CaSPR: Learning Canonical Spatiotemporal Point Cloud Representations

Davis Rempe[1] **Tolga Birdal**[1] **Yongheng Zhao**[2] **Zan Gojcic**[3]
Srinath Sridhar[4]* **Leonidas J. Guibas**[1]
[1]Stanford University [2]University of Padova [3]ETH Zürich [4]Brown University
geometry.stanford.edu/projects/caspr

Abstract

We propose CaSPR, a method to learn object-centric **Ca**nonical **S**patiotemporal **P**oint Cloud **R**epresentations of dynamically moving or evolving objects. Our goal is to enable information aggregation over time and the interrogation of object state at any spatiotemporal neighborhood in the past, observed or not. Different from previous work, CaSPR learns representations that support spacetime continuity, are robust to variable and irregularly spacetime-sampled point clouds, and generalize to unseen object instances. Our approach divides the problem into two subtasks. First, we explicitly encode time by mapping an input point cloud sequence to a spatiotemporally-canonicalized object space. We then leverage this canonicalization to learn a spatiotemporal latent representation using neural ordinary differential equations and a generative model of dynamically evolving shapes using continuous normalizing flows. We demonstrate the effectiveness of our method on several applications including shape reconstruction, camera pose estimation, continuous spatiotemporal sequence reconstruction, and correspondence estimation from irregularly or intermittently sampled observations.

1 Introduction

The visible geometric properties of objects around us are constantly evolving over time due to object motion, articulation, deformation, or observer movement. Examples include the rigid *motion* of cars on the road, the *deformation* of clothes in the wind, and the *articulation* of moving humans. The ability to capture and reconstruct these spatiotemporally changing geometric object properties is critical in applications like autonomous driving, robotics, and mixed reality. Recent work has made progress on learning object shape representations from static 3D observations [53, 57, 58, 68, 79] and dynamic point clouds [10, 12, 44, 45, 49, 55, 87]. Yet, important limitations remain in terms of the lack of temporal continuity, robustness, and category-level generalization.

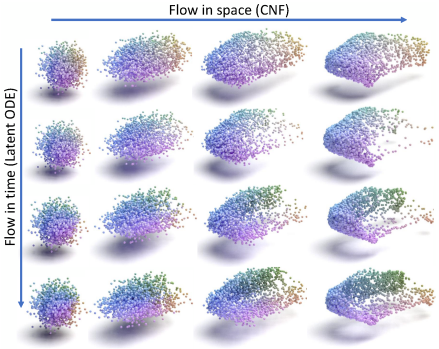

Figure 1: CaSPR builds a point cloud representation of (partially observed) objects continuously in both space (x-axis) and time (y-axis), while canonicalizing for extrinsic object properties like pose.

In this paper, we address the problem of learning object-centric representations that can aggregate and encode **spatiotemporal (ST) changes** in object shape as seen from a 3D sensor. This is challenging since dynamic point clouds captured by depth sensors or LIDAR are often incomplete and sparsely sampled over space and time. Furthermore, even point clouds corresponding to adjacent frames in a sequence will experience large sampling variation. Ideally, we would like spatiotemporal representations to satisfy several desirable properties. First, representations should allow us to

*Work done while at Stanford

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

capture object shape **continuously** over space *and* time. They should encode changes in shape due to varying camera pose or temporal dynamics, and support shape generation at arbitrary spatiotemporal resolutions. Second, representations should be **robust** to irregular sampling patterns in space and time, including support for full or partial point clouds. Finally, representations should support within-category **generalization** to unseen object instances and to unseen temporal dynamics. While many of these properties are individually considered in prior work [12, 32, 45, 49, 74], a unified and rigorous treatment of all these factors in space and time is largely missing.

We address the limitations of previous work by learning a novel object-centric ST representation which satisfies the above properties. To this end, we introduce **CaSPR** – a method to learn **Ca**nonical **S**patiotemporal **P**oint Cloud **R**epresentations. In our approach, we split the task into two: (1) *canonicalizing* an input object point cloud sequence (partial or complete) into a shared 4D container space, and (2) learning a continuous ST latent representation on top of this canonicalized space. For the former, we build upon the Normalized Object Coordinate Space (NOCS) [69, 78] which canonicalizes intra-class 3D shape variation by normalizing for extrinsic properties like position, orientation, and scale. We extend NOCS to a 4D **Temporal-NOCS (T-NOCS)**, which additionally normalizes the duration of the input sequence to a unit interval. Given dynamic point cloud sequences, our ST canonicalization yields spacetime-normalized point clouds. In Sec. 5, we show that this allows learning representations that generalize to novel shapes and dynamics.

We learn ST representations of canonicalized point clouds using *Neural Ordinary Differential Equations* (Neural ODEs) [10]. Different from previous work, we use a **Latent ODE** that operates in a lower-dimensional *learned latent space* which increases efficiency while still capturing object shape dynamics. Given an input sequence, the canonicalization network and Latent ODE together extract features that constitute an ST representation. To continuously *generate* novel spatiotemporal point clouds conditioned on an input sequence, we further leverage invertible *Continuous Normalizing Flows* (CNFs) [7, 25] which transform Gaussian noise directly to the visible part of an object's shape at a desired timestep. Besides continuity, CNFs provide direct likelihood evaluation which we use as a training loss. Together, as shown in Fig. 1, the Latent ODE and CNF constitute a generative model that is continuous in spacetime and robust to sparse and varied inputs. Unlike previous work [12, 45], our approach is continuous and explicitly avoids treating time as another spatial dimension by respecting its unique aspects (*e.g.*, unidirectionality).

We demonstrate that CaSPR is useful in numerous applications including (1) continuous spacetime shape reconstruction from sparse, partial, or temporally non-uniform input point cloud sequences, (2) spatiotemporal 6D pose estimation, and (3) information propagation via space-time correspondences under rigid or non-rigid transformations. Our experiments show improvements to previous work while also providing insights on the emergence of intra-class shape correspondence and the learning of *time unidirectionality* [20]. In summary, our contributions are:

1. The CaSPR encoder network that consumes dynamic object point cloud sequences and canonicalizes them to normalized spacetime (T-NOCS).
2. The CaSPR representation of canonicalized point clouds using a Latent ODE to explicitly encode temporal dynamics, and an associated CNF for generating shapes continuously in spacetime.
3. A diverse set of applications of this technique, including partial or full shape reconstruction, spatiotemporal sequence recovery, camera pose estimation, and correspondence estimation.

2 Related Work

Neural Representations of Point Sets Advances in 2D deep architectures leapt into the realm of point clouds with PointNet [57]. The lack of locality in PointNet was later addressed by a diverse set of works [17, 18, 42, 66, 70, 73, 80, 84, 91], including PointNet++ [58] – a permutation invariant architecture capable of learning both local and global point features. We refer the reader to Guo *et al*. [29] for a thorough review. Treating time as the fourth dimension, our method heavily leverages propositions from these works. Continuous reconstruction of an object's spatial geometry has been explored by recent works in learning implicit shape representations [11, 31, 47, 53].

Spatiotemporal Networks for 3D Data Analogous to volumetric 3D convolutions on video frames [39, 75, 90], a direct way to process spatiotemporal point cloud data is performing 4D convolutions on a voxel representation. This poses three challenges: (1) storing 4D volumes densely is inefficient and impractical, (2) direct correlation of spatial and temporal distances is undesirable,

and (3) the inability to account for timestamps can hinder the final performance. These challenges have fostered further research along multiple fronts. For example, a large body of works [3, 28, 44, 81] has addressed temporal changes between a pair of scans as per-point displacements or *scene flow* [76]. While representing dynamics as fields of change is tempting, such methods lack an explicit notion of time. MeteorNet [45] was an early work to learn flow on raw point cloud sequences, however it requires explicit local ST neighborhoods which is undesirable for accuracy and generalization. Prant *et al*. [55] use temporal frames as a cue of coherence to stabilize the generation of points. CloudLSTM [87] models temporal dependencies implicitly within sequence-to-sequence learning. Making use of time in a more direct fashion, MinkowskiNet [12] proposed an efficient ST 4D CNN to exploit the sparsity of point sets. This method can efficiently perform 4D sparse convolutions, but can neither canonicalize time nor perform ST aggregation. OccupancyFlow [49] used occupancy networks [47] and Neural ODEs [10] to have an explicit notion of time.

Our method can be viewed as learning the underlying *kinematic spacetime surface* of an object motion: an idea from traditional computer vision literature for dynamic geometry registration [48].

Canonicalization Regressing 3D points in a common global reference frame dates back to 6D camera relocalization and is known as *scene coordinates* [67]. In the context of learning the *normalized object coordinate space* (NOCS), [78] is notable for explicitly mapping the input to canonical *object coordinates*. Thanks to this normalization, NOCS enabled category-level pose estimation and has been extended to articulated objects [41], category-level rigid 3D reconstruction [13, 26, 33] via multiview aggregation [69], and non-rigid shape reconstruction either via deep implicit surfaces [86] or by disentangling viewpoint and deformation [50]. Chen *et al*. [8] proposed a latent variational NOCS to *generate* points in a canonical frame.

Normalizing Flows and Neural ODEs The idea of transforming noise into data dates back to whitening transforms [23] and Gaussianization [9]. Tabak and Turner [72] officially defined normalizing flows (NFs) as the composition of simple maps and used it for non-parametric density estimation. NFs were immediately extended to deep networks and high dimensional data by Rippel and Adams [61]. Rezende and Mohamed used NFs in the setting of variational inference [59] and popularized them as a standalone tool for deep generative modeling *e.g*. [35, 71]. Thanks to their invertibility and exact likelihood estimation, NFs are now prevalent and have been explored in the context of graph neural networks [43], generative adversarial networks [27], bypassing topological limitations [2, 15, 19], flows on Riemannian manifolds [24, 46, 65], equivariant flows [5, 37, 60], and connections to optimal transport [21, 51, 77, 89]. The limit case where the sequence of transformations are indexed by real numbers yields continuous-time flows: the celebrated Neural ODEs [7], their latent counterparts [62], and FFJORD [25], an invertible generative model with unbiased density estimation. For a comprehensive review, we refer the reader to the concurrent surveys of [36, 52].

Our algorithm is highly connected to PointFlow [83] and C-Flow [56]. However, we tackle encoding and generating spatiotemporal point sets in addition to canonicalization while both of these works use CNFs in generative modeling of 3D point sets without canonicalizing.

3 Background

In this section, we lay out the notation and mathematical background required in Sec. 4.

