Learned Trajectory Embedding for Subspace Clustering

Yaroslava Lochman[1] Carl Olsson[1,2] Christopher Zach[1]

[1]Chalmers University of Technology [2]Lund University

Abstract

[1]*Clustering multiple motions from observed point trajectories is a fundamental task in understanding dynamic scenes. Most motion models require multiple tracks to estimate their parameters, hence identifying clusters when multiple motions are observed is a very challenging task. This is even aggravated for high-dimensional motion models. The starting point of our work is that this high-dimensionality of motion model can actually be leveraged to our advantage as sufficiently long trajectories identify the underlying motion uniquely in practice. Consequently, we propose to learn a mapping from trajectories to embedding vectors that represent the generating motion. The obtained trajectory embeddings are useful for clustering multiple observed motions, but are also trained to contain sufficient information to recover the parameters of the underlying motion by utilizing a geometric loss. We therefore are able to use only weak supervision from given motion segmentation to train this mapping. The entire algorithm consisting of trajectory embedding, clustering and motion parameter estimation is highly efficient. We conduct experiments on the Hopkins155, Hopkins12, and KT3DMoSeg datasets and show state-of-the-art performance of our proposed method for trajectory-based motion segmentation on full sequences and its competitiveness on the occluded sequences. Project page: https://ylochman.github.io/trajectory-embedding.*

1. Introduction

Multi-model fitting is a classical problem in computer vision, and a typical example is the task of estimating individual motions from point trajectories observed in images containing multiple moving (and potentially deforming) objects. A trajectory is a sequence of tracked point coordinates extracted in different frames and is therefore also a point in a high-dimensional space. The commonly used union-of-subspace models [61] assume that point trajectories originating from the same object/motion are samples from the same (unknown) low dimensional subspace. The inference problem thus becomes that of simultaneously determining the parameters of underlying subspaces, the trajectory assignments to subspaces, and in practice also denoising of the data. Specifically, given an observation matrix $\mathtt{M}_{2F\times P}$ of point trajectories

$$\mathtt{M}_{2F\times P} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \dots & \mathbf{x}_P \\ | & & | \end{pmatrix}, \tag{1}$$

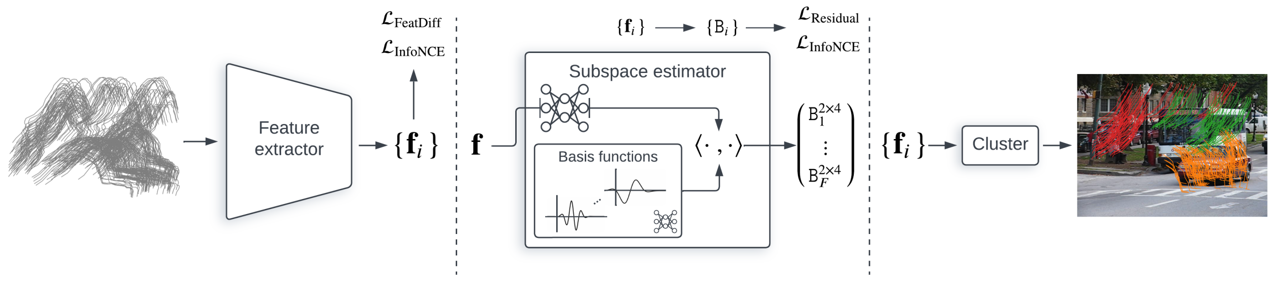
where F is the number of frames and P is the number of points, the goal is to find a $P \times P$ permutation matrix $\mathtt{P}_\pi$ that orders the trajectories into sub-matrices $\{\mathtt{X}_i\}_{i=1}^{c}$ corresponding to different motions. Each motion is represented by a $2F \times k_i$ basis matrix $\mathtt{B}_i$ whose columns span the subspace and therefore $\mathtt{X}_i = \mathtt{B}_i\mathtt{C}_i^T$, where $\mathtt{C}_i$ is a $k_i \times P_i$ shape matrix. Hence

$$\mathtt{M}_{2F\times P}\mathtt{P}_\pi \approx \begin{bmatrix}\mathtt{B}_1\mathtt{C}_1^\top & \dots & \mathtt{B}_c\mathtt{C}_c^\top\end{bmatrix}, \tag{2}$$

where $\sum_{i=1}^{c} P_i = P$, and ranks k_i are assumed to be small. In this work we assume that the ranks $\{k_1, \dots, k_c\}$ are known which is a common scenario under multiple rigid motions [49]. While estimating a single motion model $\mathtt{B}_i$ from a cluster of trajectories belonging to the same motion is relatively easy, the combination of cluster assignment and model estimation turns inference into a complicated chicken and egg problem. Further complications arise due to occlusion and partial observations which make it unlikely that we can observe all the full trajectories in all cases.

In this work we seek to learn a feature representation that allows us to solve the clustering problem separately from the motion estimation. For this purpose we train a simple feed forward network (with the overall structure depicted in Fig. 1) taking a trajectory as input and returning a corresponding feature embedding. Using a metric learning loss we ensure that the features are close if and only if corresponding trajectories belong to the same motions. Our approach allows us to input trajectories of varying lengths and different starting and ending times, making it possible to cluster trajectories using their features without completing any missing data. In summary our main contributions are:

[1]This work was supported by the Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP), funded by Knut and Alice Wallenberg Foundation.



$\mathcal{L}_{\text{FeatDiff}}$
$\mathcal{L}_{\text{InfoNCE}}$
Feature extractor
$\{\mathbf{f}_i\}$
$\mathbf{f}$
$\{\mathbf{f}_i\} \longrightarrow \{\mathtt{B}_i\} \longrightarrow$
$\mathcal{L}_{\text{Residual}}$
$\mathcal{L}_{\text{InfoNCE}}$
Subspace estimator
Basis functions
$\langle\cdot,\cdot\rangle$
$\mathtt{B}_1^{2\times4}$
$\mathtt{B}_F^{2\times4}$
$\{\mathbf{f}_i\}$
Cluster

Figure 1. Schematic overview of the proposed approach. Input trajectories are passed through the feature extractor to obtain $\{\mathbf{f}_i\}$. The subspace estimator generates the time-dependent basis using coordinate-MLP, infers basis coefficients from $\mathbf{f}$, and outputs subspace $\mathtt{B}$, where each sub-matrix $\mathtt{B}_t \in \mathbb{R}^{2\times4}$ is computed via linear combination of basis functions. $\mathcal{L}_{\text{InfoNCE}}$ is the contrastive loss that uses ground truth cluster assignments, $\mathcal{L}_{\text{Residual}}$ computes the trajectory-to-subspace geometric error, and $\mathcal{L}_{\text{FeatDiff}}$ computes the feature difference between input and reconstructed trajectory. For motion segmentation, the trajectories are clustered in a feature space using $\{\mathbf{f}_i\}$.

