RGB. Right figure shows the reconstructed mesh colored by the RGB visualization of trajectory codes.

Table 1. **Sparse NRSfM results.** Results are report with normalized mean 3D error $\times$**100**. ● are state-of-the-art NRSfM methods, ● are our baselines, ● are our proposed methods. Bold and underlined numbers denotes the best and second best result.

	drink	pickup	yoga	stretch	dance	shark
#frames	1102	357	307	370	264	240
#points	41	41	41	41	41	91
● PTA [3]	2.87	19.39	12.43	10.35	24.26	29.33
● CSF2 [20]	2.27	17.91	11.79	11.36	18.77	11.17
● PND [35]	0.37	3.72	1.40	1.56	14.54	1.35
● BMM [14]	2.66	17.31	11.50	10.34	18.64	23.11
● BMM-v2 [32]	1.19	1.98	<u>1.29</u>	<u>1.44</u>	10.60	5.51
● PAUL [60]	0.47	2.03	1.71	1.62	10.22	**0.37**
● smooth traj.	1.42	7.87	2.13	2.34	13.11	13.54
● low rank (=12)	1.24	2.11	3.87	1.82	10.25	13.19
● NTP (=4)	<u>0.34</u>	<u>1.84</u>	2.10	1.51	<u>9.78</u>	9.42
● PAUL + NTP	**0.28**	**1.16**	**1.20**	**1.32**	**8.53**	<u>0.91</u>

where $\mathcal{C}_{\text{smooth traj.}} = \sum_{i=1}^{P} \|\nabla \boldsymbol{\tau}_i\|_2^2 + \|\nabla^2 \boldsymbol{\tau}_i\|_2^2$ encourages the trajectories to be smooth. $\boldsymbol{\tau}_i$ denotes the trajectory of i^{th} point. $\mathbf{\Phi} = [\boldsymbol{\varphi}_i, \dots, \boldsymbol{\varphi}_P]$ concatenates the trajectory codes. The difference in our approach for handling sparse and dense sequences is how $\mathbf{S}$ is constructed.

Sparse NRSfM. We pick the first view as the canonical frame, *i.e.* $\mathbf{M}_1 = [\mathbf{I}_2, \mathbf{0}]$, which means $\mathbf{s}_1 = [\mathbf{w}_1^\top, \mathbf{z}^\top]^\top$, where $\mathbf{z} \in \mathbb{R}^P$ is the unknown depth values for points in the first frame. Then $\mathbf{S} = \mathbf{1}_F \otimes \mathbf{s}_1 + [\boldsymbol{\tau}_1, \dots, \boldsymbol{\tau}_P]$, where $\otimes$ denotes Kronecker product. In test time, we minimize the cost (4) with respect to $\mathbf{z}$, $\mathbf{M}_2$, ..., $\mathbf{M}_F$, $\mathbf{\Phi}$ and $\boldsymbol{\theta}_\alpha$, $\boldsymbol{\theta}_\tau$ which is the network parameters of f_α, f_τ.

Dense NRSfM. The construction of $\mathbf{S}$ is almost identical to the sparse version, except that we assume elements of $\mathbf{z}$ and $\boldsymbol{\varphi}_i$'s are drawn from a continuous depth field f_z and the trajectory code field f_φ (see Fig. 2), rather than treating them as independent variables to optimize. Both f_z and f_φ are modeled as coordinate-based MLPs. Therefore, in test time, the variables to optimize are camera matrices $\mathbf{M}_2$, ..., $\mathbf{M}_F$ and network weights $\boldsymbol{\theta}_z$, $\boldsymbol{\theta}_\varphi$, $\boldsymbol{\theta}_\alpha$ and $\boldsymbol{\theta}_\tau$. To speedup optimization and reduce memory cost, we perform stochastic gradient descent by drawing random set of points from the continuous fields at each iteration. This is

Table 2. **Dense NRSfM results on synthetic face sequences**. Our method ranks among the top. In particular, it outperforms a recent neural-based approach N-NRSfM [53] with significantly less runtime. $*$ denotes results using different hyperparameters per sequence. Results are reported by the normalized mean 3D error.