Definition 1 (Flow & Trajectory) *Let us define a d-dimensional **flow** to be a parametric family of homeomorphisms $\phi : \mathcal{M} \times \mathbb{R} \mapsto \mathcal{M}$ acting on a vector $\mathbf{z} \in \mathcal{M} \subset \mathbb{R}^d$ with $\phi_0(\mathbf{z}) = \mathbf{z}$ (identity map) and $\phi_t(\mathbf{z}) = \mathbf{z}_t$. A temporal subspace of flows is said to be a **trajectory** $\mathcal{T}(\mathbf{z}) = \{\phi_t(\mathbf{z})\}_t$ if $\mathcal{T}(\mathbf{z}) \cap \mathcal{T}(\mathbf{y}) = \emptyset$ for all $\mathbf{z} \neq \mathbf{y}$, i.e., different trajectories never intersect [14, 19].*

Definition 2 (ODE-Flow, Neural ODE & Latent ODE) *For any given flow ϕ there exists a corresponding ordinary differential equation (**ODE**) constructed by attaching an optionally time-dependent vector $f(\mathbf{z}, t) \in \mathbb{R}^d$ to every point $\mathbf{z} \in \mathcal{M}$ resulting in a vector field s.t. $f(\mathbf{z}) = \phi'(\mathbf{z})|_{t=0}$. Starting from the initial state $\mathbf{z}_0$, this ODE given by $\frac{dz(t)}{dt} = f(z(t), t)$ can be integrated for time T modeling the flow $\phi_{t=T}$:*

$$\mathbf{z}_T = \phi_T(\mathbf{z}_0) = \mathbf{z}_0 + \int_0^T f_\theta(\mathbf{z}_t, t)\, dt, \tag{1}$$

where $\mathbf{z}_t \triangleq z(t)$ and the field f is parameterized by $\boldsymbol{\theta} = \{\theta_i\}_i$. By the Picard–Lindelöf theorem [14], if f is continuously differentiable then the initial value problem in Eq (1) *has a unique solution.*

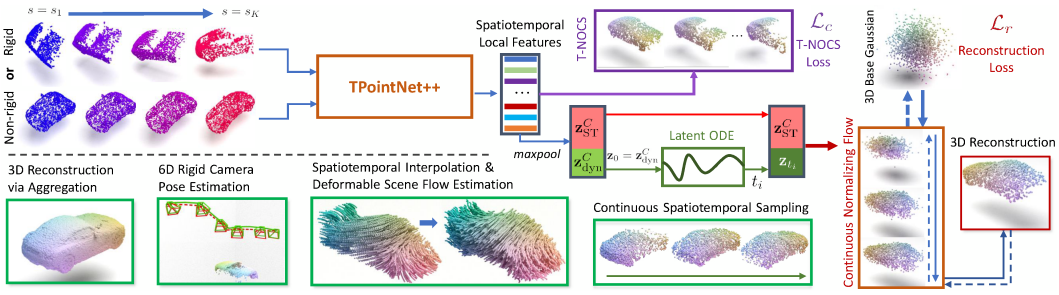

Figure 2: Architecture and applications of CaSPR. Our model consumes rigid or deformable point cloud sequences and maps them to a spatiotemporal canonical latent space whose coordinates are visualized by RGB colors (purple box). Using a Latent ODE, it advects a latent subspace forward in time to model temporal dynamics. A continuous normalizing flow [25] (shown in red) decodes the final latent code to 3D space by mapping Gaussian noise to the partial or full shape at desired timesteps. CaSPR enables multiple applications shown in green boxes. Training directions for the normalizing flow are indicated by dashed arrows.

Instead of handcrafting, ***Neural ODEs*** *[10] seek a function f that suits a given objective by modeling f as a neural network. We refer to a Neural ODE operating in a latent space as a* ***Latent ODE****.*

Numerous forms of Neural ODEs model $f(\cdot)$ to be *autonomous*, *i.e.*, time independent $f(\mathbf{z}_t) \equiv f(\mathbf{z}_t, t)$ [10, 19, 62], whose output fully characterizes the trajectory. While a Neural ODE advects single particles, **generative modeling** approximates the full target probability density which requires expressive models capable of exact density evaluation and sampling that avoids mode collapse.

Definition 3 (Continuous Normalizing Flow (CNF)) *Starting from a simple $d_{\mathcal{B}}$-dimensional base distribution p_y with $\mathbf{y}_0 \in \mathbb{R}^{d_{\mathcal{B}}} \sim p_y(\mathbf{y})$,* ***CNFs*** *[7, 25] aim to approximate the complex target distribution $p_x(\mathbf{x})$ by bijectively mapping empirical samples of the target to the base using an invertible function $g_\beta : \mathbb{R}^{d_{\mathcal{B}}} \mapsto \mathbb{R}^{d_{\mathcal{B}}}$ with parameters $\boldsymbol{\beta} = \{\beta_i\}_i$. Then the probability density function transforms with respect to the change of variables: $\log p_x(\mathbf{x}) = \log p_y(\mathbf{y}) - \log\det \nabla g_\beta(\mathbf{y})$. The warping function g can be replaced by an integral of continuous-time dynamics yielding a form similar to Neural ODEs except that we now consider distributions [25]:*

$$\log p_x(\mathbf{x}) \;=\; \log p_y(\mathbf{y}_0) - \int_0^T \mathrm{Tr}\left(\frac{\partial g_\beta(\mathbf{y}_t, t \,|\, \mathbf{z})}{\partial \mathbf{y}_t}\right) dt, \tag{2}$$

with the simplest choice that the base distribution $\mathbf{y}_0$ is in a d-dimensional ball, $p_y \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Here $\mathbf{z} \in \mathbb{R}^d$ is an optional ***conditioning*** *latent vector [83]. Note that this continuous system is* non-autonomous *i.e., time varying and every non-autonomous system can be converted to an autonomous one by raising the dimension to include time [16, 19].*

4 Method

We consider as input a sequence of potentially partial, clutter-free 3D scans (readily captured by depth sensors or LIDAR) of an object belonging to a known category. This observation is represented as a point cloud $\mathcal{X} = \{\mathbf{x}_i \in \mathbb{R}^3 = \{x_i, y_i, z_i\} \,|\, i = 1, \ldots, M'\}$. For a sequence of K potentially non-uniformly sampled timesteps, we denote a **spatiotemporal** (ST) point cloud as $\mathcal{P} = \{\mathcal{P}_k\}_{k=1}^K$, where $\mathcal{P}_k = \{\mathbf{p}_i \in \mathbb{R}^4 = \{x_i, y_i, z_i, s_k\} \,|\, i = 1, \ldots, M_k\}$, M_k is the number of points at frame $k \in [1, K]$ and at the *time* $s_k \in [s_1, s_K] \subset \mathbb{R}$ with $M = \sum_{k=1}^K M_k$. Our goal is to explain $\mathcal{P}$ by learning a *continuous representation* of shape that is invariant to extrinsic properties while aggregating intrinsic properties along the direction of time. **CaSPR** achieves this through:

1. A **canonical** spacetime container where extrinsic properties such as object pose are factored out,
2. A continuous **latent** representation which can be queried at arbitrary spacetime steps, and
3. A **generative** model capable of reconstructing partial observations conditioned on a latent code.

We first describe the method design for each of these components, depicted in Fig. 2, followed by implementation and architectural details in Sec. 4.1.

Canonicalization: The first step is *canonicalization* of a 4D ST point cloud sequence with the goal of associating observations at different time steps to a common canonical space. Unlike prior work

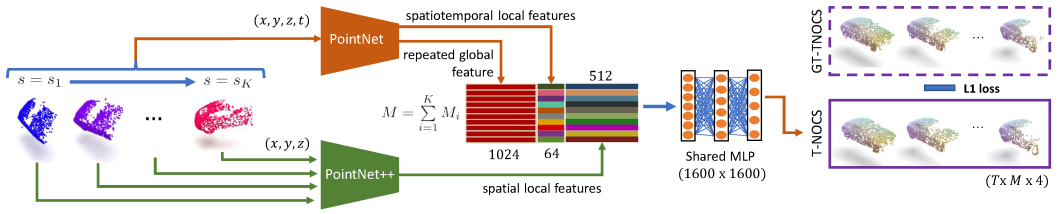

Figure 3: Architecture of our ST point-set canonicalization network, *TPointNet++*. It uses two branches that extract ST features using a 4D PointNet and per-view 3D local features via PointNet++. These features are combined and passed to an MLP to regress the T-NOCS points. Training is supervised via GT coordinates.

which assumes already-canonical inputs [49, 83], this step allows CaSPR to operate on raw point cloud sequences in world space and enables multiple applications (see Fig. 2). Other previous work has considered canonicalization of extrinsic properties from RGB images [69, 78] or a 3D point cloud [41], but our method operates on a 4D point cloud and explicitly accounts for time labels. Our goal is to find an injective *spacetime canonicalizer* $c_\alpha(\cdot) : \mathcal{P} \mapsto \overline{\mathcal{P}} \times \overline{\mathcal{Z}}$ parameterized by $\boldsymbol{\alpha} = \{\alpha_i\}_i$, that maps a point cloud sequence $\mathcal{P}$ to a canonical *unit tesseract* $\overline{\mathcal{P}} = \{\overline{\mathcal{P}}_k\}_{k=1}^K$, where $\overline{\mathcal{P}}_k = \{\overline{\mathbf{p}}_i \in \mathbb{R}^4 = \{\bar{x}_i, \bar{y}_i, \bar{z}_i, \bar{t}_k\} \in [0,1] \,|\, i = 1, \ldots, M_k\}$ and $\mathbf{z}^C \in \overline{\mathcal{Z}} \subset \mathbb{R}^d$ is the corresponding canonical *latent* representation (embedding) of the sequence. Note that in addition to position and orientation, $\overline{\mathcal{P}}$ is normalized to have time in unit duration. We refer to $\overline{\mathcal{P}}$ as **Temporal-NOCS (T-NOCS)** as it extends NOCS [69, 78]. T-NOCS points are visualized using the spatial coordinate as the *RGB* color in Fig. 2 and 3. Given a 4D point cloud in the world frame, we can aggregate the entire shape from K partial views by a simple union: $\overline{\mathcal{P}} = \bigcup_{i=1}^K c_\alpha(\mathcal{P}_i)$ [69]. Moreover, due to its injectivity, $c_\alpha(\cdot)$ preserves *correspondences*, a property useful in tasks like pose estimation or label propagation. We outline the details and challenges involved in designing a canonicalizer in Sec. 4.1.

Continuous Spatiotemporal Representation: While a global ST latent embedding is beneficial for canonicalization and aggregation of partial point clouds, we are interested in continuously modeling the ST input, *i.e.*, being able to compute a representation for unobserved timesteps at arbitrary spacetime resolutions. To achieve this, we split the latent representation: $\mathbf{z}^C = [\mathbf{z}^C_{\text{ST}}, \mathbf{z}^C_{\text{dyn}}]$ where $\mathbf{z}^C_{\text{ST}}$ is the *static* ST descriptor and $\mathbf{z}^C_{\text{dyn}}$ is used to initialize an autonomous Latent ODE $\frac{d\mathbf{z}_t}{dt} = f_\theta(\mathbf{z}_t)$ as described in Dfn. 2: $\mathbf{z}_0 = \mathbf{z}^C_{\text{dyn}} \in \mathbb{R}^d$. We choose to advect the ODE in the latent space (rather than physical space [49]) to (1) enable *learning* a space best-suited to modeling the dynamics of the observed data, and (2) improve scalability due to the fixed feature size. Due to the time-independence of f_θ, $\mathbf{z}_0$ fully characterizes the latent trajectory. Advecting $\mathbf{z}_0$ forward in time by solving this ODE until any canonical timestamp $T \leq 1$ yields a continuous representation in time $\mathbf{z}_T$ that can explain changing object properties. We finally obtain a dynamic spatiotemporal representation in the product space: $\mathbf{z} \in \mathbb{R}^D = [\mathbf{z}^C_{\text{ST}}, \mathbf{z}_T]$. Due to canonicalization to the unit interval, $T > 1$ implies *extrapolation*.

Spatiotemporal Generative Model: Numerous methods exist for point set generation [1, 26, 91], but most are not suited for sampling on the surface of a partial 4D ST point cloud. Therefore, we adapt CNFs [25, 83] as defined in Sec. 3. To generate a novel ST shape, *i.e.*, a sequence of 3D shapes $\mathcal{X}_1 \ldots \mathcal{X}_K$, we simulate the Latent ODE for $t = 0 \ldots T$ and obtain representations for each of the canonical shapes in the sequence: $\mathbf{z}_{t=0} \cdots \mathbf{z}_{t=T}$. We then sample the base distribution $\mathbf{y}_k \in \mathbb{R}^{d_{\mathcal{B}}=3} \sim p_y(\mathbf{y}) \triangleq \mathcal{N}(\mathbf{0}, \mathbf{I})$ and evaluate the conditional CNF in Eq (2) by passing each sample $\mathbf{y}_k$ through the flow $g_\beta(\mathbf{y}_k \,|\, \mathbf{z}_t)$ conditioned on $\mathbf{z}_t$. Note that the flow is time dependent, *i.e.*, non-autonomous. To increase the temporal resolution of the output samples we pick the timesteps with higher frequency, whereas to *densify* spatially, we simply generate more samples $\mathbf{y}_k$.