- We present a method to map point trajectories to discriminative feature representations, which can be seen as compact descriptions of the underlying generating motion.
- We show that clustering of trajectories can be performed reliably solely using those features, achieving state-of-the-art performance without the need for joint motion estimation and cluster assignments.
- The proposed feature extraction network is trained to be invariant with respect to random occlusions and pixel noise by utilizing a novel generative model that implicitly performs per-trajectory model estimation.
- We present and evaluate different ways to use the proposed network, such as trajectory clustering, trajectory completion and subspace estimation.

1.1. Related Work

Within geometric vision the traditionally dominant model-fitting approaches are robust statistical methods based on RANSAC [15]. In case of low dimensional models the approach provides a simple way of effectively exploring the search space. Iterative RANSAC [48], which sequentially finds models and removes their supporting inliers, is a simple option that handles settings with multiple models. It has been observed that the greedy nature of the approach makes it inefficient when a large number of models is present [6, 12, 21, 62]. In contrast, MultiRANSAC [62] simultaneously handles multiple models using iteratively fused consensus sets, and CONSAC [25] increases the efficiency of sampling by learning a sampling strategy based on previous detections. Energy minimization is used in [6, 12, 21] to select models among random proposals that jointly explain the data well. In [7] the hard point-to-model assignment is dropped in favour of progressively determining dominant models which enables clustering in consensus space.

As an alternative to consensus some works use preference [31, 32, 46]. The J-linkage approach [46] uses affinities between data and a set of models, followed by agglomerative clustering to assign the data to models. The T-linkage method [32] is based on a continuous relaxation of the binary assignment problem. In [31] the preference representation is used in a robust PCA formulation [10] while [33] uses a set cover approach.

Low dimensional geometric models have also been used to group longer point tracks, e.g. in [56] geometric two-view relations are used to create pairwise affinities for a spectral clustering stage. Segmentations computed from images pairs are combined through an averaging approach in [3, 4]. We note that for these local methods motion-specific properties should be observable across all image pairs in order to provide a largely coherent set of clusters.

Subspace models are commonly occurring in geometric computer vision tasks. Early works like [38, 47] showed that structure from motion under affine camera models is essentially a subspace fitting problem. Further, shape deformations are modeled using a low dimensional subspace in [9]. The generalization to multiple independently moving objects leads to segmentation into multiple subspaces [8, 11, 17]. Approaches for handling degenerate and articulated cases were given in [45, 57, 58]. The general problem can be seen as an instance of subspace clustering which assumes that high dimensional data lies in a union of low-dimensional subspaces. The corresponding group of methods aim to simultaneously cluster the data points and estimate the underlying subspaces [1, 26], and may also perform matrix completion [14, 27, 50]. The generalized PCA approach (GPCA) [53–55] treats the union-of-subspaces as a zero-level-set of a higher degree polynomial which is fit to the data. A popular approach to the problem is sparse subspace clustering [5, 13], which has been shown to perform well on the benchmark provided in [49]. These methods are based on the notion of self-expressiveness, i.e. all data in a d-dimensional subspace are linear combinations of d points from that subspace. This enables an efficient approach that is provably correct in the case of disjoint subspaces (see Section 2). The LRR approach [29, 30] uses a dictionary basis and enforces low rank patterns among the coefficients.

More recently, several deep learning methods have been proposed for solving the problem of unsupervised subspace clustering. The first work by Ji *et al.* [23] used deep auto-encoder and introduced a self-expressive layer. Later, a closed form solution of the self-expressive layer was derived [43] which improved its efficiency. Alternatively, it was shown [51] that using over-complete representations increases robustness to noise and improves overall performance. Zhang *et al.* [59] directly learns a self-expressive data representation. The approach is highly scalable since it circumvents the need for a quadratic number of coefficients of the original self-expressive model.

While general subspace learning models typically do not use temporal information, this can be of great importance when analysing and predicting motion. For the reconstruction of deforming objects, Akhter *et al* [2] noted the duality between trajectory space and shape space. The use of pre-defined trajectory-bases such as a DCT basis (motivated by the smoothness assumption of time-dependent trajectories [20, 37]) leads to a reduction in the number of unknowns [2]. Finally, a probabilistic framework for subspace clustering with temporal information is presented in [19].

2. Overview of the Approach

In this section we present our approach and the underlying architecture of our network. Our goal is to learn function f_θ that takes an observed point trajectory as input and yields a (relatively low-dimensional) feature embedding that is appropriate for trajectory grouping. For this purpose we adopt a small PointNet-style network [39]. One of the advantages of this type of network, apart from its simplicity, is the built-in invariance with respect to the permutation of the points in the input tensor. In order to obtain a suitable feature representation, we specify a number of desirable properties that we try to infuse through the architecture and training:

- The features should be adequate for grouping/clustering of trajectories. For this reason we want a latent space with small within-cluster-distances and large between-cluster-distances. This is achieved using an InfoNCE loss.
- Inspired by traditional motion segmentation formulations, we aim for features to encode meaningful "geometric information" about the underlying motion subspaces. For this purpose, we design a decoding module that estimates a basis $\mathtt{B}$ for the underlying motion subspace that the trajectory belongs to. This prevents input memorization and greatly improves generalization to unseen data.
- Despite input tracks being of varying lengths we opt for a feature embedding of a fixed size. This simplifies treatment of problem instances with missing data and trajectories starting and ending at different times. The decoding module therefore takes an extra variable, specifying at which time instance the rows of the basis matrix $\mathtt{B}$ should be sampled. In this way the model allows us to sub-sample and predict motion information beyond the trajectory samples.
- Finally, feature embeddings and intermediate predictions are preferred to behave smoothly with respect to added input noise. To achieve this we construct a "noise-free" version of the observed trajectory from the predicted basis, which we require to be close to the original trajectory through a residual loss. Additionally, this reconstructed trajectory is subsequently fed into the network to obtain a corresponding feature vector, which is also required to be close to the original feature embedding.

Because of the properties above we obtain a trajectory-feature from which we can extract the full motion subspace. Thus we essentially learn a function that maps a single trajectory to a subspace basis. For a general collection of subspaces, where trajectories can belong to multiple of them, this may not be possible (using a purely geometric approach). A simple example is line fitting to image points, where points in the intersection of two models can belong to any of them. In such cases geometry alone is not enough to resolve ambiguous assignments but other cues such as spatial smoothness of assignments [21] have to be used. However, in high-dimensional settings with low-dimensional subspaces the situation is somewhat different. In these cases a typical assumption is that of disjoint subspaces [13], that is, two subspaces only intersect at the origin, ensuring that there is a function that takes (non-zero) trajectories to their corresponding subspaces. We remark that while this assumption is sufficient for our purposes, it is not necessary. Since our method learns a data-driven mapping from point tracks to subspaces, it is able to focus on other trajectory properties than geometric ones. Therefore it can potentially handle the case of intersecting subspaces as long as the trajectory assignment is unambiguous.

The disjoint subspace assumption We conclude this section by illustrating that the disjoint subspace assumption is valid in practical settings by analysing the data in the Hopkins 155 dataset. Figure 2 shows that re-using subspaces to explain trajectories in the other clusters generally leads to higher errors.