	TB [4]	MP [41]	VA [15]	DSTA [13]	CDF [18]	CMDR [19]	GM [33]	JM [31]	SMSR [5]	PPTA [2]	EM-FEM [1]	N-NRSfM [53]	NTP(Ours)
traj. A	0.125	0.061	0.035	0.037	0.089	0.032	0.029	0.028	0.030	0.031	0.039	0.045 / 0.032*	0.031
traj. B	0.135	0.076	0.038	0.043	0.091	0.037	0.031	0.033	0.032	0.057	0.030	0.049 / 0.039*	0.034

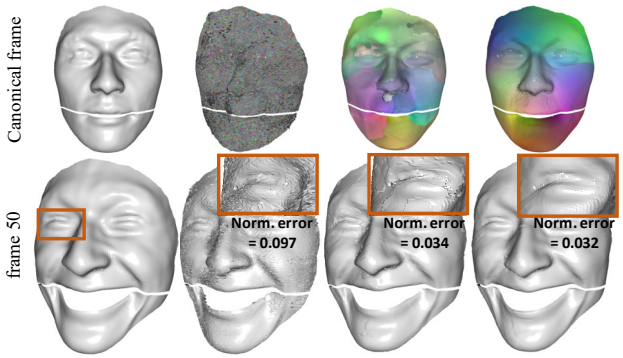


Figure 3. **Ablations with and without the continuous fields for dense NRSfM.** The first column shows the ground-truth 3D meshes from the 1st and 50th frame of the synthetic face traj. A sequence. Results with and without f_z, f_φ are shown on the rest of columns. On the top row, meshes are colored to visualize the recovered trajectory code vectors. The bottom row zooms in to highlight the geometric difference of the reconstructions. It shows that without spatial regularization from both fields, both the reconstruction and trajectory codes are extremely noisy. Only adding f_z without f_φ still exhibits discontinuities in the reconstruction due to the spatial discontinuities in the trajectory codes. Finally, the full method produced smooth and more accurate result.

not applicable for the shape-decoder-based approaches, *e.g.*, N-NRSfM [53]. See supplementary for details.

5.2. Experiments

Dataset. We report performance on standard benchmarks used in NRSfM literature: (i) sequences of articulated motions, drink, pickup, yoga, stretch, dance, shark, each consisting of 41-91 points per frame. (ii) two sequences of synthetic faces with 99 frames and 28,000 points per frame, captured by different camera trajectories denoted as traj. A and traj. B. [15]. In addition, we experiment with real world dense sequences, barn owl, real face and Kinect sequences.

Metrics. We employ normalized mean 3D error which is the most commonly used metric in NRSfM literature. To evaluate normalized 3D error, the reconstructed shape $\mathbf{S}$ is first aligned to the 3D ground truth $\mathbf{S}_{\text{GT}}$ by orthogonal Procrustean analysis. Then the error is calculated as $\|\mathbf{S}_{\text{align}} - \mathbf{S}_{\text{GT}}\|_F \;/\; \|\mathbf{S}_{\text{GT}}\|_F$.

Sparse NRSfM results. In Tab. 1, we collected several classic methods, PTA [3], CSF2 [20] and BMM [14], as well as the current state-of-the arts methods, PND [35], PAUL [60] and a revised version of BMM by Kumar [32]. Among those, PTA uses low-rank trajectory prior which is the most relevant work to our approach. Our method (denoted as NT prior) significantly outperforms PTA and is comparable to the state-of-the-arts methods except the shark sequence. In addition, motivated by Kumar's finding that low rank priors may be underrated due to insufficient algorithms used in prior works, we implemented a stronger low-rank baseline using the effective optimization procedure adapted from this work. In addition, we also created a baseline with simple smoothness prior. The details of these two baselines are given in the supplementary. Finally, we found that a simple combination of our method to the neural shape-based approach PAUL [60], yields the new state-of-the-art (technical details are in supplementary).

Figure 4. **Qualitative 3D reconstruction results on real non-rigid sequences.** Meshes are colored to visualize trajectory codes.

Dense NRSfM results. We performed ablations to our method by removing f_z and f_φ as shown in Fig. 3. We found that both f_z and f_φ helps improve the details of the reconstruction. See caption of Fig. 3 for more details.

In Tab. 2 we quantitatively compared our method to state-of-the-art dense NRSfM methods on synthetic face sequences. Our method ranks among the top methods, with only $\sim$0.002 behind the top method JM [31]. We formed a closer comparison to N-NRSfM [53] which also uses a neural prior but defined on the shape space. N-NRSfM is sensitive to the hyperparameters and needs sequence-specific tuning to be able to give comparable results to our default settings. More importantly, thanks to the scalability of modeling in the trajectory space, our model is much smaller compared to N-NRSfM and runs much faster. It takes $\sim$15 minutes for our method on a single RTX2080 GPU while N-NRSfM runs for 3+ hours.

Finally, we show qualitative results on real sequences in Fig. 4. More details are provided in the supplementary.

6. Lidar scene flow integration

Task. Given a sequence of sparse lidar point clouds capturing dynamic objects, the goal is to estimate the trajectory for each point across the point clouds. Estimating point trajectories enables practical applications for annotating and processing lidar data. For example, we could register each point cloud directly to a specific time frame, thus performing point cloud densification. A naive solution for this task is to integrate pairwise scene flows estimated by an off-the-shelf method. However, such an approach has the following challenges: (i) because the lidar points are sparse, tracing scene flow across a long sequence most likely will end up in empty/sparser regions causing trajectory drifts; (ii) scene flow integration using pairwise scene flow does not capture the temporal statistics of the scene—only spatial information is used during the integration.