4.1 Network Architecture

We now detail our implementations of the canonicalizer c_α, Latent ODE network f_θ, and CNF g_β.

TPointNet++ $c_\alpha(\cdot)$: The design of our canonicalizer is influenced by (1) the desire to avoid ST neighborhood queries, (2) to treat time as important as the spatial dimensions, and (3) injecting how an object appears during motion in space into its local descriptors resulting in more expressive features. While it is tempting to directly apply existing point cloud architectures such as PointNet [57] or PointNet++ [58], we found experimentally that they were individually insufficient (*c.f.* Sec. 5). To meet our goals, we instead introduce a hybrid *TPointNet++* architecture as shown in Fig. 3 to implement c_α and canonicalize $\mathcal{P}$ to $\overline{\mathcal{P}}$. TPointNet++ contains a PointNet branch that consumes the

Figure 4: Canonicalization applications. Partial shape reconstruction (left section) shows pairs of GT (left) and predicted shapes (right). Pose estimation (right section) shows GT (green, solid) and predicted (red, dashed) camera pose based on regressed T-NOCS points. Points are colored by their T-NOCS location.

entire 4D point cloud to extract both a 1024-dimensional global feature and 64-dimensional per-point ST features. This treats time explicitly and equally to each spatial dimension. We also use PointNet++ to extract a 512-dimensional local feature at each input point by applying it at each cross-section in time with no timestamp. We feed all features into a shared multi-layer perceptron (MLP) to arrive at 1600-dimensional embeddings corresponding to each input point.

We use the pointwise embeddings in two ways: (1) they are passed through a shared linear layer followed by a *sigmoid* function to estimate the T-NOCS coordinates $\widehat{\mathcal{P}}$ which approximate the ground truth $\overline{\mathcal{P}}$, and (2) we max-pool all per-point features into a single latent representation of T-NOCS $\mathbf{z}^C \in \mathbb{R}^{1600}$ which is used by the Latent ODE and CNF as described below. The full canonicalizer $c_\alpha(\cdot)$ can be trained independently for T-NOCS regression, or jointly with a downstream task.

Latent ODE $f_\theta(\cdot)$ and Reconstruction CNF $g_\beta(\cdot)$: The full CaSPR architecture is depicted in Fig. 2. It builds upon the embedding from TPointNet++ by first splitting it into two parts $\mathbf{z}^C = [\mathbf{z}^C_{\text{ST}}, \mathbf{z}_0 \triangleq \mathbf{z}^C_{\text{dyn}}]$. The dynamics network of the Latent ODE f_θ is an MLP with three hidden layers of size 512. We use a Runge-Kutta 4(5) solver [38, 63] with adaptive step sizes which supports backpropagation using the adjoint method [10]. The static feature, $\mathbf{z}^C_{\text{ST}} \in \mathbb{R}^{1536}$ is skip-connected and concatenated with $\mathbf{z}_T$ to yield $\mathbf{z} \in \mathbb{R}^{1600}$ which conditions the reconstruction at $t = T$.

To sample the surface represented by $\mathbf{z}$, we use a FFJORD conditional-CNF [25, 83] as explained in Sec. 3 and 4 to map 3D Gaussian noise $\mathbf{y}_0 \in \mathbb{R}^{d_{\mathcal{B}}=3} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ onto the shape surface. The dynamics of this flow $g_\beta(\mathbf{y}_t, t \,|\, \mathbf{z})$ are learned with a modified MLP [25] which leverages a gating mechanism at each layer to inject information about the current context including $\mathbf{z}$ and current time t of the flow. This MLP contains three hidden layers of size 512, and we use the same solver as the Latent ODE. Please refer to the supplement for additional architectural details.

Training and Inference: CaSPR is trained with two objectives that use the GT canonical point cloud sequence $\overline{\mathcal{P}}$ as supervision. We primarily seek to maximize the *log-likelihood* of canonical spatial points on the surface of the object when mapped to the base Gaussian using the CNF. This reconstruction loss is $\mathcal{L}_r = -\sum_{k=1}^{K}\sum_{i=1}^{M_k} \log p_x(\bar{\mathbf{x}}_i \,|\, \mathbf{z}_{t_k})$ where $\bar{\mathbf{x}}_i$ is the spatial part of $\bar{\mathbf{p}}_i \in \overline{\mathcal{P}}_k$ and the log-likelihood is computed using Eq (2). Secondly, we supervise the T-NOCS predictions from TPointNet++ with an L1 loss $\mathcal{L}_c = \sum_{i=1}^{M} |\widehat{\mathbf{p}}_i - \bar{\mathbf{p}}_i|$ with $\bar{\mathbf{p}}_i \in \overline{\mathcal{P}}$ and $\widehat{\mathbf{p}}_i \in \widehat{\mathcal{P}}$. We jointly train TPointNet++, the Latent ODE, and CNF for $\boldsymbol{\alpha}$, $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ respectively with the final loss $\mathcal{L} = \mathcal{L}_r + \mathcal{L}_c$. During inference, TPointNet++ processes a raw point cloud sequence of an unseen shape and motion to obtain the ST embedding and canonicalized T-NOCS points. The Latent ODE, initialized by this embedding, is solved forward in time to any number of canonical "query" timestamps. For each timestamp, the Latent ODE produces the feature to condition the CNF which reconstructs the object surface by the forward flow of Gaussian samples. The combined continuity of the Latent ODE and CNF enables CaSPR to reconstruct the input sequence at any desired ST resolution.

5 Experimental Evaluations

We now evaluate the canonicalization, representation, and reconstruction capabilities of CaSPR, demonstrate its utility in multiple downstream tasks, and justify design choices.

Dataset and Preprocessing: We introduce a new dataset containing simulated rigid motion of objects in three ShapeNet [6] categories: cars, chairs, and airplanes. The motion is produced with randomly generated camera trajectories (Fig. 4) and allows us to obtain the necessary inputs and supervision for CaSPR: sequences of raw partial point clouds from depth maps with corresponding canonical T-NOCS point clouds. Each sequence contains $K = 10$ frames with associated timestamps.

Table 2: Partial surface sequence reconstruction. Chamfer (CD) and Earth Mover's Distances (EMD) are multiplied by 10^3. On the left (*10 Observed*), 10 frames are given as input and all are reconstructed. On the right, 3 frames are used as input (*3 Observed*), but methods also reconstruct intermediate unseen steps (*7 Unobserved*).

		10 Observed		*3 Observed*		*7 Unobserved*	
Method	**Category**	**CD**	**EMD**	**CD**	**EMD**	**CD**	**EMD**
PointFlow	Cars	**0.454**	12.838	**0.455**	12.743	**0.525**	13.911
CaSPR-Atlas		0.492	19.528	0.540	22.099	0.530	19.635
CaSPR		0.566	**10.103**	0.590	**11.464**	0.584	**11.259**
PointFlow	Chairs	0.799	17.267	0.796	17.294	0.950	18.442
CaSPR-Atlas		**0.706**	48.665	0.723	48.912	0.749	47.322
CaSPR		0.715	**13.009**	**0.681**	**13.310**	**0.683**	**13.564**
PointFlow	Airplanes	0.251	9.500	0.252	9.534	0.281	9.814
CaSPR-Atlas		0.237	18.827	0.255	18.525	0.269	17.933
CaSPR		**0.231**	**6.026**	**0.215**	**6.144**	**0.216**	**6.175**

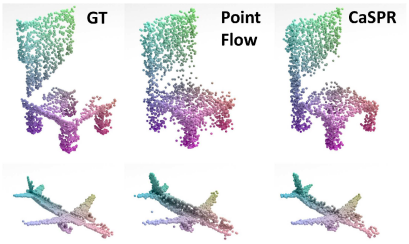

Figure 5: Reconstruction results. CaSPR accurately captures occlusion boundaries for camera motion at observed and unobserved timesteps, unlike linear feature interpolation with PointFlow.

Raw point cloud sequences are labeled with uniform timestamps from $s_1 = 0.0$ to $s_K = 5.0$ while canonicalized timestamps range from $\bar{t}_1 = 0$ to $\bar{t}_K = 1$. For training, 5 frames with 1024 points are randomly subsampled from each sequence, giving non-uniform step sizes between observations. At test time, we use a *different spatiotemporal sampling* for sequences of held-out object instances: all 10 frames, each with 2048 points. Separate CaSPR models are trained for each shape category.

Evaluation Procedure: To measure canonicalization errors, T-NOCS coordinates are split into the spatial and temporal part with GT given by $\bar{\mathcal{X}}$ and $\bar{\mathbf{t}}$ respectively. The *spatial error* at frame k is $\frac{1}{M_k}\sum_{i=1}^{M_k} \|\widehat{\mathbf{x}}_i - \bar{\mathbf{x}}_i\|_2$ and the *temporal error* is $\frac{1}{M_k}\sum_{i=1}^{M_k} |\widehat{t}_i - \bar{t}_i|$. For reconstruction, the Chamfer Distance (CD) and Earth Mover's Distance (EMD) are measured (and reported multiplied by 10^3). Lower is better for all metrics; we report the median over all test frames because outlier shapes cause less informative mean errors. Unless stated otherwise, qualitative point cloud results (*e.g.*, Fig. 4) are colored by their canonical coordinate values (so corresponding points should have the same color).

5.1 Evaluations and Applications

Table 1: Canonicalization performance.

Method	**Category**	**Spatial Err**	**Time Err**
MeteorNet	Cars	0.0633	**0.0001**
PointNet++ No Time		0.0530	—
PointNet++ w/ Time		0.0510	0.0005
PointNet		0.0250	0.0012
TPointNet++ No Time		0.0122	—
TPointNet++	Cars	**0.0118**	0.0011
TPointNet++	Chairs	0.0102	0.0008
TPointNet++	Airplanes	0.0064	0.0009

Canonicalization: We first evaluate the accuracy of canonicalizing raw partial point cloud sequences to T-NOCS using TPointNet++. Tab. 1 shows median errors over all frames in the test set. The bottom part evaluates TPointNet++ on each shape category while the top compares with baselines on cars (please see supplementary for more details). Notably, for spatial prediction, TPointNet++ outperforms variations of both PointNet [57] and PointNet++ [58], along with their spatiotemporal extension MeteorNet [45]. This indicates that our ST design yields more distinctive features both spatially and temporally. MeteorNet and PointNet++ (with time) achieve impressive time errors thanks to skip connections that pass the input timestamps directly towards the end of the network. Qualitative results of canonicalization are in Fig. 4.

Representation and Reconstruction: We evaluate CaSPR's ability to represent and reconstruct observed and unobserved frames of *raw* partial point cloud sequences. The full model is trained on each category separately using both $\mathcal{L}_r$ and $\mathcal{L}_c$, and is compared to two baselines. The first is a variation of CaSPR where the CNF is replaced with an AtlasNet [26] decoder using 64 patches – an alternative approach to achieve spatial continuity. This model is trained with $\mathcal{L}_c$ and a CD loss (rather than $\mathcal{L}_r$). The second baseline is the deterministic PointFlow [83] autoencoder trained to reconstruct a *single canonical* partial point cloud. This model operates on a single timestep and receives the **already canonical** point cloud as input: an easier problem. We achieve temporal continuity with PointFlow by first encoding a pair of adjacent observed point clouds to derive two shape features, and then linearly interpolating to the desired timestamp – one alternative to attain temporal continuity. The interpolated feature conditions PointFlow's CNF to sample the partial surface, similar to CaSPR.