Formally, two linear subspaces $\mathcal{S}_i$ and $\mathcal{S}_j$ are disjoint if $\dim(\mathcal{S}_i \oplus \mathcal{S}_j) = \dim(\mathcal{S}_i) + \dim(\mathcal{S}_j)$. Hence if $\mathtt{B}_i$ and $\mathtt{B}_j$ contain bases for the two subspaces, the rank of the concatenated matrix, $\eta_{ij} = \text{rank}((\mathtt{B}_i\ \mathtt{B}_j))$, fulfills $\eta_{ij} < \text{rank}(\mathtt{B}_i) + \text{rank}(\mathtt{B}_j)$ if the subspaces intersect. We extracted subsequences of varying length from the Hopkins155 dataset and used the ground-truth clusters to generate the corresponding subspaces — for all pairs $\{\mathtt{B}_i, \mathtt{B}_j\}$, where $i \neq j$ and $\text{shape}(\mathtt{B}_i) = \text{shape}(\mathtt{B}_j)$, the rank η_{ij} of the concatenated

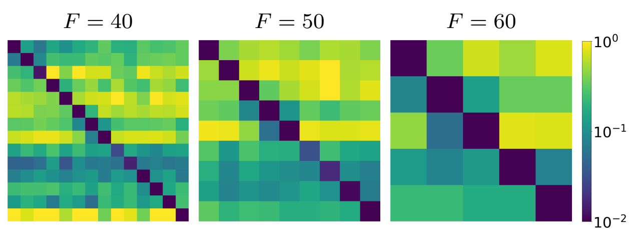


Figure 2. RMS normalized error of the trajectory clusters with respect to the subspaces in Hopkins155 dataset (subsequences of length $F = 40$, 50, and 60). An (i, j) cell corresponds to the trajectory cluster of the ith motion and the subspace of jth motion.

matrix was found to be 8[2]. This suggests that up to numerics the subspaces do not intersect. In the following Section we describe the specific architecture and the employed losses in our approach in more detail.

3. Architecture and Training Losses

The schematic overview of the proposed architecture and training losses is shown in Fig. 1. Using the hypothesis that most observed trajectories come from a unique motion model, we aim to learn a function f_θ mapping a point trajectory directly to a representation of the corresponding motion model. We interpret the output $f_\theta(\mathbf{x})$ as an abstract description of the generating motion, from which a concrete representation to measure geometric consistency can be extracted. We choose linear (low-dimensional) subspaces as the concrete realizations of motion models. Specifically, we learn mappings f_θ and g_ϕ using training data such that

1. f_θ maps a trajectory of varying length to a fixed-length feature embedding, and
2. g_ϕ converts this feature vector for a given query time into the corresponding rows of the subspace representation.

3.1. Feature Extraction

To construct a discriminative function $f_\theta(\cdot)$, that is able to infer the motion from an input point trajectory, we first leverage the ability of the neural networks to compress high-dimensional complex data space into a lower-dimensional latent space with task specific invariance properties, from which it is easier to infer the output. A feature extractor for the point trajectories $f_\theta : \mathbf{x} \mapsto \mathbf{f} \in \mathbb{R}^d$ is therefore trained to produce invariant feature representations using ground truth clusters. In particular, an invariance with respect to the measurement noise and the variations within the motion clusters is of our interest. We therefore use a contrastive loss, namely a variant of the InfoNCE loss [36], to conduct metric learning, leading to a spread of feature embeddings in order to improve feature-based motion clustering,

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{|\mathcal{D}|} \sum_{(i,j,l,k)\in\mathcal{D}} \log\left(\frac{p_{ij}}{p_{ij} + p_{lk}}\right),$$

$$p_{ij} = \exp\left(-\frac{\|\mathbf{f}_i - \mathbf{f}_j\|_2^2}{T}\right), \tag{3}$$

where $\mathbf{f}_i = f_\theta(\mathbf{x}_i)$ and $\mathcal{D}$ is the set of quadruples (i, j, l, k) where (i, k) belong to the same cluster and (l, k) are random pairs, and T is the temperature (hyper-)parameter, which we fix to 1. $\mathcal{L}_{\text{InfoNCE}}$ is an instance of a supervised contrastive loss and is tightly connected to the triplet loss used in metric learning [24].

As mentioned earlier, the proposed architecture resembles PointNet [39] with several modifications. The employed encoder consists of 1D temporal convolutional layers applied to each trajectory independently, with x and y values representing two input channels. A global max-pooling in a temporal domain is then applied on the time-dependent representations, followed by a small MLP and an L_2 normalization. The features are therefore lying on the $d-1$-dimensional unit sphere. Note that we do not apply pooling across the points (global context layer) due to the disjoint subspace assumption. For the experiments, we set the dimensionality of the latent space to $d = 128$.

3.2. Subspace Estimation

The latent vector $f_\theta(\mathbf{x})$ ideally identifies a motion model, but for geometric consistency this feature embedding requires conversion into a suitable mathematical motion model. We assume that observed point trajectories are generated by low-rank subspace models, and therefore the feature embedding is mapped by a subsequent network g_ϕ to the corresponding matrix representation of the associated subspace. Specifically, the mapping $g_\phi : \mathcal{F} \times \mathbb{R} \to \mathbb{R}^{2\times r}$ produces rows $\mathtt{B}_t$ for the given motion model $\mathbf{f} = f_\theta(\mathbf{x})$ and the time index t. Ideally the input trajectory $\mathbf{x}$ lies in the predicted subspace; if the trajectory $\mathbf{x}$ is consisting of (2D) points observed at times $t_1, \dots, t_F$, then the output subspace is

$$\mathtt{B} = \begin{pmatrix} g_\phi(f_\theta(\mathbf{x}), t_1) \\ \vdots \\ g_\phi(f_\theta(\mathbf{x}), t_F) \end{pmatrix}, \tag{4}$$

and the (squared) residual is given in closed-form as

$$\min_{\mathbf{c}} \|\mathbf{x} - \mathtt{B}\mathbf{c}\|_2^2 = \|\mathbf{x} - \mathtt{B}\mathtt{B}^\dagger\mathbf{x}\|_2^2, \tag{5}$$

where $\mathbf{c}$ is the coefficient vector and $\mathtt{B}^\dagger = (\mathtt{B}^\top\mathtt{B})^{-1}\mathtt{B}^\top$. If the network identified the subspace correctly, then the residual is in the order of the observation noise level (and therefore small).

[2]The matrix rank is computed as the number of singular values greater than $\delta\sigma_1$, where σ_1 is the largest singular value, $\delta = 2F\varepsilon$, and $\varepsilon \approx 10^{-7}$ is the numerical zero for `torch.float32`.