6.1. Method

The trajectory prior f_{NT} is optimized to estimate the full trajectory for every point in a sequence of lidar input. The input points are represented using 4D coordinates, denoting the spatial position and the frame number it is captured. Feeding a point $\mathbf{p}=(x, y, z, t)$ to f_{NT} yields a trajectory represented as a sequence of 3D positions, *i.e.*, $\boldsymbol{\tau}=[\tau_1, \tau_2, \ldots, \tau_F]$. We note that the 3D positions τ_t are defined in the trajectory space, with an offset to the camera coordinate. Thus the estimated positions for $\mathbf{p}$ at another frame t' is $\mathbf{p}'=g_{t'}(\mathbf{p})=(\tau_{t'}-\tau_t+(x, y, z), t')$. Since $\mathbf{p}'$ and $\mathbf{p}$ are 4D positions of the same physical point, thus $\boldsymbol{\tau}'=f_{\text{NT}}(\mathbf{p}')$ should equals to $\boldsymbol{\tau}$. This motivates a cyclic consistency constraint to reduce drift as

$$\mathcal{C}_{\text{con.}}(\mathbf{p}, t') = \|\boldsymbol{\tau} - \boldsymbol{\tau}'\|_2^2. \tag{5}$$

At runtime, we optimize the pairwise truncated Chamfer distance with the consistency cost

$$\sum_{t=1}^{F} \mathcal{C}_{\text{CD}}(\mathcal{P}_{t\pm1}, \mathcal{P}_{t\to t\pm1}) + \sum_{\mathbf{p}\in\mathcal{P}_t,\ t'\in[1,F]} \mathcal{C}_{\text{con.}}(\mathbf{p}, t'), \tag{6}$$

where $\mathcal{P}_t$ is the set of points at frame t, and $\mathcal{P}_{t\to t'}=\{g_{t'}(\mathbf{p}) \mid \mathbf{p}\in\mathcal{P}_t\}$ is the points warped from frame t to t' according to the trajectories given by f_{NT}. Due to the rapid change of sparse point clouds, we only evaluate $\mathcal{C}_{\text{CD}}$ between neighboring ±1 frames. Longer temporal distance is too challenging for the truncated Chamfer distance.

6.2. Experiment

Datasets. We used the autonomous driving dataset Argoverse [9], which is a large-scale dataset with challenging dynamic motions of various object categories. We used the official tracking validation dataset to create our own trajectory dataset. In each scene, we chose the first 25 frames sampled at 10 Hz as a long trajectory. We also used the ground-truth poses of objects in the scene and the ground-truth ego motions of the autonomous vehicle (AV) to create pseudo-ground-truth trajectories.

Table 3. **Trajectory reconstruction results on the Argoverse dataset.** We report accuracy, outliers, and Chamfer distance results for all three methods with different integration strategies. We evaluated the methods using the entire point cloud scene (full) and only on dynamic regions (dyn.). Our method are more accurate despite dynamic or rigid motions in the scene. Note that for Acc_0.5, Acc_1, and Out., we reported for 24 frame intervals. For Chamfer distance, we only reported for the full scene due to the number of points for dynamic regions is not consistent across frames. ↑ means higher is better and ↓ means lower is better.

	FlowStep3D [30] (KNN Int.)		NSFP [36] (KNN Int.)		NSFP [36] (Euler Int.)		NTP (Ours)	
	full	dyn.	full	dyn.	full	dyn.	full	dyn.
Acc_0.5 (%) ↑	5.25	62.14	45.18	63.54	45.28	63.56	**52.28**	**69.49**
Acc_1 (%) ↑	6.81	62.49	59.86	70.25	59.52	69.70	**69.88**	**73.55**
Out. (%) ↓	87.20	36.19	28.90	23.21	29.33	23.69	**20.95**	**21.77**
cd-10 (m) ↓	4.27	—	5.35	—	4.43	—	**2.79**	—
cd-24 (m) ↓	12.68	—	21.30	—	14.09	—	**9.77**	—