Tab. 2 reports median CD and EMD at reconstructed test steps for each method. We evaluate two cases: (1) models receive and reconstruct all 10 observed frames (left), and (2) models get the first, middle, and last steps of a sequence and reconstruct both these 3 observed and 7 unobserved frames (right). CaSPR outperforms PointFlow in most cases, even at observed timesteps, despite operating on raw point clouds in the world frame instead of canonical. Because PointFlow reconstructs each

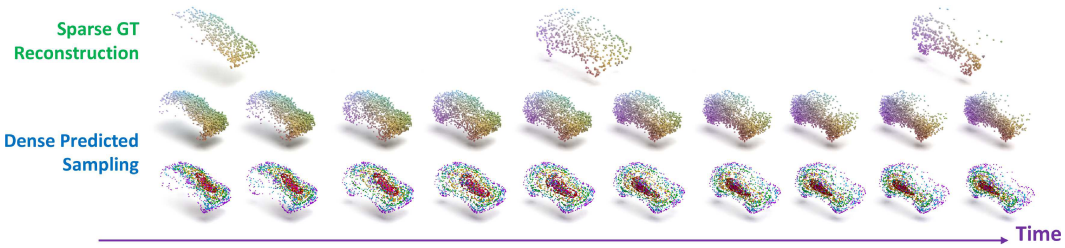

Figure 7: Continuous interpolation results. From three sparse frames of input with GT canonical points shown on top, CaSPR reconstructs the sequence more densely in space and time (middle). Contours of the Gaussian flowed to the car surface are shown on bottom (red is highest probability).

frame independently, it lacks temporal context resulting in degraded occlusion boundaries (Fig. 5) and thus higher EMD. CaSPR gives consistent errors across observed and unobserved frames due to the learned motion prior of the Latent ODE, in contrast to linear feature interpolation that sees a marked performance drop for unobserved frames. The AtlasNet decoder achieves small CD since this is the primary training loss, but has difficulty reconstructing the correct point distribution on the partial surface due to the patch-based approach, resulting in much higher EMD for all cases.

Multiview Reconstruction: A direct application of TPointNet++ is partial shape reconstruction of observed geometry through a union of predicted T-NOCS spatial points. Due to the quantitative accuracy of TPointNet++ at each frame (Tab. 1), aggregated results closely match GT for unseen instances in all categories as shown in Fig. 4 (left).

Table 3: Pose estimation using T-NOCS.

Method	Category	Trans Err	Rot Err(°)	Point Err
RPM-Net	Cars	**0.0049**	**1.1135**	**0.0066**
CaSPR		0.0077	1.3639	0.0096
RPM-Net	Chairs	**0.0026**	**0.4601**	**0.0036**
CaSPR		0.0075	1.5035	0.0091
RPM-Net	Airplanes	**0.0040**	**0.5931**	**0.0048**
CaSPR		0.0051	0.9456	0.0057

Rigid Pose Estimation: The world–canonical 3D point correspondences from TPointNet++ allow fitting rigid object (or camera) pose at observed frames using RANSAC [22]. Tab. 3 reports median test errors showing TPointNet++ is competitive with RPM-Net [85], a recent **specialized** architecture for robust iterative rigid registration. Note here, RPM-Net takes *both* the raw depth and **GT** T-NOCS points as input. Translation and rotation errors are the distance and degree angle difference from the GT transformation. Point error measures the per frame median distance between the GT T-NOCS points transformed by the predicted pose and the input points. Qualitative results are in Fig. 4 (right).

Rigid Spatiotemporal Interpolation: The full CaSPR model can densely sample a sparse input sequence in space-time as shown in Fig. 7. The model takes three input frames of 512 points (corresponding GT T-NOCS points shown on top) and reconstructs an arbitrary number of steps with 2048 points (middle). The representation can be sampled at any ST resolution but, in practice, is limited by memory. The CNF maps Gaussian noise to the visible surface (bottom). Points are most dense in high probability areas (shown in red); in our data this roughly corresponds to where the camera is focused on the object surface at that timestep.

Table 4: Reconstructing 10 observed timesteps (left) and maintaining temporal correspondences (right) on Warping Cars.

	Reconstruction		*Correspondences*	
Method	**CD**	**EMD**	**Dist** t_1	**Dist** t_{10}
OFlow	1.512	20.401	**0.011**	**0.031**
CaSPR	**0.955**	**11.530**	0.013	0.035

Non-Rigid Reconstruction and Temporal Correspondences: CaSPR can represent and reconstruct deformable objects. We evaluate on a variation of the Warping Cars dataset introduced in Occupancy Flow (OFlow) [49] which contains 10-step sequences of *full* point clouds sampled from ShapeNet [6] cars deforming over time. The sequences in this dataset are already consistently aligned and scaled, so CaSPR is trained only using $\mathcal{L}_r$.

Tab. 4 compares CaSPR to OFlow on reconstructing deforming cars at 10 observed time steps (left) and on estimating correspondences over time (right). To measure correspondence error, we (1) sample 2048 points from the representation at $\bar{t}_1$, (2) find their closest points on the GT mesh, and (3) advect the samples to $\bar{t}_{10}$ and measure the mean distance to the corresponding GT points at both steps. Tab. 4 reports median errors over all $\bar{t}_1$ and $\bar{t}_{10}$. For OFlow, samples are advected using the predicted flow field in physical space, while for CaSPR we simply use the same Gaussian samples at each step of the sequence.

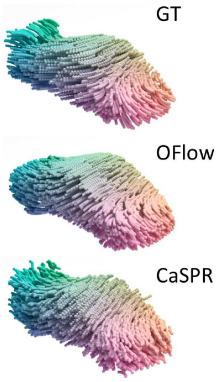

Figure 6: Deforming car reconstruction.

CaSPR outperforms OFlow on reconstruction due to overly-smoothed outputs from the occupancy network, while both methods accurately maintain correspondences over time. Note that CaSPR advects system state in a learned latent space and temporal correspondences *naturally emerge* from the CNF when using consistent base samples across timesteps. Fig. 6 visualizes sampled point trajectories for one sequence.

Cross-Instance Correspondences: We observe consistent behavior from the CNF across objects within a shape category too. Fig. 8 shows reconstructed frames from various chair and airplane sequences with points colored by their corresponding location in the sampled Gaussian (before the flow). Similar colors across instances indicate the same part of the base distribution is mapped there. This could potentially be used, for instance, to propagate labels from known to novel object instances.

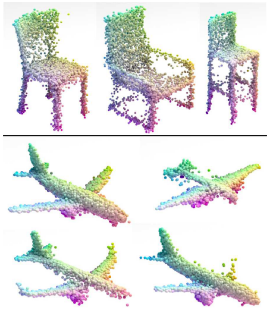

Figure 8: Cross-instance correspondences emerge naturally using a CNF.

Learning the Arrow of Time: A desirable property of ST representations is an understanding of the *unidirectionality* of time [20]: how objects evolve forward in time. We demonstrate this property with CaSPR by training on a dataset of 1000 sequences of a single rigid car where the camera always rotates counter-clockwise at a fixed distance (but random height). CaSPR achieves a median CD of 0.298 and **EMD of 7.114** when reconstructing held-out sequences *forward in time*. However, when the same test sequences are *reversed* by flipping the timestamps, accuracy drastically drops to CD 1.225 and **EMD 88.938**. CaSPR is sensitive to the arrow of time due to the directionality of the Latent ODE and the global temporal view provided by operating on an entire sequence jointly.

Shape & Motion Disentanglement We evaluate how well CaSPR disentangles shape and motion as a result of the latent feature splitting $\mathbf{z}^C = [\mathbf{z}^C_{\text{ST}}, \mathbf{z}^C_{\text{dyn}}]$. For this purpose, we transfer motion between two sequences by embedding both of them using TPointNet++, then taking the static feature $\mathbf{z}^C_{\text{ST}}$ from the first and the dynamic feature $\mathbf{z}^C_{\text{dyn}}$ from the second. Fig. 9 shows qualitative results where each row is a different sequence; the first frame of the shape sequence is on the left, the point trajectories of the motion sequence in the middle, and the final CaSPR-sampled trajectories using the combined feature are on the right. If these features perfectly disentangle shape and motion, we should see the shape of the first sequence with the motion of the second after reconstruction. Apparently, the explicit feature split in CaSPR does disentangle static and dynamic properties of the object to a large extent.

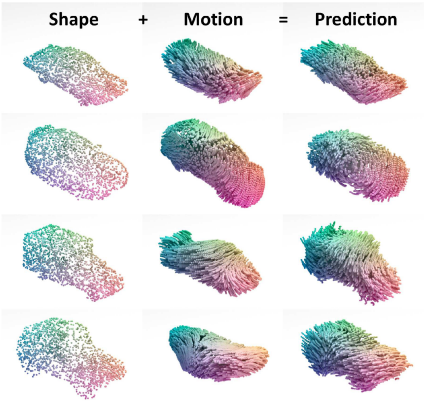

Figure 9: Disentanglement examples on warping cars data.

6 Conclusion

We introduced CaSPR, a method to canonicalize and obtain object-centric representions of raw point cloud sequences, which supports spatiotemporal sampling at arbitrary resolutions. We demonstrated CaSPR's utility on rigid and deformable object motion and in applications like spatiotemporal interpolation and estimating correspondences across time and instances.

Limitations and Future Work: CaSPR leaves ample room for future exploration. We currently only support batch processing, but online processing is important for real-time applications. Additionally, CaSPR is expensive to train. Our canonicalization step requires dense supervision of T-NOCS labels which may not be available for real data. While the network is well-suited for ST interpolation, the extrapolation abilities of CaSPR need further investigation. CaSPR is object-centric, and further work is needed to generalize to object collections and scenes. Additionally, outlier shapes can cause noisy sampling results and if the partial view of an object is ambiguous or the object is symmetric, TPointNet++ may predict a flipped or rotated canonical output.

Finally, using a single CNF for spatial sampling is fundamentally limited by an inability to model changes in topology [15, 19]. To capture fine-scale geometric details of shapes, this must be addressed.

Broader Impact

CaSPR is a fundamental technology allowing the aggregation and propagation of dynamic point cloud information – and as such it has broad applications in areas like autonomous driving, robotics, virtual/augmented reality and medical imaging. We believe that our approach will have a mostly positive impact but we also identify potential undesired consequences below.

Our method will enhance the capabilities of existing sensors and allow us to build models of objects from sparse observations. For instance, in autonomous driving or mixed reality, commonly used LIDAR/depth sensors are limited in terms of spatial and temporal resolution or sampling patterns. Our method creates representations that overcome these limitations due to the capability to continuously sample in space and time. This would enable these sensors to be cheaper and operate at lower spacetime resolutions saving energy and extending hardware lifespans. Our approach could also be useful in spatiotemporal information propagation. We can propagate sparse labels in the input over spacetime, leading to denser supervision. This would save manual human labeling effort.

Like other learning-based methods, CaSPR can produce biased results missing the details in the input. In a self driving scenario, if an input LIDAR point cloud only partially observes a pedestrian, CaSPR may learn representations that misses the pedestrian completely. If real-world systems rely excessively on this incorrect representation it could lead to injuries or fatalities. We look forward to conducting and fostering more research in other applications and negative impacts of our work.