Subspace basis functions The network g_ϕ can be a small but generic MLP. Alternatively, we may leverage existing domain knowledge and choose the structure of g_ϕ accordingly. In particular we assume that all motions are linear combination of a finite, but continuous time trajectory basis $\{h^j\}$. The basis functions h^j can be fully fixed (such as a cosine basis with pre-determined frequencies and phase shifts [2]), parametric (such as the cosine basis with trainable frequencies and phase shifts [44]) or entirely non-parametric (i.e. represented by a generic MLP). Our choice of basis functions is a damped version of a cosine basis and is given as follows,

$$h^j_\psi(t) = e^{-(\alpha_j(t-\mu_j))^2}\cos(\beta_j t + \gamma_j), \tag{6}$$

where $\psi = (\mu_j, \alpha_j, \beta_j, \gamma_j)_j$ are trainable parameters. The mapping g_ϕ can therefore be written as a composition

$$g_\phi(\mathbf{f}, t) = \omega_\zeta(\mathbf{f})\mathbf{h}_\psi(t), \tag{7}$$

where $\mathbf{h}_\psi(t) = (h^j_\psi(t))_j$, and the trainable function ω_ζ extracts the basis coefficients from $\mathbf{f}$. If we have N basis functions (i.e. $\dim(\mathbf{h}_\psi(t)) = N$), then $\omega_\zeta(\mathbf{f})$ is a $2r \times N$ matrix. The total set of trainable parameters in the subspace estimation network is $\phi = (\zeta, \psi)$. Finally, the network prediction can be compactly written as

$$\mathtt{B} = \mathtt{H}_\psi(\mathbf{t})\Omega_\zeta(f_\theta(\mathbf{x})), \tag{8}$$

where $\Omega_\zeta(\mathbf{f}) = \text{reshape}_{2N\times r}(\omega_\zeta(\mathbf{f}))$ and

$$\mathtt{H}_\psi(\mathbf{t}) = \begin{pmatrix} \mathbf{h}_\psi(t_1)^\top & \\ & \mathbf{h}_\psi(t_1)^\top \\ & \vdots \\ \mathbf{h}_\psi(t_F)^\top & \\ & \mathbf{h}_\psi(t_F)^\top \end{pmatrix} \in \mathbb{R}^{2F\times 2N}. \tag{9}$$

We set the number of subspace basis functions to $N = 64$ in our experiments. Since the predicted subspace matrix $\mathtt{B}$ augments the feature representation (see Sec. 3.3 below), we enforce a unique subspace representation by fixing the top block of $\mathtt{B}$ to the identity matrix (by multiplying with the respective inverse).

Time dependence and time invariance The network represented by g_ϕ can be interpreted as taking a "feature" token $\mathbf{f}$ and a time coordinate t as input, and is therefore somewhat analogous to the coordinate-MLP employed for implicit neural primitives such as [35, 44] or time series modeling [16]. The continuous time basis $\{h^j\}$ can be seen as domain-specific equivalent to positional coding, but its design is based on the expected motion patterns and does not necessarily meet the requirements of generic positional coding [40, 44, 60].

The formulation above is intentionally not time-invariant: the same trajectory observed at different starting times may predict different motion models. This is usually a desired behavior for motion segmentation, e.g. two cars traveling the same path at different time form different motion segments. As with CNNs applied to images, low-level trajectory features are extracted by convolutional layers, and time-dependence is incorporated via final fully connected and pooling layers.

3.3. Training

The total training loss is the weighted sum of a contrastive loss $\mathcal{L}_{\text{InfoNCE}}$ applied on $\{\mathbf{v}_i\}$, where $\mathbf{v}_i = (\text{flatten}(\mathtt{B}_i)^\top\ \mathbf{f}_i^\top)^\top$, the reconstruction loss $\mathcal{L}_{\text{Residual}}$

$$\mathcal{L}_{\text{Residual}} = \tfrac{1}{P}\sum_{j=1}^{P}\|\mathbf{x}_j - \tilde{\mathbf{x}}_j\|_2^2, \tag{10}$$

and the feature difference loss

$$\mathcal{L}_{\text{FeatDiff}} = \tfrac{1}{P}\sum_{j=1}^{P}\|f_\theta(\mathbf{x}_j) - f_\theta(\tilde{\mathbf{x}}_j)\|_2^2, \tag{11}$$

where $\tilde{\mathbf{x}}_j = \mathtt{B}(\mathbf{x}_j)\mathtt{B}(\mathbf{x}_j)^\dagger\mathbf{x}_j$ is the projection of $\mathbf{x}_j$ onto $\mathtt{B}(\mathbf{x}_j)$. In short, $\mathcal{L}_{\text{InfoNCE}}$ forces both an approximate invariance of the feature extractor f_θ with respect to the variations within clusters and smoothness of the coordinate-MLP g_ϕ, $\mathcal{L}_{\text{Residual}}$ ensures geometric consistency of the subspaces with respect to input trajectories, and $\mathcal{L}_{\text{FeatDiff}}$ forces an approximate invariance of f_θ with respect to the pixel noise and can also be seen as enforcing smoothness of f_θ. We train the network in two stages. First, we pre-train f_θ with $\mathcal{L}_{\text{InfoNCE}}$ until convergence. Then we add g_ϕ and train it together with f_θ using the total loss. To enforce the approximate invariance with respect to random detection failures, we synthesize missing points during training by applying dropout to the input sequences.

3.4. Trajectory Completion

Let $B_{\theta,\phi}(\mathbf{x}, \mathbf{t})$ provide the subspace for an input trajectory $\mathbf{x} \in \mathbb{R}^{2F}$ and time vector $\mathbf{t} = (t_1, ..., t_F)^\top$. Assume now that $\mathbf{x}$ is only partially observed, and let $\mathbf{w} \in \{0,1\}^{2F}$ be the visibility mask. Further, let $\bar{\mathbf{w}} := \mathbf{1} - \mathbf{w}$. The goal of the trajectory completion is to estimate missing values $\bar{\mathbf{x}}$ that minimize the following objective

$$J(\bar{\mathbf{x}}) := \left\|(\mathtt{I} - \mathtt{B}(\bar{\mathbf{x}})\mathtt{B}(\bar{\mathbf{x}})^\dagger)(\mathbf{w}\odot\mathbf{x} + \bar{\mathbf{w}}\odot\bar{\mathbf{x}})\right\|^2, \tag{12}$$

where $\mathtt{B}(\bar{\mathbf{x}}) = B_{\theta,\phi}(\mathbf{w}\odot\mathbf{x} + \bar{\mathbf{w}}\odot\bar{\mathbf{x}}, \mathbf{t})$. Back-propagation for gradient-based determination of $\bar{\mathbf{x}}$ is in principle possible but computationally expensive. Alternatively, we rely on the approximate invariance of the feature extractor f_θ with respect to the length of the input sequence and extrapolate the output subspace in time using $g_\phi(\mathbf{f}, t)$. More