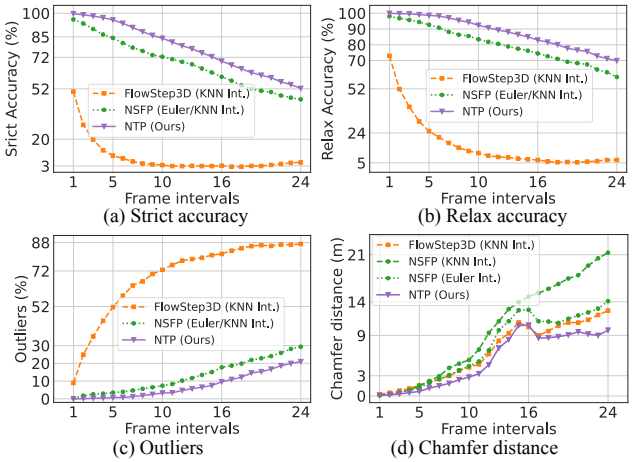
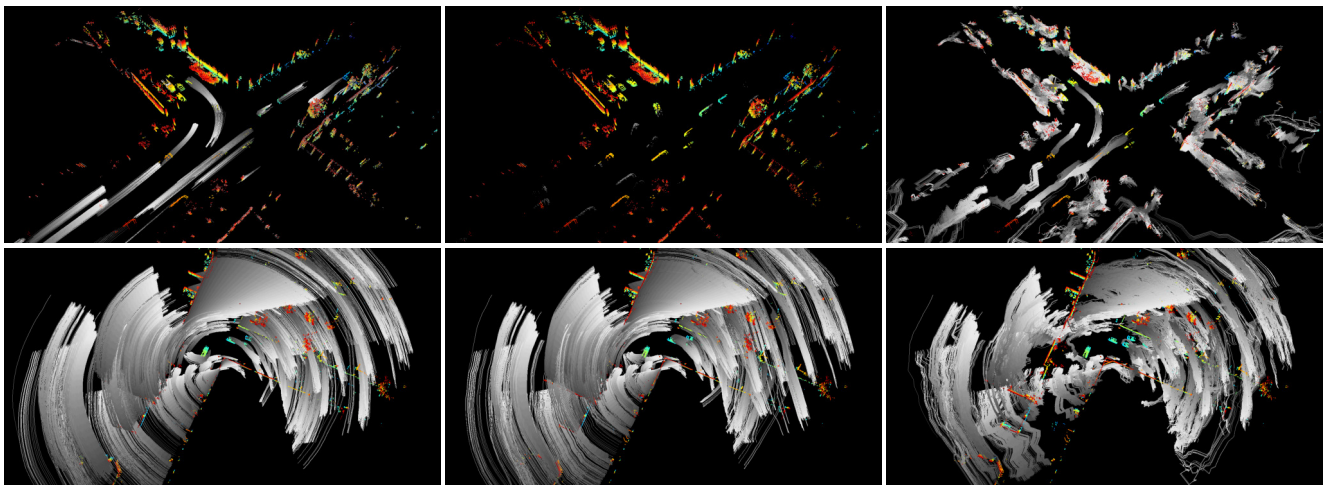


Figure 5. **Trajectory accuracy at different frame intervals.** Our method achieved the best accuracy across all frame intervals.

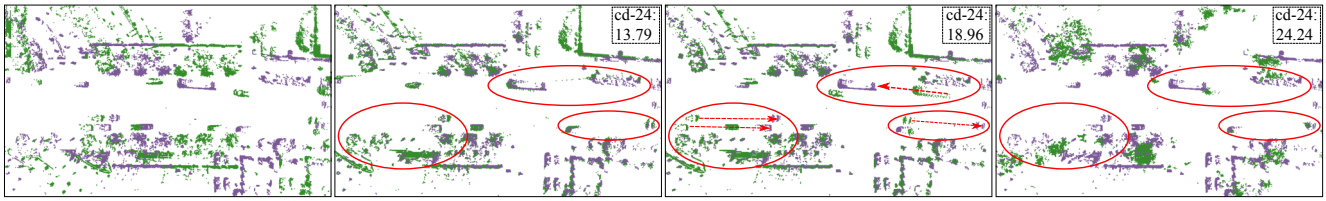
Metrics. In all experiments, we reported: **1) Chamfer distance (cd-n)** between the deformed point cloud and the original point cloud as a metric. We chose cd-10, cd-24 to represent the Chamfer distance from 1st frame to 11th frame, 1st frame to 25th frame, respectively. **2) Strict accuracy (Acc_0.5)** is the percentage that the error between the predicted trajectory and the ground truth is below 0.5 meters. **3) Relax accuracy (Acc_1)** indicates the percentage that the error is below 1.0 meter. **4) Outliers (Out.)** counts the percentage that the error is larger than 3.0 meters.

Baselines. We have also implemented the learning-based scene flow method FlowStep3D [30] and the non-learning-based method Neural Scene Flow Prior [36] as our baselines. **1) FlowStep3D** is a learning method which uses the



(a) NTP (Ours) (b) NSFP (Euler Int.) (c) FlowStep3D (KNN Int.)