Acknowledgments and Funding Disclosure

This work was supported by grants from the Stanford-Ford Alliance, the SAIL-Toyota Center for AI Research, the Samsung GRO program, the AWS Machine Learning Awards Program, NSF grant IIS-1763268, and a Vannevar Bush Faculty Fellowship. The authors thank Michael Niemeyer for providing the code and shape models used to generate the warping cars dataset. Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

References

[1] Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.: Learning representations and generative models for 3d point clouds. In: Proceedings of the International Conference on Machine Learning (ICML). pp. 40–49. PMLR (2018)

[2] Atanov, A., Volokhova, A., Ashukha, A., Sosnovik, I., Vetrov, D.: Semi-conditional normalizing flows for semi-supervised learning. arXiv preprint arXiv:1905.00505 (2019)

[3] Behl, A., Paschalidou, D., Donné, S., Geiger, A.: Pointflownet: Learning representations for rigid motion estimation from point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7962–7971 (2019)

[4] Bertsekas, D.P.: A distributed asynchronous relaxation algorithm for the assignment problem. In: IEEE Conference on Decision and Control. pp. 1703–1704 (1985)

[5] Biloš, M., Günnemann, S.: Equivariant normalizing flows for point processes and sets. arXiv preprint arXiv:2010.03242 (2020)

[6] Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)

[7] Chen, C., Li, C., Chen, L., Wang, W., Pu, Y., Carin, L.: Continuous-time flows for efficient inference and density estimation. In: Proceedings of the International Conference on Machine Learning (ICML) (2018)

[8] Chen, D., Li, J., Wang, Z., Xu, K.: Learning canonical shape space for category-level 6d object pose and size estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 11973–11982 (2020)

[9] Chen, S.S., Gopinath, R.A.: Gaussianization. In: Proceedings of the Advances in Neural Information Processing Systems (NeurIPS). pp. 423–429 (2001)

[10] Chen, T.Q., Rubanova, Y., Bettencourt, J., Duvenaud, D.K.: Neural ordinary differential equations. In: Proceedings of the Advances in Neural Information Processing Systems (NeurIPS) (2018)

[11] Chen, Z., Zhang, H.: Learning implicit fields for generative shape modeling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5939–5948 (2019)

[12] Choy, C., Gwak, J., Savarese, S.: 4d spatio-temporal convnets: Minkowski convolutional neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3075–3084 (2019)

[13] Choy, C.B., Xu, D., Gwak, J., Chen, K., Savarese, S.: 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 628–644. Springer (2016)

[14] Coddington, E.A., Levinson, N.: Theory of ordinary differential equations. Tata McGraw-Hill Education (1955)

[15] Cornish, R., Caterini, A.L., Deligiannidis, G., Doucet, A.: Relaxing bijectivity constraints with continuously indexed normalising flows. arXiv preprint arXiv:1909.13833 (2019)

[16] Davis, J.Q., Choromanski, K., Sindhwani, V., Varley, J., Lee, H., Slotine, J.J., Likhosterov, V., Weller, A., Makadia, A.: Time dependence in non-autonomous neural odes. In: ICLR Workshop on Integration of Deep Neural Models and Differential Equations (2020)

[17] Deng, H., Birdal, T., Ilic, S.: Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors. In: Proceedings of the European Conference on Computer Vision (ECCV) (September 2018)

[18] Deng, H., Birdal, T., Ilic, S.: Ppfnet: Global context aware local features for robust 3d point matching. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 195–205 (2018)

[19] Dupont, E., Doucet, A., Teh, Y.W.: Augmented neural odes. In: Proceedings of the Advances in Neural Information Processing Systems (NeurIPS). pp. 3134–3144 (2019)

[20] Eddington, A.: The nature of the physical world: The Giffor Lectures 1927, vol. 23. BoD–Books on Demand (2019)

[21] Finlay, C., Jacobsen, J.H., Nurbekyan, L., Oberman, A.M.: How to train your neural ode. arXiv preprint arXiv:2002.02798 (2020)

[22] Fischler, M.A., Bolles, R.C.: 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)

[23] Friedman, J.H.: Exploratory projection pursuit. Journal of the American statistical association **82**(397), 249–266 (1987)

[24] Gemici, M.C., Rezende, D., Mohamed, S.: Normalizing flows on riemannian manifolds. arXiv preprint arXiv:1611.02304 (2016)

[25] Grathwohl, W., Chen, R.T., Bettencourt, J., Sutskever, I., Duvenaud, D.: Ffjord: Free-form continuous dynamics for scalable reversible generative models. arXiv preprint arXiv:1810.01367 (2018)

[26] Groueix, T., Fisher, M., Kim, V.G., Russell, B.C., Aubry, M.: A papier-mâché approach to learning 3d surface generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 216–224 (2018)

[27] Grover, A., Dhar, M., Ermon, S.: Flow-gan: Combining maximum likelihood and adversarial learning in generative models. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)

[28] Gu, X., Wang, Y., Wu, C., Lee, Y.J., Wang, P.: Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3254–3263 (2019)

[29] Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., Bennamoun, M.: Deep learning for 3d point clouds: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI) (2020)

[30] Hanshu, Y., Jiawei, D., Vincent, T., Jiashi, F.: On robustness of neural ordinary differential equations. In: Proceedings of the International Conference on Learning Representations (ICLR) (2019)

[31] Hao, Z., Averbuch-Elor, H., Snavely, N., Belongie, S.: Dualsdf: Semantic shape manipulation using a two-level representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020)

[32] Kanazawa, A., Zhang, J.Y., Felsen, P., Malik, J.: Learning 3d human dynamics from video. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5614–5623 (2019)

[33] Kar, A., Tulsiani, S., Carreira, J., Malik, J.: Category-specific object reconstruction from a single image. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1966–1974 (2015)

[34] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

[35] Kingma, D.P., Dhariwal, P.: Glow: Generative flow with invertible 1x1 convolutions. In: Advances in Neural Information Processing Systems. pp. 10215–10224 (2018)

[36] Kobyzev, I., Prince, S., Brubaker, M.A.: Normalizing flows: Introduction and ideas. arXiv preprint arXiv:1908.09257 (2019)

[37] Köhler, J., Klein, L., Noé, F.: Equivariant flows: sampling configurations for multi-body systems with symmetric energies. arXiv preprint arXiv:1910.00753 (2019)

[38] Kutta, W.: Beitrag zur naherungsweisen integration totaler differentialgleichungen. Z. Math. Phys. **46**, 435–453 (1901)

[39] Lea, C., Flynn, M.D., Vidal, R., Reiter, A., Hager, G.D.: Temporal convolutional networks for action segmentation and detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 156–165 (2017)

[40] Lebowitz, J.L.: Boltzmann's entropy and time's arrow. Physics today **46**, 32–32 (1993)

[41] Li, X., Wang, H., Yi, L., Guibas, L.J., Abbott, A.L., Song, S.: Category-level articulated object pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3706–3715 (2020)

[42] Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: Pointcnn: Convolution on x-transformed points. In: Proceedings of the Advances in Neural Information Processing Systems (NeurIPS). pp. 820–830 (2018)

[43] Liu, J., Kumar, A., Ba, J., Kiros, J., Swersky, K.: Graph normalizing flows. In: Advances in Neural Information Processing Systems. pp. 13556–13566 (2019)

[44] Liu, X., Qi, C.R., Guibas, L.J.: Flownet3d: Learning scene flow in 3d point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019)

[45] Liu, X., Yan, M., Bohg, J.: Meteornet: Deep learning on dynamic 3d point cloud sequences. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 9246–9255 (2019)

[46] Lou, A., Lim, D., Katsman, I., Huang, L., Jiang, Q., Lim, S.N., De Sa, C.: Neural manifold ordinary differential equations. arXiv preprint arXiv:2006.10254 (2020)

[47] Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4460–4470 (2019)

[48] Mitra, N.J., Flöry, S., Ovsjanikov, M., Gelfand, N., Guibas, L.J., Pottmann, H.: Dynamic geometry registration. In: Symposium on geometry processing. pp. 173–182 (2007)

[49] Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A.: Occupancy flow: 4d reconstruction by learning particle dynamics. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 5379–5389 (2019)

[50] Novotny, D., Ravi, N., Graham, B., Neverova, N., Vedaldi, A.: C3dpo: Canonical 3d pose networks for non-rigid structure from motion. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 7688–7697 (2019)

[51] Onken, D., Fung, S.W., Li, X., Ruthotto, L.: Ot-flow: Fast and accurate continuous normalizing flows via optimal transport. arXiv preprint arXiv:2006.00104 (2020)

[52] Papamakarios, G., Nalisnick, E., Rezende, D.J., Mohamed, S., Lakshminarayanan, B.: Normalizing flows for probabilistic modeling and inference. arXiv preprint arXiv:1912.02762 (2019)

[53] Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: DeepSDF: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019)

[54] Pickup, L.C., Pan, Z., Wei, D., Shih, Y., Zhang, C., Zisserman, A., Scholkopf, B., Freeman, W.T.: Seeing the arrow of time. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2035–2042 (2014)

[55] Prantl, L., Chentanez, N., Jeschke, S., Thuerey, N.: Tranquil clouds: Neural networks for learning temporally coherent features in point clouds. In: Proceedings of the International Conference on Learning Representations (ICLR) (2020)

[56] Pumarola, A., Popov, S., Moreno-Noguer, F., Ferrari, V.: C-flow: Conditional generative flow models for images and 3d point clouds. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2020)

[57] Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 652–660 (2017)

[58] Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in neural information processing systems (2017)

[59] Rezende, D., Mohamed, S.: Variational inference with normalizing flows. In: International Conference on Machine Learning. pp. 1530–1538 (2015)

[60] Rezende, D.J., Racanière, S., Higgins, I., Toth, P.: Equivariant hamiltonian flows. arXiv preprint arXiv:1909.13739 (2019)

[61] Rippel, O., Adams, R.P.: High-dimensional probability estimation with deep density models. arXiv preprint arXiv:1302.5125 (2013)

[62] Rubanova, Y., Chen, T.Q., Duvenaud, D.K.: Latent ordinary differential equations for irregularly-sampled time series. In: Proceedings of the Advances in Neural Information Processing Systems (NeurIPS) (2019)

[63] Runge, C.: Über die numerische auflösung von differentialgleichungen. Mathematische Annalen **46**(2), 167–178 (1895)

[64] Runz, M., Buffier, M., Agapito, L.: Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In: International Symposium on Mixed and Augmented Reality (ISMAR). pp. 10–20. IEEE (2018)

[65] Salman, H., Yadollahpour, P., Fletcher, T., Batmanghelich, K.: Deep diffeomorphic normalizing flows. arXiv preprint arXiv:1810.03256 (2018)

[66] Shen, Y., Feng, C., Yang, Y., Tian, D.: Mining point cloud local structures by kernel correlation and graph pooling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). vol. 4 (2018)

[67] Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., Fitzgibbon, A.: Scene coordinate regression forests for camera relocalization in rgb-d images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2930–2937 (2013)

[68] Sitzmann, V., Thies, J., Heide, F., Nießner, M., Wetzstein, G., Zollhofer, M.: Deepvoxels: Learning persistent 3d feature embeddings. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2437–2446 (2019)

[69] Sridhar, S., Rempe, D., Valentin, J., Bouaziz, S., Guibas, L.J.: Multiview aggregation for learning category-specific shape reconstruction. In: Proceedings of the Advances in Neural Information Processing Systems (NeurIPS) (2019)

[70] Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.H., Kautz, J.: Splatnet: Sparse lattice networks for point cloud processing. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2530–2539 (2018)

[71] Sun, H., Mehta, R., Zhou, H.H., Huang, Z., Johnson, S.C., Prabhakaran, V., Singh, V.: Dual-glow: Conditional flow-based generative model for modality transfer. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 10611–10620 (2019)

[72] Tabak, E.G., Turner, C.V.: A family of nonparametric density estimation algorithms. Communications on Pure and Applied Mathematics **66**(2), 145–164 (2013)

[73] Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: Kpconv: Flexible and deformable convolution for point clouds. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 6411–6420 (2019)

[74] Varanasi, K., Zaharescu, A., Boyer, E., Horaud, R.: Temporal surface tracking using mesh evolution. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 30–43. Springer (2008)

[75] Varol, G., Laptev, I., Schmid, C.: Long-term temporal convolutions for action recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI) **40**(6), 1510–1517 (2017)