	Hopkins155									Hopkins12		KT3DMoSeg	
	2 motions			3 motions			All						
Method	Mean	Median	Time	Mean	Median	Time	Mean	Median	Time	Mean	Median	Mean	Median
RANSAC [48]	5.56	1.18	175ms	22.94	22.03	258ms	9.76	3.21	194ms	-	-	-	-
GPCA [54]	4.59	0.38	324ms	28.66	28.26	738ms	10.34	2.54	417ms	-	-	34.60	33.95
MSL [45]	4.14	0.00	11h 4m	8.23	1.76	1d 23h	5.03	0.00	19h 11m	-	-	-	-
LSA [57]	3.45	0.59	7.58s	9.73	2.33	15.96s	4.94	0.90	9.47s	-	-	38.30	38.58
ALC_5 [42]	3.03	0.00	-	6.26	1.02	-	3.76	0.26	5m 15s	3.81	0.17	24.31	19.04
ALC_{sp} [42]	2.40	0.43	-	6.69	0.67	-	3.37	0.49	6m 11s	1.28	1.07	-	-
LRR [30]	4.10	0.22	-	9.89	0.56	-	5.41	0.53	1.1s	-	-	33.67	36.01
SSC [13]	0.82	0.00	-	2.45	0.20	-	2.45	0.20	920ms	-	-	33.88	33.54
RSIM [22]	0.78	0.00	-	1.77	0.28	-	1.01	0.00	176ms	0.68	0.70	-	-
MultiConsensus [7]	-	-	-	-	-	-	4.40	-	40ms	-	-	-	-
Ours	0.63	0.0	7ms	0.60	0.0	10ms	0.62	0.0	9ms	5.12	2.04	5.85	0.80

Table 1. Motion segmentation results on the Hopkins155 [49], Hopkins12 [41] and KT3DMoSeg [56] datasets. The evaluation metrics is classification error (%). '-' indicates that the results were not reported / no code available.

specifically, the subspace of $\mathbf{w} \odot \mathbf{x} + \bar{\mathbf{w}} \odot \bar{\mathbf{x}}$ can be estimated from $\mathbf{x}_{\text{vis}}$—a fully visible sub-trajectory of the trajectory $\mathbf{x}$—as long as the feature representation $\mathbf{f}$ can be inferred (i.e. $\mathbf{x}_{\text{vis}}$ is sufficiently long). If we fix this estimate $\mathtt{B}$ and let $\mathtt{W} := \operatorname{diag}(\mathbf{w})$, $\bar{\mathtt{W}} := \operatorname{diag}(\bar{\mathbf{w}})$, the new objective can be stated as

$$\tilde{J}(\bar{\mathbf{x}}) := \left\| \left(\mathtt{I} - \mathtt{B}\mathtt{B}^\dagger\right) \mathtt{W}\mathbf{x} + \left(\mathtt{I} - \mathtt{B}\mathtt{B}^\dagger\right) \bar{\mathtt{W}}\bar{\mathbf{x}} \right\|^2, \qquad (13)$$

and the minimizer $\bar{\mathbf{x}}^*$ of $\tilde{J}$ is given by

$$\bar{\mathbf{x}}^* = -\left(\left(\mathtt{I} - \mathtt{B}\mathtt{B}^\dagger\right)\bar{\mathtt{W}}\right)^\dagger \left(\mathtt{I} - \mathtt{B}\mathtt{B}^\dagger\right)\mathtt{W}\mathbf{x} =: \mathtt{A}(\mathtt{B})\mathbf{x}. \qquad (14)$$

It is now possible to refine the estimated subspace using $\bar{\mathbf{x}}^*$ by feeding $\mathbf{w} \odot \mathbf{x} + \bar{\mathbf{w}} \odot \bar{\mathbf{x}}^*$ into the network. Overall, we apply the following iterative procedure for trajectory completion,

$$\begin{cases} \mathtt{B}_0 \leftarrow B_{\theta,\phi}(\mathbf{x}_{\text{vis}}, \mathbf{t}) \\ \bar{\mathbf{x}}_{i+1} \leftarrow \mathtt{A}(\mathtt{B}_i)\mathbf{x} \\ \mathtt{B}_{i+1} \leftarrow B_{\theta,\phi}(\mathbf{w} \odot \mathbf{x} + \bar{\mathbf{w}} \odot \bar{\mathbf{x}}_{i+1}, \mathbf{t}). \end{cases} \qquad (15)$$

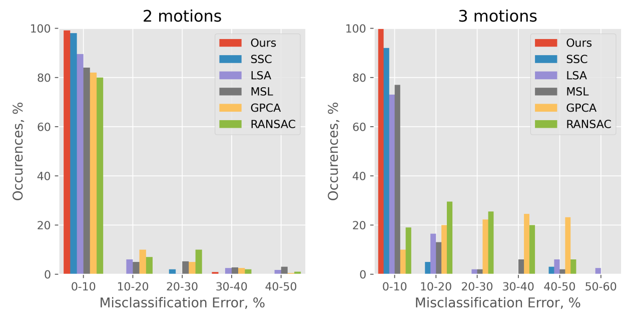


Figure 3. Histograms of the classification errors for the groups with (left) 2 and (right) 3 motions in Hopkins155 dataset.

4. Experiments

The proposed network was trained on the two motion segmentation datasets: Hopkins155 [49] and KT3DMoSeg [56], yielding two different sets of weights. In our experiments, we used NVIDIA GeForce RTX 3080, CUDA 11.5, PyTorch 1.11.

To obtain groups of trajectories, it is sufficient to perform a partial forward pass through the (trained) feature extractor f_θ and cluster in the feature space. We use hierarchical clustering as it provides a meaningful framework for selecting the number of clusters, i.e. motions. The subspaces can then be fit to each cluster. If the goal is to estimate the subspace for a single input trajectory, a full forward pass through f_θ and g_ϕ should be performed. For motion segmentation with missing data, the largest fully visible blocks of trajectories (but ignoring single occluded points) can be used to estimate the initial subspaces $\mathtt{B}$, followed by trajectory completion as outlined in (15). The completed trajectories can then be used to cluster in the feature space.

4.1. Motion Segmentation

We evaluate the performance of the proposed network in terms of motion segmentation on Hopkins155 [49], Hopkins12 [41], and KT3DMoSeg [56] datasets. We first compare the proposed method to the state-of-the-art algorithms that similarly assume rank-4 of $\mathtt{B}$: sequential RANSAC [48], GPCA [54], MSL [45], LSA [57] (with the global subspace dimensionality $D = 4n$ for n clusters), ALC_5 and ALC_{sp} [42], LRR [30], SSC [13], RSIM [22], and MultiConsensus [7]. We then also compare our method to the other approaches making different assumptions on the underlying motion model [3, 4, 28, 56]. The metrics used to evaluate clustering performance is the classification error (%) which is standard for this benchmark [49].

Results on Hopkins155 dataset Hopkins155 [49] is the commonly used benchmark for trajectory-based motion

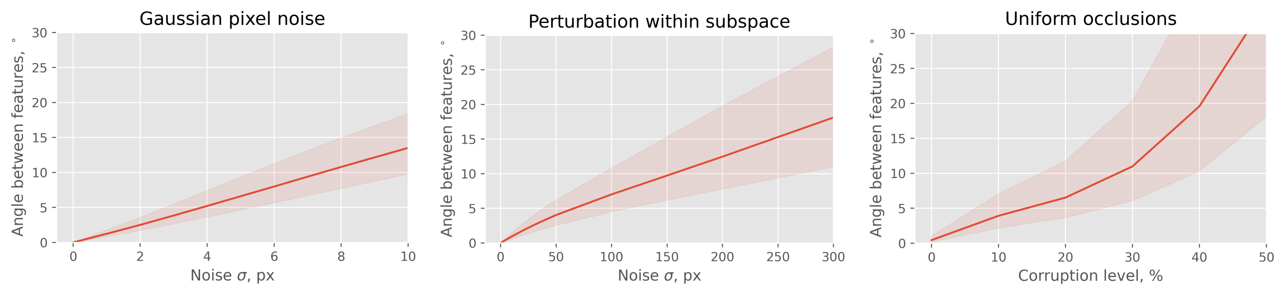


Figure 4. The approximate invariance measured as an angle between features extracted from the reference trajectory and its perturbed version using (left) Gaussian pixel noise, (middle) perturbation within subspace, and (right) uniform occlusions. The gradually increasing level of corruption is shown on the x-axis.

segmentation. The dataset consists of 155 sequences with 63-556 point trajectories generated by two or three motions in the scene. Each trajectory is fully visible, and the length varies from 15 to 100 frames. The averaged results are shown in the left block of Table 1, and the histograms of classification errors are shown in Fig. 3. Our network is very fast and produces the most accurate segmentations on fully visible trajectories.