Figure 6. **Visual trajectory reconstructions on the Argoverse dataset.** Given a dynamic driving sequence of 25 lidar frames sampled at 10 Hz, we applied our method to estimate the trajectories of each point—the long-term flow of a point from the past (1st) to the current (25th) frame. The first row shows a scene where the AV is stopped at an intersection waiting for the crossing traffics. The second row shows the AV making a right turn. We only show the point cloud of the current 25th frame. The gray lines represent the estimated trajectories for each point. Darker gray represents the most recent motion, and lighter gray represents the older motion. Note how the trajectories estimated by our method are smooth. The NSFP with Euler integration lost tracks for most dynamic objects (*e.g.*, dynamic cars shown in the first row). Because FlowStep3D only outputs sparse scene flows, its integration leads to noisy trajectories and drifts over time.



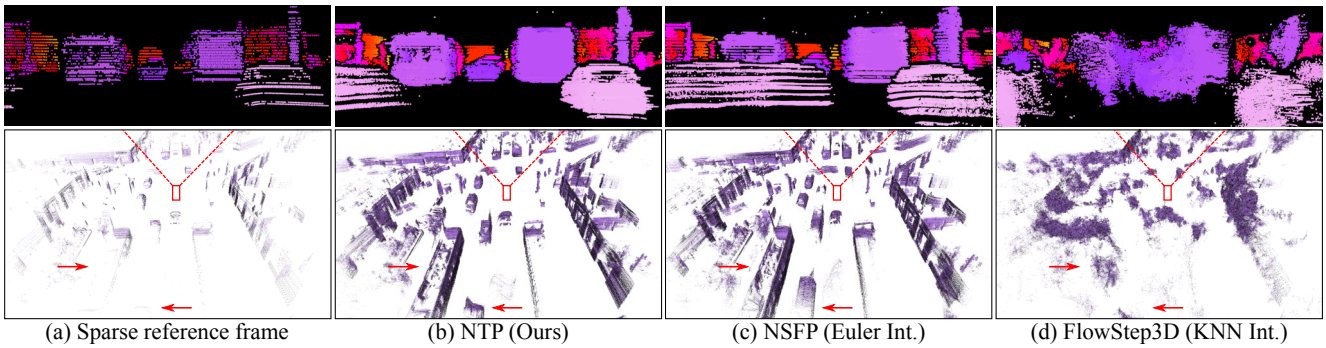
(a) Before alignment (b) NTP (Ours) (c) NSFP (Euler Int.) (d) FlowStep3D (KNN Int.)

Figure 7. **Comparison on point cloud warping on the Argoverse dataset.** (a) The **purple** point cloud is the reference frame (25th frame), and the **green** point cloud is the source frame (1st frame). We want to warp the source (1st) to the reference (25th) frame using different methods to estimate the trajectory field. Note that this is a registration of two far-away point clouds where objects drastically move. The alignment results of different methods are shown in (b, c, d). The Chamfer distance metric (cd-24) is shown on the top right. The **red** circles highlight regions where our method achieved significantly better registrations for challenging dynamic objects. The intuition is that our trajectory field is better constrained and allows for warps between distant frames. The **red** dashed arrows in (c) show miss alignments.

PointNet++ [48]-based features to iteratively find the non-rigid motions between a pair of point clouds. We used the official pre-trained model on the FlyingThings3D dataset provided by the authors. **2) Neural Scene Flow Prior (NSFP)** is a non-learning-based method that uses an MLP to directly optimize the scene flow from raw point clouds. We took the official implementation provided by the authors.

Trajectory reconstruction. We used three different integration methods to reconstruct the trajectories of a dynamic lidar sequence using estimated pairwise scene flows: **1) Our method (Ours)** to directly query a trajectory from the estimated trajectory field of previous frames. **2) Euler integration (Euler Int.)** as used in NSFP. Because NSFP is a continuous representation, we input the queried point cloud to the predicted scene flow field to get current motions, and integrate the long term trajectory across the entire point cloud sequence using Euler integration. **3) KNN integration (KNN Int.)** uses the k-nearest-neighbors algorithm to find the correspondences between two consecutive frames, and integrates the per-pair flows to a long term trajectory according to these correspondences. More details are in the supplementary material.

We compared our method with NSFP and FlowStep3D on the Argoverse dataset and summarized the results in Fig. 5 and Tab. 3. Note that for NSFP, we did not report direct trajectory query results, since NSFP can easily overfit to a specific scene using direct optimization, which is unfair to compare to. The deep learning-based FlowStep3D is a discrete flow prediction that is unable to fit a flow field and do Euler integration. The metric cd-n only measures the distance between the deformed point cloud and the reference point cloud, which is not reliable for measuring the



(a) Sparse reference frame (b) NTP (Ours) (c) NSFP (Euler Int.) (d) FlowStep3D (KNN Int.)