[76] Vedula, S., Baker, S., Rander, P., Collins, R., Kanade, T.: Three-dimensional scene flow. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (1999)

[77] Villani, C.: Optimal transport: old and new, vol. 338. Springer Science & Business Media (2008)

[78] Wang, H., Sridhar, S., Huang, J., Valentin, J., Song, S., Guibas, L.J.: Normalized object coordinate space for category-level 6d object pose and size estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2642–2651 (2019)

[79] Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., Jiang, Y.G.: Pixel2mesh: Generating 3d mesh models from single rgb images. In: Proceedings of the European Conference on Computer Vision (ECCV) (2018)

[80] Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. ACM Transactions on Graphics **38**(5), 1–12 (2019)

[81] Wang, Z., Li, S., Howard-Jenkins, H., Prisacariu, V., Chen, M.: Flownet3d++: Geometric losses for deep scene flow estimation. In: IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 91–98 (2020)

[82] Wu, Y., He, K.: Group normalization. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 3–19 (2018)

[83] Yang, G., Huang, X., Hao, Z., Liu, M.Y., Belongie, S., Hariharan, B.: Pointflow: 3d point cloud generation with continuous normalizing flows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 4541–4550 (2019)

[84] Yang, Z., Litany, O., Birdal, T., Sridhar, S., Guibas, L.: Continuous geodesic convolutions for learning on 3d shapes. In: IEEE Winter Conference on Applications of Computer Vision (WACV) (2021)

[85] Yew, Z.J., Lee, G.H.: Rpm-net: Robust point matching using learned features. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 11824–11833 (2020)

[86] Zakharov, S., Kehl, W., Bhargava, A., Gaidon, A.: Autolabeling 3d objects with differentiable rendering of sdf shape priors. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 12224–12233 (2020)

[87] Zhang, C., Fiore, M., Murray, I., Patras, P.: Cloudlstm: A recurrent neural model for spatiotemporal point-cloud stream forecasting. arXiv preprint arXiv:1907.12410 (2019)

[88] Zhang, H., Gao, X., Unterman, J., Arodz, T.: Approximation capabilities of neural ordinary differential equations. arXiv preprint arXiv:1907.12998 (2019)

[89] Zhang, L., Wang, L., et al.: Monge-ampere flow for generative modeling. arXiv preprint arXiv:1809.10188 (2018)

[90] Zhang, S., Guo, S., Huang, W., Scott, M.R., Wang, L.: V4d: 4d convonlutional neural networks for video-level representation learning. In: Proceedings of the International Conference on Learning Representations (ICLR) (2020)

[91] Zhao, Y., Birdal, T., Deng, H., Tombari, F.: 3d point capsule networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1009–1018 (2019)

[92] Zhao, Y., Birdal, T., Lenssen, J.E., Menegatti, E., Guibas, L., Tombari, F.: Quaternion equivariant capsule networks for 3d point clouds. arXiv preprint arXiv:1912.12098 (2019)

Appendices

We expand on discussions in Sec. A, provide additional evaluations in Sec. B, explain details of dataset generation and architecture implementation in Sec. C and Sec. D, and give details of experiments from the main paper in Sec. E.

A Discussions

Remarks on ODE-Nets The requirements of homeomorphisms and differentiability impose certain limitations. First, neural ODEs lack a universal approximation capability as non-intersecting trajectories cannot learn to approximate arbitrary topologies [88][2]. On the other hand, it is also shown that this very property brings intrinsic robustness to ODE-Nets [30]. Moreover, the requirement of invertibility in CNFs is proven to hamper the approximation quality of the target distribution [15]. In fact, for a perfect recovery and likelihood evaluation, non-invertibility is a requirement [15]. Nonetheless, the extent to which these limitations restrict the applicability of Neural ODEs and CNFs is still an active research topic.

Why can't we use existing point cloud networks as a canonicalizer? Extending PointNet++ to time (similar to MeteorNet [45]) requires some form of a spatiotemporal neighborhood query or using time as an auxiliary input feature diminishing its contribution. Spatiotemporal neighborhood queries are undesirable as they necessitate difficult hyperparameter tuning and limit the network's ability to holistically understand the motion. For example, learning the arrow of time (as CaSPR does in Sec. 5 of the main paper) would be difficult when using local spatiotemporal queries. PointNet can somewhat remedy this by operating on the full 4D point cloud at once, treating time equally important as the spatial dimensions. However, we found that PointNet by itself is incapable of extracting descriptive local features, which are essential for an accurate mapping to T-NOCS.

On the arrow of time Due to the second law of thermodynamics, the entropy of an isolated system tends to increase with time, making the direction of time irreversible [40] *i.e. it is more common for a motion to cause multiple motions than for multiple motions to collapse into one consistent motion* [54]. This causality is confirmed in computer vision by showing that the statistics of natural videos are not symmetric under time reversal [54]. Any method processing spacetime inputs should then be sensitive to this direction so as to yield distinctive representations rather than being *invariant* to it. As shown in the experiments of the main paper, thanks to the inclusion of timestamps and the Latent ODE advecting forward in time, CaSPR is highly aware of this unidirectionality and it is one of the reasons why it can extract robust spatiotemporal latent features.

On disentanglement In the main paper, we have demonstrated experimentally that static and dynamic feature disentanglement is achieved to a large extent. Note that CaSPR involves no mechanism that can guarantee a theoretically disentangled latent space such as the one of [92]. Our design *softly* encourages the canonicalization network to respect the subspace nature by only advecting the dynamic feature with the ODE. Though this is not a CaSPR-specific drawback and many SoTA disentanglement networks rely upon the same intuition.

Limitations Using a CNF to sample the object surface does come with some limitations as mentioned in prior work [83] and discussed above. The inherent properties of CNFs may hamper the ability to capture fine-scale geometric detail. We observe this in chairs with back slats and other thin structures that are not captured by our Reconstruction CNF as shown in the left panel of Fig. A1. Additionally, outlier shapes can cause noisy sampling results (shown in the middle). One current limitation of TPointNet++ is its inability to handle symmetry when canonicalizing a point cloud sequence. If the partial view of an object is ambiguous or the object is symmetric, TPointNet++ may predict a flipped or rotated canonical output as shown in the right panel.

[2] Augmented-Neural ODEs [64] propose to operate on a higher dimensional space as one workaround.

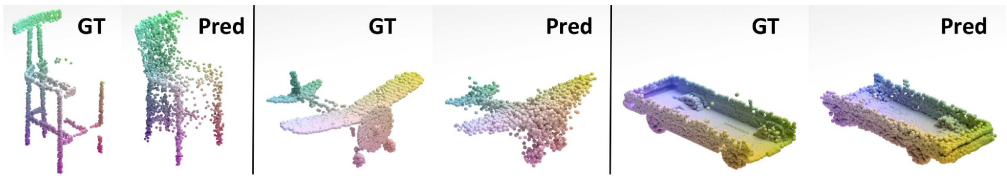

Figure A1: Failure cases of CaSPR. The CNF has difficulty capturing local details and very thin structures (left) along with uncommon shapes (middle). TPointNet++ has trouble with symmetry or ambiguity in partial views, resulting in reflected or rotated predictions (right).

B Additional Evaluations

We provide evaluations of CaSPR omitted from the main paper due to space constraints. Please see Section 5 of the main paper for an explanation of evaluation metrics and the primary results.

Table B1: CaSPR ablations for reconstruction of rigid car sequences over 10 observed frames.

Method	CD	EMD	NFE
No $\mathcal{L}_c$	0.605	11.482	**38.2**
No Factorization	0.635	11.249	101.3
No Input Aug	0.577	10.253	38.9
Full Arch	**0.566**	**10.103**	39.6

B.1 Ablation Study

We compare the full CaSPR architecture (*Full Arch*) to multiple ablations in Tab. B1. This includes: (i) not using the canonicalization loss (*No* $\mathcal{L}_c$), (ii) not factorizing the latent ST feature and instead feeding the entire vector to the Latent ODE (*No Factorization*), and (iii) using no pairwise terms (see Sec. D) as input augmentation (*No Input Aug*). In addition to reconstruction metrics, we report the mean number of function evaluations (NFE) for the Latent ODE. This is the average number of times the ODE solver queries the dynamics network while integrating forward in time for a single sequence. Each method is trained on the rigid cars category and reconstructs all 10 input frames for evaluation. The full CaSPR architecture performs best. Note that the static/dynamic feature factorization is especially important to limit the complexity of Latent ODE dynamics.

Table B2: Reconstruction errors with varying numbers of input points per frame for rigid car motion.

Num Points	CD	EMD
2048	0.5657	10.1028
1024	**0.5486**	**10.0339**
512	0.5904	10.5188
256	0.8222	13.8275
128	1.4730	20.7233

B.2 Sparsity in Space and Time

We evaluate CaSPR's ability to reconstruct partial point cloud sequences from the rigid car category under sparsity in both space and time. Given 10 input frames, Tab. B2 shows the performance for reconstructing all 10 observed frames with a varying number of points available at each frame. Performance is consistent until 256 or fewer points are given at which point it drops off rapidly. Tab. B3 shows performance when varying the number of available observed timesteps for each test sequence. Observed timesteps are distributed as evenly as possible over the 10-step sequence for this evaluation. Performance is stable even with 3 observed frames, but does significantly drop when only 2 frames are given (*i.e.* the first and last steps).

Table B3: Reconstruction errors with a varying number of observed input frames.

	Observed		*Unobserved*	
Num Observed	CD	EMD	CD	EMD
10 steps	**0.5657**	**10.1028**	—	—
7 steps	0.5701	10.3406	**0.5609**	**10.0304**
5 steps	0.5664	10.4310	0.5620	10.3374
3 steps	0.5904	11.4641	0.5837	11.2586
2 steps	0.7095	14.8348	0.7233	16.3499

B.3 Reconstructing Longer Sequences

We evaluate CaSPR when trained on the rigid motion car dataset with sequences of 25 frames (rather than 10 as in the main paper). During training, we randomly subsample 10 frames (rather than 5) from each sequence, and evaluate with the full 25-frame sequence as input (rather than 10). Tab. B4 shows reconstruction performance compared to the model in the main paper which uses the 10-frame sequence dataset. We see there is a minimal difference in performance, indicating CaSPR is capable of handling longer-horizon motion.

Table B4: Reconstruction errors for longer sequences on rigid car data.

Test Seq Length	CD	EMD
10	0.566	**10.103**
25	**0.534**	10.815

B.4 Multi-Category Model

We evaluate CaSPR when trained on all shape categories together: cars, chairs, and airplanes. This determines the extent of the category-level restriction on our method. Results compared to models trained on each category separately are shown in Tab. B5. Models are evaluated by reconstructing all 10 observed time steps. As expected, there is a performance drop when training a single joint model, however errors are still reasonable and in most cases better than the *PointFlow* baseline in terms of EMD (see Tab. 2 in main paper).

Table B5: Reconstruction errors training on all categories jointly.

Train Data	Test Data	CD	EMD
Cars	Cars	**0.566**	**10.103**
All	Cars	0.728	13.631
Chairs	Chairs	**0.715**	**13.009**
All	Chairs	1.231	15.632
Airplanes	Airplanes	**0.231**	**6.026**
All	Airplanes	0.391	8.213
All	All	0.798	12.578

B.5 Canonicalizing for Deformation

We evaluate the ability of TPointNet++ to canonicalize non-rigid transformations. Given a deforming car sequence from the Warping Cars dataset, the task is to remove the deformation at each step, leaving the base shape without any warping. To achieve this, we train TPointNet++ with $\mathcal{L}_c$ only, and supervise every step in a sequence with the same GT canonical point cloud that contains no deformation. Note that Warping Cars is already canonical in terms of rigid transformations, so the network needs to learn to factor out non-rigid deformation only. Results are shown in Tab. B6 where we compare TPointNet++ to a baseline that simply copies the input points to the output (*Identity*, which performs reasonably since there is no rigid transformation). *Identity* trivially gives a perfect time error, but TPointNet++ achieves a much lower spatial error, effectively removing the deformation from each step. This is qualitatively shown in Fig. E6. This strategy of canonicalization offers an explicit way to extract temporal correspondences over time, rather than relying on the CNF to naturally exhibit correspondences (main paper Sec. 5).