Results on Hopkins12 dataset Hopkins12 [41] is a commonly used dataset for evaluating subspace clustering with missing data. It consists of 12 corrupted sequences from Hopkins155 dataset, where the corruption was done manually by labeling entries of trajectories as missing. The corruption rate of the observation matrix $\mathtt{M}$ varies from 1% to 23%. For ALC_5 and ALC_{sp} [42], we report l^1-based entry completion results. The results are shown in the middle block of Table 1. The proposed network shows promising results. Note that the maximum corruption rate of the individual trajectories in the Hopkins12 dataset is 93% and, unlike all the state-of-the-art approaches, the proposed trajectory completion method is not global, i.e. it works with individual trajectories. Therefore, if a provided individual sub-trajectory does not contain sufficient relevant information for motion estimation (which becomes more likely as the trajectory gets shorter), incorporating information from all tracks becomes necessary to disambiguate the true motion. Utilizing such global trajectory information is left for future research.

Results on KT3DMoSeg dataset KT3DMoSeg [56] is an adapted version of the KITTI dataset [18, 34] containing 22 sequences with 158-1018 point trajectories and annotated segmentations. It is more challenging than Hopkins155 in several aspects. There is a dominant motion corresponding to static "background" scene which makes cluster distribution unbalanced. The trajectories are also highly occluded: the corruption rate of $\mathtt{M}$ goes up to 54%, and for the individual trajectories — up to 75%. Lastly, the sequences have shorter length of 10 to 20 frames. The results for this dataset are shown in the right block of Table 1 indicating that the proposed approach outperforms previous methods.

Methods with different assumptions We also compared our approach to the other methods making different assumptions. Two-perspective-view (TPV) [28] assumes perspective cameras and uses pairwise epipolar relations. Spectral clustering methods proposed by [56]: kernel addition (KerAdd), co-regularization (CoReg), and subset constrained (Subset) spectral clustering also look at pairwise relations, but use fundamental matrices, affine matrices or homographies to compute the pairwise affinities. SYNCH [3] and MODE [4] similarly fit multiple fundamental matrices to each pair of images, but perform averaging. They assume that only pairwise matches are given. These methods are not directly comparable to ours, however the proposed method proves to be competitive as evidenced in Table 2.

	Hopkins155	Hopkins12		KT3DMoSeg	
Method	Mean	Mean	Med.	Mean	Med.
TPV [28]	2.34	-	-	-	-
KerAdd [56]	0.36	0.11	0.0	8.31	1.02
CoReg [56]	0.46	0.06	0.0	7.92	0.75
Subset [56]	0.31	0.06	0.0	8.08	0.71
SYNCH-tracks [3]	3.67	5.46	0.57	-	-
SYNCH [3]	1.19	3.19	0.28	-	-
MODE [4]	1.37	4.33	0.38	-	-
Our method	0.62	5.12	2.04	5.85	0.80

Table 2. Comparison to the other methods making different assumptions. TPV [28] has no code publicly available.

4.2. Approximate Invariances of f_θ

We experimentally validate the approximate invariance of the trained feature extractor f_θ with respect to different types of synthetic data corruption: (1) Gaussian pixel noise with the standard deviation σ gradually increasing up to 10 px, (2) perturbations within the subspace, where the trajectories are first corrupted by Gaussian pixel noise with σ up to 300 px and then projected back to the underlying subspace, (3) synthesized uniform occlusions with missing data levels going up to 50%, and (4) tracking failures, where

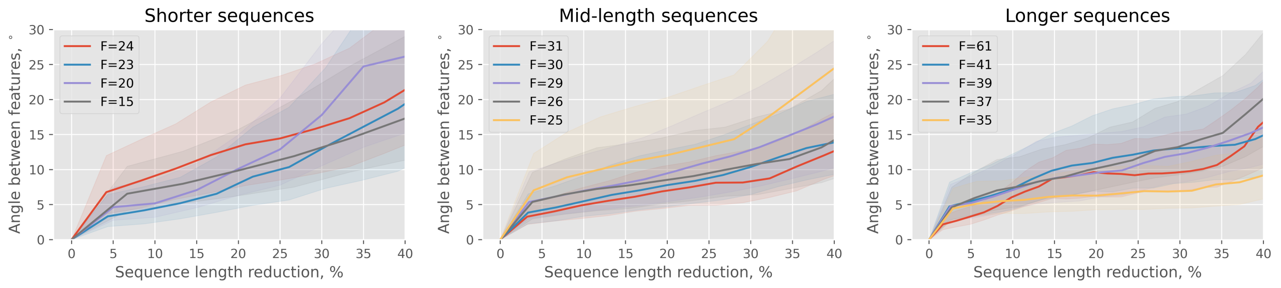


Figure 5. The approximate invariance measured as an angle between features extracted from the reference trajectory and its shortened version for (left) shorter sequences with $F = 15 \dots 24$, (middle) mid-length sequences with $F = 25 \dots 31$, and (right) longer sequences with $F = 35 \dots 61$. The gradually increasing percentage of sequence length reduction is shown on the x-axis.

the rearmost points are gradually excluded from the trajectories. All types of corruption are applied on the training sequences of Hopkins155 dataset. The invariances are measured using the angle between the features extracted from original and corrupted trajectories. The smaller the angle is, the more invariant the feature is to the certain type of data corruption.

Results for the pixel noise, perturbations within the subspace, and uniform occlusions are shown in Fig. 4. Note that the features are L_2-normalized in the d-dimensional space for $d = 128$. In such a high-dimensional space the two randomly picked vectors are likely to be almost orthogonal [52]. The network proves to be approximately invariant hence robust to these types of corruption, and especially to the pixel noise. Since 8-10 px is quite large (for the images of sizes 320×240, 640×480, or 720×480), this result also shows robustness of f_θ to occasional outlying points in the trajectory. The results for the synthesized tracking failures are shown in Fig. 5, where a single curve corresponds to a certain initial trajectory length F. The angles are slightly higher for shorter sequences, which is expected since they contain less information. Note that during training we do not force the network to produce the same feature representations for the trajectories of the same point starting at the same time but having different length.