Figure 8. **Point cloud densification on the Argoverse dataset.** Given 25 point cloud frames of a dynamic driving scene, the task is to accumulate all frames against a reference frame. In our case, we accumulated the past 24 frames against the current frame (25th). The first column shows the reference point cloud (bottom) and projection onto a depth map (top). Note how sparse a single lidar frame is. The other columns show the densification results from different methods. The **red** rectangles indicate the position of the autonomous vehicle (AV), and dashed lines show the viewpoint of the depth maps. The **red** arrows highlight some densified objects. Our method produced denser point clouds with better geometry. NSFP had noticeable drifts and loss of tracks. FlowStep3D produced noisy results.

3D geometry. For example, we have found that although FlowStep3D had a low cd-24 in some cases, the predicted deformed shapes are noisy. We complement these metrics with accuracy and outliers to directly compare with the pseudo-ground-truth labels. Our method (NTP) achieved comparable—if not much better—results across all metrics. We also found that our method has impressive results when handling both rigid regions and non-rigid regions in the scene.

We show visual comparison results on Argoverse in Fig. 6. Our method produced reliable, clean trajectories for all examples across 25 frames. In the first row, the NSFP barely produced any trajectories for the dynamic cars. And in the second row, the NSFP misrepresented the trajectories for turning vehicles. While for the deep learning-based FlowStep3D, the predicted trajectories were messy and distorted for either rigid motions or non-rigid motions. Fig. 6 clearly shows the advantage of using a continuous trajectory prior over a continuous scene flow prior or a discrete flow estimation, especially when dealing with long term trajectories with a lot of dynamic objects and large rotations.

Point cloud densification. To densify point clouds through a long term sequence, we used the same strategies (*i.e.*, direct trajectory query for our method, Euler integration for NSFP, and KNN integration for FlowStep3D) to first generate trajectories between the query frame and the reference frame pairs. For example, we generated trajectory from 1st frame to 25th frame, 2nd frame to 25th frame, 3rd frame to 25th frame, *etc*. After generating all trajectories for point cloud pairs, we can easily get all deformed point clouds before the 25th frame and integrate all points to get the final densified point cloud at the 25th frame.

We show a detailed registration result of the 1st frame to the 25th frame in Fig. 7. The registration result is consistent with our findings that our method outperformed in the challenging scene, but NSFP was inferior dealing with large motions, FlowStep3D only captured scarce and noisy 3D points. An example of dense depth and dense point clouds of an Argoverse dynamic scene is shown in Fig. 8. Our method gave clean depth with rigid background as well as non-rigid moving objects in detail without large noise. The results of NSFP are worse than ours, and FlowStep3D completely failed the task. This indicates that one useful application of our method is reliable point cloud densification through a long trajectory. The application can extend to generating dense depths or dense lidar HD maps.

7. Limitations

The limitations of our work are: (i) it requires runtime optimization using GPUs, thus not yet applicable to real-time applications; (ii) despite our method performs well on tested NRSfM benchmarks, we note that it is still an extremely challenging problem. Our method may fail for sequences with strong ambiguities between camera and object motions; (iii) like other NRSfM methods, we require long-term 2D correspondences provided as input, which may be difficult to reliably acquire in real world. Combining trajectory reconstruction with multi-frame 2D flows is a potential future work based on our current result; See supplementary for examples of failure cases. (iv) for lidar scene flow integration, we experimented with scenes with up to 25 frames. Longer sequences might cause trajectory drifts. (vi) limited by the Chamfer distance cost, our method might suffer when using extremely sparse point clouds.

8. Conclusion

We presented a neural trajectory prior for solving under-constrained low-level vision tasks, *e.g.*, NRSfM and lidar scene flow integration. Our results are promising and competitive to many state-of-the-art methods for NRSfM and lidar scene flow integration problems. Our simple formulation can be potentially extended to self-supervised learning for depth estimation and motion prediction.

References

[1] Antonio Agudo, JMM Montiel, Lourdes Agapito, and Begoña Calvo. Online dense non-rigid 3D shape and camera motion recovery. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2014. 5

[2] Antonio Agudo and Francesc Moreno-Noguer. A scalable, efficient, and accurate solution to non-rigid structure from motion. *Computer Vision and Image Understanding (CVIU)*, 167:121–133, 2018. 5

[3] Ijaz Akhter, Yaser Sheikh, Sohaib Khan, and Takeo Kanade. Nonrigid structure from motion in trajectory space. In *Neural Information Processing Systems (NeurIPS)*, pages 41–48, 2009. 1, 2, 4, 5

[4] Ijaz Akhter, Yaser Sheikh, Sohaib Khan, and Takeo Kanade. Trajectory space: A dual representation for nonrigid structure from motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 33(7):1442–1456, 2010. 5

[5] Mohammad Dawud Ansari, Vladislav Golyanik, and Didier Stricker. Scalable dense monocular surface reconstruction. pages 78–87. IEEE, 2017. 5