Table B6: Canonicalization performance for deforming cars.

Method	Spatial Err	Time Err
Identity	0.0583	**0.0000**
TPointNet++	**0.0221**	0.0012

B.6 Label Propagation through Canonicalization

We evaluate the ability of T-NOCS canonicalization to establish correspondences by propagating point-wise labels both throughout a sequence and to new sequences of different object instances. Given a semantic segmentation of the partial point cloud at the *first* frame of a sequence at time s_1, the first task is to label all subsequent steps in the sequence at times $s_2, \ldots, s_k$, *i.e*. propagate the segmentation forward in time. Secondly, we want to label all frames of sequences containing *different* object instances *i.e*. propagate the segmentation to different objects of the same class. We achieve both through canonicalization with TPointNet++: all frames in each sequence are mapped to T-NOCS, then unknown points are labeled by finding the closest point in the given labeled frame at s_1. If the closest point in s_1 is not within a distance of 0.05 in the canonical space, it is marked "Unknown". This may happen if part of the shape is not visible in the first frame due to self-occlusions.

Results of this label propagation for a subset of the chairs (1315 sequences) and airplanes (1215 sequences) categories of the rigid motion test set are shown in Tab. B7. We report median point-wise accuracy over all points (*Total Acc*) and for points successfully labeled by our approach (*Known Acc*). For the instance propagation task, we randomly use $1/3$ of test sequences as "source" sequences where the first frame is labeled, and the other $2/3$ are "target" sequences to which labels are propagated. In this case, accuracy is reported only for target sequences. Qualitative results are shown in Fig. B1.

Table B7: Segmentation label propagation performance. *Total Acc* is point-wise accuracy over all points; *Known Acc* is only for points that our method successfully labels.

Task	Category	Total Acc	Known Acc
Temporal Propagation	Chairs	0.9419	0.9804
	Airplanes	0.9580	0.9676
Instance Propagation	Chairs	0.6553	0.8425
	Airplanes	0.7744	0.8006

B.7 Extrapolating Motion

We evaluate CaSPR's ability to extrapolate future motion without being explicitly trained to do so. In particular, the model is given the first 5 frames in each sequence and must predict the following 5 frames. The ability to predict future motion based on the

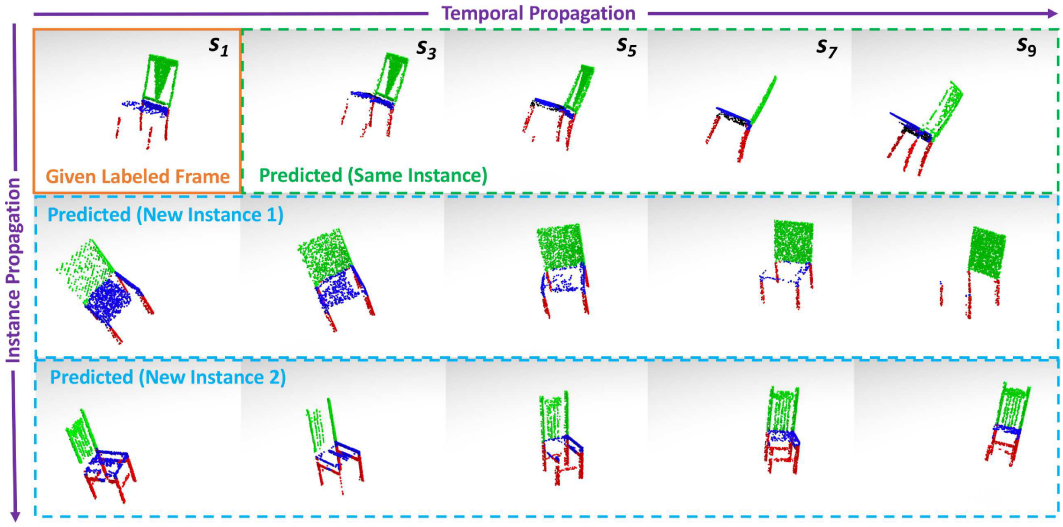

Figure B1: Example of semantic segmentation label propagation over time and across instances through T-NOCS canonicalization. The given labels in the first frame of the top sequence (orange box) are transferred to later frames in the same sequence (green dashed box) and to other sequences with different object instances (blue dashed boxes) by comparing to the labeled frame in the shared canonical space.

learned prior would be valuable in real-time settings. We evaluate the already-trained full reconstruction models for each object category (from Tab. 2 in the main paper). Note that these models are supervised with observed frames - they are not trained to predict unseen future states. Results are shown in Tab. B8. Clearly there is a sharp performance drop between observed and extrapolated frames as we might expect, though performance is actually on par with the AtlasNet baseline (Tab. 2, main paper) in some cases. We note that qualitatively, the model produces reasonable future motion based on what it has seen and even hallucinates unseen parts of the shape, though it cannot handle sudden changes in direction.

Table B8: Reconstruction of extrapolated frames.

	5 Observed		*5 Extrapolated*	
Category	**CD**	**EMD**	**CD**	**EMD**
Cars	0.597	9.833	1.023	21.055
Chairs	0.687	12.502	1.010	20.648
Airplanes	0.224	5.719	0.286	9.625

C Datasets Details

Rigid Motion Dataset Please see Section 5 of the main paper for an introduction to our new dataset containing rigid motion for ShapeNet [6] cars, chairs, and airplanes. This simulated dataset gives us the ability to capture a wide range of trajectories and acquire the necessary inputs and supervision to train and evaluate CaSPR.

We generate these motions within the Unity game engine[3]. For each object instance, we simulate a camera trajectory around the object (placed at the origin) that starts at a random location and continues for 50 timesteps. The camera always points towards the origin and its location is parameterized as a point on the surface of a sphere centered at the origin: by a longitudinal and latitudinal angle along with a radius. To produce a trajectory, each of these parameters is gradually increased or decreased independently. When a parameter reaches a set limit, its direction is reversed, producing interesting and challenging motions. At each step of the trajectory, a depth map and NOCS map [78] are rendered from the current camera view. An example NOCS map from the dataset is shown in Fig. C1. Example camera trajectories and the resulting aggregate canonical point cloud are shown in Fig. E4.

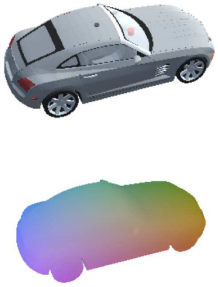

Figure C1: NOCS map from rigid motion dataset.

[3]https://unity.com/

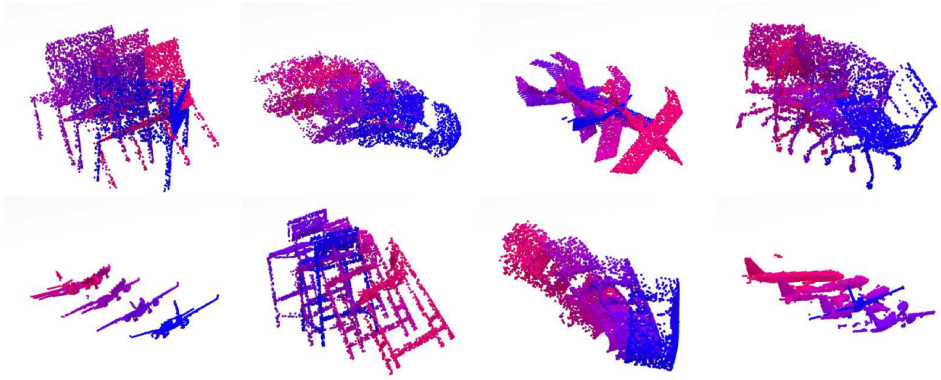

Figure C2: Examples from the rigid motion dataset. Partial point cloud sequences resulting from rendered data depth maps are shown; color shifts from blue to red over time.

The rendered frames are further processed to produce the final dataset of raw depth and canonical T-NOCS point cloud sequences. The rendered trajectory for each object instance is split into 5 sequences (with 10 steps each). 4096 pixels on the object are uniformly sampled from each depth map to extract *raw* partial point cloud sequences in the world (camera) frame that are used as the input to CaSPR. Examples of these partial sequences are shown in Fig. C2. Each input point cloud in a sequence is given a timestamp in uniform steps from 0.0 to 5.0. The same sampled pixels are taken from the NOCS map to extract a corresponding canonical partial point cloud and given a timestamp from 0.0 to 1.0: this represents the supervision for CaSPR. In total, the car category contains 2527 object instances ($12,635$ sequences), chairs contains 5000 objects ($25,000$ sequences), and airplanes has 4045 objects ($20,225$ sequences). Each category is split 80/10/10 into train/val/test. The val/test sets are entirely made up of object instances and camera motions that do not appear in the training split.

Note that during training and inference, only a subset of the available 4096 points at each step in the dataset are used, as detailed in the main paper (usually 1024 during training and 2048 during evaluation). Additionally, during training a subset of the available 10 frames are randomly sampled from each sequence, giving non-uniform step sizes between observations. These subsampled sequences are shifted so that $s_1 = 0.0$ before being given to CaSPR, making things practically easier as it ensures that the Latent ODE always starts from $t_1 = 0$ for any sequence in a batch.

Warping Cars Dataset In Section 5.1 of the main paper ("Non-Rigid Reconstruction and Temporal Correspondences"), we use a variation of the Warping Cars dataset from Occupancy Flow (OFlow) [49]. We generate our version of this dataset with code kindly provided by the authors of that work. The dataset contains the same car models as our rigid motion dataset, however they are watertight versions that allow determining occupancy, which is needed to train OFlow. Same as the rigid motion dataset, we generate 5 sequences for each car instance with 10 frames of motion each. Consistent with the OFlow paper, we sample 100k points per sequence on the surface of the object that are in correspondence over time and can be used as inputs to CaSPR and OFlow; we also sample 100k points in the unit cube containing the object with corresponding occupancy labels for OFlow. Note that this data gives point clouds on the *complete* object rather than the partial surface, and there is no rigid motion in the dataset – only deformation. This means the sequences are already canonical in the sense that cars are consistently aligned and scaled. We also use input timestamps from 0.0 to 1.0, so the data is already canonical in time as well.

D Implementation Details

We next cover additional architectural and training details of our method. Please see Section 4.1 of the main paper for the primary discussion of our architecture and training procedure. We implement our method using PyTorch[4].

[4]https://pytorch.org/

TPointNet++ The PointNet [57] component operates on the entire 4D input point cloud and extracts a 1024-dimensional global feature and 64-dimensional per-point features. We use the vanilla classification PointNet architecture with 3 shared fully-connected (FC) layers (64, 128, 1024), ReLU non-linearities, and a final max-pool function. The per-point features come from the output of the first FC layer, while the global feature is the output of the max-pool. We do not use the input or feature transform layers, and replace all batch normalization with group normalization [82] using 16 groups, which is crucial to good performance with small batch sizes.