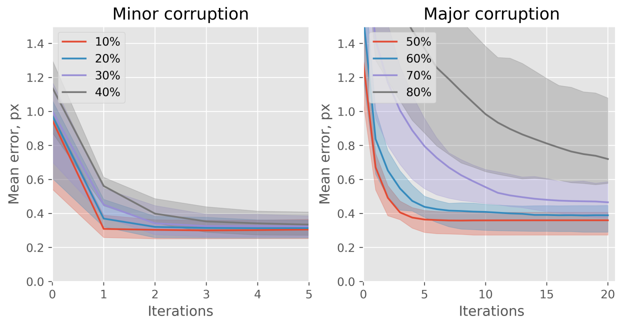


Figure 6. Mean pixel error of recovering from random occlusions for the corruption rates of (left) 10-40% and (right) 50-80%.

4.3. Trajectory Completion

We further evaluate the completion accuracy of the proposed method on the trajectories from Hopkins155 dataset with synthesized uniform occlusions. The performance is measured using the mean pixel error between the ground truth (i.e., detected/tracked) values of the synthetically occluded points and the predicted values. The error as a function of iterations for different corruption rates is shown in Fig. 6, where we split the results into a minor and a major corruption group. From the results we conclude that it is sufficient to run the algorithm (15) for 3 to 5 iterations for minor corruptions which takes 1s on average, and 5 to 20 iterations for major corruptions which takes 3s on average.

5. Conclusion

In this paper we propose to learn a mapping from point trajectories to feature vectors in an embedding space, which fully identify the generating motion. We illustrate that these feature vectors can be directly used for subspace clustering, thereby eliminating the need for simultaneous estimation of assignments and subspace models. The proposed training loss ensures not only the utility of the obtained embedding for clustering, but also forces the feature vector to encode the actual motion parameters, which are retrieved by a dedicated coordinate-MLP. Comparisons to baselines on Hopkins155 and KT3DMoSeg highlight the efficiency and accuracy of our method, providing top clustering results with running time below 10ms.

In this work we focus on the linear subspace framework as sole motion model to explain observed trajectories. One direction for future work is how to incorporate additional and/or different prior assumptions on the motion model into our method. Further, the linear subspace assumption is strongly tied to affine cameras, which only approximate the more realistic pinhole camera model. Therefore extending this work to more general camera models is a challenging direction for future research.

References

[1] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 94–105, 1998. 2

[2] 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*, 33(7):1442–1456, 2010. 3, 5

[3] Federica Arrigoni and Tomas Pajdla. Motion segmentation via synchronization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. 2, 6, 7

[4] Federica Arrigoni and Tomas Pajdla. Robust motion segmentation from pairwise matches. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 671–681, 2019. 2, 6, 7

[5] Liang Bai and Jiye Liang. Sparse subspace clustering with entropy-norm. In *International conference on machine learning*, pages 561–568. PMLR, 2020. 2

[6] Daniel Barath and Jiri Matas. Progressive-x: Efficient, anytime, multi-model fitting algorithm. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3780–3788, 2019. 2

[7] Daniel Barath, Denys Rozumnyi, Ivan Eichhardt, Levente Hajder, and Jiri Matas. Finding geometric models by clustering in the consensus space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5414–5424, 2023. 2, 6

[8] T.E. Boult and L. Gottesfeld Brown. Factorization-based segmentation of motions. In *Proceedings of the IEEE Workshop on Visual Motion*, pages 179–186, 1991. 2

[9] Christoph Bregler, Aaron Hertzmann, and Henning Biermann. Recovering non-rigid 3d shape from image streams. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, pages 690–696. IEEE, 2000. 2

[10] Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *J. ACM*, 58(3), 2011. 2

[11] J. Costeira and T. Kanade. A multi-body factorization method for motion analysis. In *Proceedings of IEEE International Conference on Computer Vision*, pages 1071–1076, 1995. 2

[12] A. Delong, A. Osokin, H. Isack, and Y. Boykov. Fast approximate energy minimization with label costs. *International Journal of Computer Vision*, 96(1):1–27, 2012. 2

[13] Ehsan Elhamifar and René Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE transactions on pattern analysis and machine intelligence*, 35, 2013. 2, 3, 6

[14] Jicong Fan and Madeleine Udell. Online high rank matrix completion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8690–8698, 2019. 2

[15] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 2

[16] Elizabeth Fons, Alejandro Sztrajman, Yousef El-Laham, Alexandros Iosifidis, and Svitlana Vyetrenko. Hypertime: Implicit neural representation for time series. *arXiv preprint arXiv:2208.05836*, 2022. 5

[17] C.W. Gear. Feature grouping in moving objects. In *Proceedings of 1994 IEEE Workshop on Motion of Non-rigid and Articulated Objects*, pages 214–219, 1994. 2

[18] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 7

[19] Behnam Gholami and Vladimir Pavlovic. Probabilistic temporal subspace clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3066–3075, 2017. 3

[20] Paulo FU Gotardo and Aleix M Martinez. Computing smooth time trajectories for camera and deformable shape in structure from motion with occlusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(10):2051–2065, 2011. 3

[21] Hossam Isack and Yuri Boykov. Energy-based geometric multi-model fitting. *International journal of computer vision*, 97(2):123–147, 2012. 2, 3

[22] Pan Ji, Mathieu Salzmann, and Hongdong Li. Shape interaction matrix revisited and robustified: Efficient subspace clustering with corrupted and incomplete data. In *Proceedings of the IEEE International Conference on computer Vision*, pages 4687–4695, 2015. 6

[23] Pan Ji, Tong Zhang, Hongdong Li, Mathieu Salzmann, and Ian Reid. Deep subspace clustering networks. *Advances in neural information processing systems*, 30, 2017. 3

[24] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673, 2020. 4

[25] Florian Kluger, Eric Brachmann, Hanno Ackermann, Carsten Rother, Michael Ying Yang, and Bodo Rosenhahn. Consac: Robust multi-model fitting by conditional sample consensus. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4634–4643, 2020. 2

[26] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *Acm transactions on knowledge discovery from data (tkdd)*, 3(1): 1–58, 2009. 2

[27] Connor Lane, Ron Boger, Chong You, Manolis Tsakiris, Benjamin Haeffele, and Rene Vidal. Classifying and comparing approaches to subspace clustering with missing data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. 2

[28] Zhuwen Li, Jiaming Guo, Loong-Fah Cheong, and Steven Zhiying Zhou. Perspective motion segmentation via

collaborative clustering. In *Proceedings of the IEEE international conference on computer vision*, pages 1369–1376, 2013. 6, 7

[29] Guangcan Liu and Shuicheng Yan. Latent low-rank representation for subspace segmentation and feature extraction. In *2011 International Conference on Computer Vision*, pages 1615–1622, 2011. 3

[30] Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. Robust recovery of subspace structures by low-rank representation. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):171–184, 2012. 3, 6

[31] Luca Magri and Fusiello Andrea. Robust multiple model fitting with preference analysis and low-rank approximation. In *Procedings of the British Machine Vision Conference 2015*, pages 20–1, 2015. 2