[6] Adrien Bartoli, Yan Gérard, Franccois Chadebecq, and Toby Collins. On template-based reconstruction from a single view: Analytical solutions and proofs of well-posedness for developable, isometric and conformal surfaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2026–2033. IEEE, 2012. 2

[7] Alberto Bietti and Julien Mairal. On the inductive bias of neural tangent kernels. *Neural Information Processing Systems (NeurIPS)*, 2019. 3

[8] Christoph Bregler. Recovering non-rigid 3D shape from image streams. Citeseer. 2

[9] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3D tracking and forecasting with rich maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8748–8757, 2019. 6

[10] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5939–5948, 2019. 1

[11] Nathaniel Chodosh and Simon Lucey. When to use convolutional neural networks for inverse problems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[12] Joao Paulo Costeira and Takeo Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision (IJCV)*, 29(3):159–179, 1998. 2

[13] Yuchao Dai, Huizhong Deng, and Mingyi He. Dense non-rigid structure-from-motion made easy—a spatial-temporal smoothness based solution. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages 4532–4536. IEEE, 2017. 5

[14] Yuchao Dai, Hongdong Li, and Mingyi He. A simple prior-free method for non-rigid structure-from-motion factorization. *International Journal of Computer Vision (IJCV)*, 107(2):101–122, 2014. 4, 5

[15] Ravi Garg, Anastasios Roussos, and Lourdes Agapito. Dense variational reconstruction of non-rigid surfaces from monocular video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1272–1279, 2013. 5

[16] Ravi Garg, Anastasios Roussos, and Lourdes Agapito. A variational approach to video registration with subspace constraints. *International Journal of Computer Vision (IJCV)*, 104(3):286–314, 2013. 2, 4

[17] P. Ghosh, M. S. M. Sajjadi, A. Vergari, M. J. Black, and B. Schölkopf. From variational to deterministic autoencoders. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Apr. 2020. 3

[18] Vladislav Golyanik, Torben Fetzer, and Didier Stricker. Introduction to coherent depth fields for dense monocular surface recovery. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2017. 5

[19] Vladislav Golyanik, André Jonas, and Didier Stricker. Consolidating segmentwise non-rigid structure from motion. In *Journal of Machine Vision and Applications*, pages 1–6. IEEE, 2019. 5

[20] Paulo FU Gotardo and Aleix M Martinez. Non-rigid structure from motion with complementary rank-3 spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3065–3072. IEEE, 2011. 4, 5

[21] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. HPLFlownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3254–3263, 2019. 2

[22] Jihun Hamm and Daniel D Lee. Grassmann discriminant analysis: a unifying view on subspace-based learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 376–383, 2008. 3

[23] Reinhard Heckel and Mahdi Soltanolkotabi. Compressive sensing with un-trained neural networks: Gradient descent finds a smooth approximation. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 4149–4158. PMLR, 2020. 3

[24] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. 3

[25] Minyoung Huh, Hossein Mobahi, Richard Zhang, Brian Cheung, Pulkit Agrawal, and Phillip Isola. The low-rank simplicity bias in deep networks. *arXiv preprint arXiv:2103.10427*, 2021. 3

[26] Junhwa Hur and Stefan Roth. Self-supervised multi-frame monocular scene flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2684–2694, 2021. 2

[27] Michal Irani. Multi-frame optical flow estimation using subspace constraints. In *Proceedings of the International Conference on Computer Vision (ICCV)*, volume 1, pages 626–633. IEEE, 1999. 2

[28] Sebastian Hoppe Nesgaard Jensen, Mads Emil Brix Doest, Henrik Aanæs, and Alessio Del Bue. A benchmark and evaluation of non-rigid structure from motion. *International Journal of Computer Vision (IJCV)*, 129(4):882–899, 2021. 2

[29] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *STAT*, 1050:1, 2014. 3

[30] Yair Kittenplon, Yonina C Eldar, and Dan Raviv. FlowStep3D: Model unrolling for self-supervised scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4114–4123, 2021. 2, 6

[31] Suryansh Kumar. Jumping manifolds: Geometry aware dense non-rigid structure from motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5346–5355, 2019. 1, 2, 3, 5

[32] Suryansh Kumar. Non-rigid structure from motion: Prior-free factorization method revisited. In *Proceedings of the IEEE Workshop on Applications of Computer Vision (WACV)*, 2020. 4, 5

[33] Suryansh Kumar, Anoop Cherian, Yuchao Dai, and Hongdong Li. Scalable dense non-rigid structure-from-motion: A grassmannian perspective. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 254–263, 2018. 1, 2, 3, 5

[34] Suryansh Kumar, Yuchao Dai, and Hongdong Li. Multi-body non-rigid structure-from-motion. In *Proceedings of the International Conference on 3D Vision (3DV)*, pages 148–156. IEEE, 2016. 1, 2