[32] Luca Magri and Andrea Fusiello. T-linkage: A continuous relaxation of j-linkage for multi-model fitting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3954–3961, 2014. 2

[33] Luca Magri and Andrea Fusiello. Multiple model fitting as a set coverage problem. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3318–3326, 2016. 2

[34] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3061–3070, 2015. 7

[35] 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. *Communications of the ACM*, 65(1):99–106, 2021. 5

[36] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 4

[37] Hyun Soo Park, Takaaki Shiratori, Iain Matthews, and Yaser Sheikh. 3d reconstruction of a moving point from a series of 2d projections. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part III 11*, pages 158–171. Springer, 2010. 3

[38] C.J. Poelman and T. Kanade. A paraperspective factorization method for shape and motion recovery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):206–218, 1997. 2

[39] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 3, 4

[40] Sameera Ramasinghe and Simon Lucey. Beyond periodicity: Towards a unifying framework for activations in coordinate-mlps. In *European Conference on Computer Vision*, pages 142–158. Springer, 2022. 5

[41] Shankar Rao, Roberto Tron, Rene Vidal, and Yi Ma. Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(10):1832–1845, 2009. 6, 7

[42] Shankar R Rao, Roberto Tron, René Vidal, and Yi Ma. Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008. 6, 7

[43] Junghoon Seo, Jamyoung Koo, and Taegyun Jeon. Deep closed-form subspace clustering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. 3

[44] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020. 5

[45] Yasuyuki Sugaya and Kenichi Kanatani. Geometric structure of degeneracy for multi-body motion segmentation. In *International Workshop on Statistical Methods in Video Processing*, pages 13–25. Springer, 2004. 2, 6

[46] Roberto Toldo and Andrea Fusiello. Robust multiple structures estimation with j-linkage. In *Computer Vision – ECCV 2008*, pages 537–547. Springer Berlin Heidelberg, 2008. 2

[47] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams: a factorization method. *International journal of computer vision*, 9(2):137–154, 1992. 2

[48] Philip HS Torr. Geometric motion segmentation and model selection. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 356(1740):1321–1340, 1998. 2, 6

[49] Roberto Tron and René Vidal. A benchmark for the comparison of 3-d motion segmentation algorithms. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2007. 1, 2, 6

[50] Manolis Tsakiris and Rene Vidal. Theoretical analysis of sparse subspace clustering with missing entries. In *International Conference on Machine Learning*, pages 4975–4984. PMLR, 2018. 2

[51] Jeya Maria Jose Valanarasu and Vishal M Patel. Overcomplete deep subspace clustering networks. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 746–755, 2021. 3

[52] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*. Cambridge university press, 2018. 8

[53] René Vidal and Richard Hartley. Motion segmentation with missing data using powerfactorization and gpca. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, pages II–II. IEEE, 2004. 2

[54] Rene Vidal, Yi Ma, and Shankar Sastry. Generalized principal component analysis (gpca). *IEEE transactions on pattern analysis and machine intelligence*, 27(12):1945–1959, 2005. 6

[55] René Vidal, Roberto Tron, and Richard Hartley. Multiframe motion segmentation with missing data using powerfactorization and gpca. *International Journal of Computer Vision*, 79:85–105, 2008. 2

[56] Xun Xu, Loong Fah Cheong, and Zhuwen Li. Motion segmentation by exploiting complementary geometric models.

In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2859–2867, 2018. 2, 6, 7

[57] Jingyu Yan and Marc Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006, Proceedings, Part IV 9*, pages 94–106. Springer, 2006. 2, 6

[58] Lihi Zelnik-Manor and Michal Irani. Temporal factorization vs. spatial factorization. In *Computer Vision-ECCV 2004: 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part II 8*, pages 434–445. Springer, 2004. 2

[59] Shangzhi Zhang, Chong You, René Vidal, and Chun-Guang Li. Learning a self-expressive network for subspace clustering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12393–12403, 2021. 3

[60] Jianqiao Zheng, Sameera Ramasinghe, and Simon Lucey. Rethinking positional encoding. *arXiv preprint arXiv:2107.02561*, 2021. 5

[61] 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)*, 2014. 1

[62] Marco Zuliani, Charles S Kenney, and BS Manjunath. The multiransac algorithm and its application to detect planar homographies. In *IEEE International Conference on Image Processing 2005*, pages III–153. IEEE, 2005. 2

Learned Trajectory Embedding for Subspace Clustering

Supplementary Material

A. Experimental Details

As discussed in the main paper, the feature extractor f_θ takes a variable-length input trajectory $\mathbf{x}$ with two input channels representing the x and y values and outputs the feature $\mathbf{f} \in \mathbb{R}^d$, and the dimensionality of the latent space is set to $d = 128$. Specifically, the network consists of the three 1D convolutional layers (with channels changing as follows: $2 \rightarrow 64 \rightarrow 128 \rightarrow 512$), followed by max pooling in the temporal domain and two linear layers ($512 \rightarrow 128 \rightarrow 128$). All convolutional operations in f_θ use kernels of size 3 with stride 1. The subspace estimator consists of the parametric basis functions h_ψ^j with trainable parameters $(\mu_j, \alpha_j, \beta_j, \gamma_j)$ and a multilayer perceptron ω_ζ which infers the subspace basis coefficients from the feature $\mathbf{f}$. In particular, ω_ζ has three linear layers ($128 \rightarrow 512 \rightarrow 1024 \rightarrow 512$), and the resulting 512-dimensional vector is reshaped into a coefficient matrix $\Omega_\zeta(\mathbf{f}) \in \mathbb{R}^{128\times4}$ used in (8). ReLU activation is used after each convolutional and each linear layer. During training, we used Adam optimizer with a learning rate set to 0.001, reduced at each epoch with an exponential decay of 0.999.

B. Ablation Studies

We conduct an ablation study in which we first train only the feature extractor f_θ with InfoNCE loss (and obtain network weights θ_1). Subsequently, we include the subspace estimator g_ϕ and continue training with the entire loss, resulting in weights θ_2. The two sets of weights are compared using the classification error of clustering in the feature space. Table 3 shows that the performance on validation and test data improves with the full architecture and the complete loss, proving the advantage of training the feature extractor f_θ together with the subspace estimator g_ϕ.

		Validation subset		Test subset	
Weights	Arch. + loss	Mean	Median	Mean	Median
θ_1	f_θ + InfoNCE loss	1.80	0.00	2.97	0.00
θ_2	f_θ + g_ϕ + total loss	1.51	0.00	0.85	0.21

Table 3. Classification error (%) of clustering with f_{θ_1} trained using f_θ + InfoNCE, and f_{θ_2} trained using f_θ & g_ϕ + total loss.

C. Time Complexity

As discussed in the main paper, our method is very fast. Its time complexity is analysed below. A single trajectory inference requires $\mathcal{O}(F)$ computations due to the convolutional structure. Passing N full trajectories is therefore $\mathcal{O}(NF)$, and the subsequent clustering requires $\mathcal{O}(N^2)$. The trajectory completion comprises matrix operations of size up to $2F \times 2F$ hence costs $\mathcal{O}(F^3)$. It can also be sped up by employing randomized singular value decomposition.