[35] Minsik Lee, Jungchan Cho, Chong-Ho Choi, and Songhwai Oh. Procrustean normal distribution for non-rigid structure from motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1280–1287, 2013. 4, 5

[36] Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Neural scene flow prior. In *Neural Information Processing Systems (NeurIPS)*, 2021. 1, 2, 3, 6

[37] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. FlowNet3D: Learning scene flow in 3D point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 529–537, 2019. 2

[38] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 405–421. Springer, 2020. 1, 3

[39] Himangi Mittal, Brian Okorn, and David Held. Just go with the flow: Self-supervised scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11177–11185, 2020. 2

[40] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4D reconstruction by learning particle dynamics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5379–5389, 2019. 1, 3

[41] Marco Paladini, Alessio Del Bue, João Xavier, Lourdes Agapito, Marko Stošić, and Marija Dodig. Optimal metric projections for deformable and articulated structure-from-motion. *International Journal of Computer Vision (IJCV)*, 96(2):252–276, 2012. 5

[42] Shaifali Parashar, Daniel Pizarro, and Adrien Bartoli. Robust isometric non-rigidstructure-from-motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*. 1

[43] Shaifali Parashar, Daniel Pizarro, and Adrien Bartoli. Isometric non-rigid shape-from-motion in linear time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 3

[44] Shaifali Parashar, Mathieu Salzmann, and Pascal Fua. Local non-rigid structure-from-motion from diffeomorphic mappings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1, 2

[45] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, 2019. 1, 3

[46] Jhony Kaesemodel Pontes, James Hays, and Simon Lucey. Scene flow from point clouds with or without learning. In *Proceedings of the International Conference on 3D Vision (3DV)*. IEEE, 2020. 2

[47] Gilles Puy, Alexandre Boulch, and Renaud Marlet. FLOT: Scene Flow on Point Clouds Guided by Optimal Transport. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 2

[48] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Neural Information Processing Systems (NeurIPS)*, pages 5105–5114, 2017. 7

[49] Sameera Ramasinghe and Simon Lucey. Beyond periodicity: Towards a unifying framework for activations in coordinate-mlps. 3

[50] Zhile Ren, Orazio Gallo, Deqing Sun, Ming-Hsuan Yang, Erik B Sudderth, and Jan Kautz. A fusion approach for multi-frame optical flow estimation. In *Proceedings of the IEEE Workshop on Applications of Computer Vision (WACV)*, pages 2077–2086. IEEE, 2019. 2

[51] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015. 4

[52] Basri Ronen, David Jacobs, Yoni Kasten, and Shira Kritchman. The convergence rate of neural networks for learned functions of different frequencies. *Neural Information Processing Systems (NeurIPS)*, 32:4761–4771, 2019. 3

[53] Vikramjit Sidhu, Edgar Tretschk, Vladislav Golyanik, Antonio Agudo, and Christian Theobalt. Neural dense non-rigid structure from motion with latent space constraints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 1, 2, 3, 5

[54] Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*, 2020. 3

[55] Ivan Tishchenko, Sandro Lombardi, Martin Oswald, and Marc Pollefeys. Self-supervised learning of non-rigid residual flow and ego-motion. In *Proceedings of the International Conference on 3D Vision (3DV)*, 2020. 2

[56] Lorenzo Torresani, Aaron Hertzmann, and Chris Bregler. Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(5):878–892, 2008. 1

[57] Jack Valmadre and Simon Lucey. General trajectory prior for non-rigid reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1394–1401. IEEE, 2012. 1

[58] Christoph Vogel, Konrad Schindler, and Stefan Roth. Piecewise rigid scene flow. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1377–1384, 2013. 2

[59] Chaoyang Wang, Chen-Hsuan Lin, and Simon Lucey. Deep nrsfm++: Towards unsupervised 2d-3d lifting in the wild. In *2020 International Conference on 3D Vision (3DV)*, pages 12–22, 2020. 1

[60] Chaoyang Wang and Simon Lucey. PAUL: Procrustean autoencoder for unsupervised lifting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 434–443, 2021. 1, 3, 4, 5

[61] Zirui Wang, Shuda Li, Henry Howard-Jenkins, Victor Prisacariu, and Min Chen. FlowNet3D++: Geometric losses for deep scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 91–98, 2020. 2

[62] Wenxuan Wu, Zhi Yuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. PointPWC-Net: Cost volume on point clouds for (self-) supervised scene flow estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 88–107. Springer, 2020. 2

[63] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into deep learning. *CoRR*, abs/2106.11342, 2021. 3

[64] Yingying Zhu, Dong Huang, Fernando De La Torre, and Simon Lucey. Complex non-rigid motion 3D reconstruction by union of subspaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1542–1549, 2014. 2

[65] Yingying Zhu and Simon Lucey. Convolutional sparse coding for trajectory reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 37(3):529–540, 2013. 2