The PointNet++ [58] component operates on each frame of the point cloud sequence independently and does not receive the timestamp as input. The input points to this part of the network are augmented with pairwise terms x^2, y^2, z^2, xy, yz, and xz, which we found improves reconstruction performance (see Sec. B.1). We use a modified version of the segmentation architecture which contains 5 set abstraction (SA) layers (PointNet dimensions, radii, number points out): $([[16, 16, 32], [32, 32, 64]], [0.8, 0.4], 1024) \rightarrow ([[32, 32, 64], [32, 32, 64]], [0.4, 0.2], 512) \rightarrow ([[64, 64, 128], [64, 96, 128]], [0.2, 0.1], 256) \rightarrow ([[128, 256, 256], [128, 256, 256]], [0.1, 0.05], 64) \rightarrow ([[256, 256, 512], [256, 256, 512]], [0.05, 0.02], 16)$. These are followed by 5 feature propagation (FP) layers which each have 2 layers with hidden size 512, and a final shared MLP with layers $(512, 512)$ to produce the final per-point 512-dimensional local feature. ReLU non-linearities are used throughout, and we again replace all batch normalization with group normalization [82] using 16 groups.

The final shared MLP which processes the concatenated features from PointNet and PointNet++ also uses group normalization and ReLU.

There are a few things of note with this architecture. First of all, it avoids any spatiotemporal neighborhood queries since time is handled entirely with PointNet which treats the timestamps as an additional spatial dimension. This allows the network to decide which time windows are most important to focus on. Second, the architecture can easily generalize to sequences with differing numbers of points and frames since both are processed almost entirely independently (the only components affected by changing these at test-time are the PointNet max-pooling and the PointNet++ spatial neighborhood queries).

Latent ODE The Latent ODE is given a 64-dimensional latent state $\mathbf{z}_0 \triangleq \mathbf{z}_{\text{dyn}}^C$ which can be advected to any canonical timestamp from 0.0 to 1.0. The dynamics of the Latent ODE is an MLP with 3 hidden layers (512, 512, 512) which uses Tanh non-linearities. We use the *torchdiffeq* package[5] [10] which implements both the ODE solver along with the adjoint method to enable backpropagation. We use the *dopri15* solver which is an adaptive-step Runge-Kutta 4(5) method. We use a relative tolerance of 1e-3 and absolute tolerance of 1e-4 both at training and test time.

Reconstruction CNF Our reconstruction CNF adapts the implementation of FFJORD [25] for point clouds from PointFlow [83]. The dynamics of the CNF are parameterized by a neural network that uses 3 hidden *ConcatSquashLinear* layers (512, 512, 512), which are preceeded and followed by a *Moving Batch Normalization* layer. We use Softplus non-linearities after each layer. Please see [83] for full details. In short, each layer takes as input the current hidden state (512-dimensional at hidden layers or 3-dimensional x, y, z at the first layer), the conditioning shape feature (1600-dimensional in CaSPR), and the current time of the flow (scalar), and uses this information to update the hidden state (or output the 3-dimensional derivative at the last layer). The ODE is again solved using *dopri15*, this time with both a relative and absolute tolerence of 1e-5. We use the adjoint method for backpropagation and jointly optimize for the final flow time T along with the parameters of network.

Training and Inference In practice, the full loss function is $\mathcal{L} = w_r \mathcal{L}_r + w_c \mathcal{L}_c$ where the contributions of the reconstruction and canonicalization terms are weighted as $w_r = 0.01$ and $w_c = 100$ as to be similar scales. No weight decay is used. We use the Adam [34] optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$) with a learning rate of 1e-4. During training, we periodically compute the validation set loss, and after convergence use the weights with the best validation performance as the final trained model. The number of epochs trained depends on the dataset and the task. We train across up to 4 NVIDIA Tesla V100 GPUs which allows for a batch size of up to 20 sequences of 5 frames each. As noted in previous work [83], solving and backpropagating through ODEs (two in our case: Latent and CNF)

[5] https://github.com/rtqichen/torchdiffeq

results in slow training: it takes about 5 days for the full CaSPR architecture using the multi-gpu setup. The full CaSPR network contains about 16 million trainable parameters. Inference for a 10-step sequence of rigid car motion with 2048 points at each step takes on average 0.598 seconds.

E Experimental Details and Supplemental Results

Here we give details of experiments shown in Section 5 of the main paper along with some supporting results for these experiments (*e.g.* means, standard deviations, and visualizations).

Evaluation Procedure To evaluate reconstruction error, we use the Chamfer Distance (CD) and Earth Mover's Distance (EMD). For our purposes, we define the CD and EMD between two point clouds $\mathcal{X}_1, \mathcal{X}_2$ each with N points as

$$d_{CD}\left(\mathcal{X}_1, \mathcal{X}_2\right) = \frac{1}{N} \sum_{\mathbf{x}_1 \in \mathcal{X}_1} \min_{\mathbf{x}_2 \in \mathcal{X}_2} \|\mathbf{x}_1 - \mathbf{x}_2\|_2^2 + \frac{1}{N} \sum_{\mathbf{x}_2 \in \mathcal{X}_2} \min_{\mathbf{x}_1 \in \mathcal{X}_1} \|\mathbf{x}_1 - \mathbf{x}_2\|_2^2$$

$$d_{EMD}\left(\mathcal{X}_1, \mathcal{X}_2\right) = \min_{\phi: \mathcal{X}_1 \to \mathcal{X}_2} \frac{1}{N} \sum_{\mathbf{x}_1 \in \mathcal{X}_1} \|\mathbf{x}_1 - \phi(\mathbf{x}_1)\|_2^2$$

where $\phi : \mathcal{X}_1 \to \mathcal{X}_2$ is a bijection. In practice, we use a fast approximation of the EMD based on [4]. Both CD and EMD are always reported multiplied by 10^3.

As noted in the main paper, for these reconstruction metrics and the canonicalization error metrics, we report the median values over all test frames. This is motivated by the fact that ShapeNet [6] contains some outlier shapes which result in large errors that unfairly bias the mean and do not accurately reflect comprehensive method performance. For completeness, we also report mean and standard deviation for these metrics in this document for main paper experiments. Note that CD and EMD, along with the spatial canonicalization error, are all reported in the canonical space where the shape lies within a unit cube. This helps intuit the severity of reported errors.

Table E1: Canonicalization performance mean and (standard deviation). Supplements Tab. 1 in the main paper.

Method	Category	Spatial Err	Time Err
MeteorNet	Cars	0.0834 (0.0801)	**0.0002** (0.0015)
PointNet++ No Time		0.0649 (0.0468)	—
PointNet++ w/ Time		0.0715 (0.0811)	0.0006 (0.0012)
PointNet		0.0485 (0.0952)	0.0016 (0.0015)
TPointNet++ No Aug		0.0225 (0.0501)	0.0015 (0.0014)
TPointNet++ No Time		**0.0224** (0.0570)	—
TPointNet++	Cars	0.0229 (0.0617)	0.0013 (0.0012)
TPointNet++	Chairs	0.0162 (0.0337)	0.0008 (0.0006)
TPointNet++	Airplanes	0.0148 (0.0412)	0.0009 (0.0007)

Although we randomly subsample 1024 points at each frame for training, during evaluation we always use the same 2048 points (unless specifically stated otherwise) to make evaluation consistent across compared methods. Unless otherwise stated, CaSPR and all compared baselines reconstruct the same number of points as in the input (*e.g.* for evaluation, each input frame has 2048 points, so we sample 2048 points from our Reconstruction CNF).

Canonicalization In this experiment, we train TPointNet++ by itself with only the canonicalization loss $\mathcal{L}_c$ on each category of the rigid motion dataset. In order to make the number of parameters comparable across all baselines, we use hidden layers of size 1024 (rather than 1600) in the final shared MLP for the full TPointNet++ architecture only. We compare to the following baselines which are all trained with the same $\mathcal{L}_c$:

- *MeteorNet* [45]: A recent method that extends PointNet++ to process point cloud sequences through spatiotemporal neighborhood queries. We adapt the *MeteorNet-seg* version of the architecture with *direct grouping* for our task by adding an additional *meteor direct module* layer, as well as two fully connected layers before the output layer. Additionally, we slightly modify feature sizes to make the model capacity comparable to other methods. We found the spatiotemporal radii hyperparameters difficult to tune and in the end we opted for 10 uniformly sampled radii between $(0.03, 0.05)$ in the first layer, which were doubled in each subsequent layer.
- *PointNet++ No Time*: An ablation of TPointNet++ that removes the PointNet component. This leaves PointNet++ processing each frame independently followed by the shared MLP, and therefore has no notion of time.

Table E3: Partial surface sequence reconstruction results showing mean and (standard deviation). Supplements Tab. 2 in the main paper.

		10 Observed		*3 Observed*		*7 Unobserved*	
Method	**Category**	**CD**	**EMD**	**CD**	**EMD**	**CD**	**EMD**
PointFlow	Cars	**0.537** (0.272)	15.986 (11.130)	**0.538** (0.270)	**15.967** (11.065)	**0.700** (0.732)	17.362 (12.276)
CaSPR-Atlas	Cars	0.814 (1.729)	26.922 (28.562)	0.874 (2.051)	29.171 (29.479)	0.853 (1.705)	26.416 (27.582)
CaSPR	Cars	0.795 (1.048)	**14.242** (21.619)	0.846 (1.261)	16.564 (24.296)	0.824 (1.108)	**16.217** (23.011)
PointFlow	Chairs	**0.907** (0.519)	20.254 (11.938)	**0.907** (0.514)	20.225 (11.899)	1.245 (1.299)	21.971 (13.417)
CaSPR-Atlas	Chairs	1.007 (1.243)	54.406 (24.970)	1.030 (1.221)	54.827 (25.250)	1.061 (1.277)	52.964 (24.355)
CaSPR	Chairs	1.013 (1.426)	**15.287** (9.837)	0.972 (1.498)	**15.757** (11.154)	**1.000** (1.542)	**16.145** (11.620)
PointFlow	Airplanes	**0.367** (0.366)	11.852 (8.768)	**0.366** (0.363)	11.862 (8.725)	**0.446** (0.527)	12.335 (9.146)
CaSPR-Atlas	Airplanes	0.587 (1.196)	23.444 (17.386)	0.653 (1.369)	23.165 (16.932)	0.663 (1.400)	22.661 (16.853)
CaSPR	Airplanes	0.536 (1.468)	**8.827** (12.650)	0.536 (1.682)	**8.992** (13.219)	0.530 (1.673)	**9.031** (12.792)

- *PointNet++ w/ Time*: This is the same ablation as above, but modified so that the PointNet++ receives the timestamp of each point as an additional input feature. Note that local neighborhood queries are still performed only on spatial points, but they may be across timesteps so we use increased radii of $(0.05, 0.1, 0.2, 0.6, 1.2, 2.0)$. This baseline represents a naive way to incorporate time, but dilutes its contributions since it is only an auxiliary feature.
- *PointNet*: An ablation of TPointNet++ that removes the PointNet++ component. This leaves only PointNet operating on the full 4D spatiotemporal point cloud. This baseline treats time equally as the spatial dimensions, but inherently lacks local geometric features.
- *TPointNet++ No Time*: An ablation of TPointNet++ that only regresses the spatial part of the T-NOCS coordinate (and not the normalized timestamp). This baseline still takes the timestamps as input, it just doesn't regress the last time coordinate.
- *TPointNet++ No Aug*: An ablation of TPointNet++ that does not augment the input points to PointNet++ with pairwise terms as described previously. This baseline was omitted from the main paper for brevity, so a comparison of median performance is shown in Tab. E2.

Each model is trained for 220 epochs on the cars category. TPointNet++ is trained for 120 and 70 epochs on the airplanes and chairs categories, respectively, due to the increased number of objects. Median canonicalization errors are in Tab. 1 of the main paper; the mean and standard deviations are shown in Tab. E1.

Table E2: Canonicalization performance without input augmentation.

Method	**Category**	**Spatial Err**	**Time Err**
No Aug	Cars	0.0138	0.0012
Full Arch	Cars	**0.0118**	**0.0011**

Representation and Reconstruction In this experiment, we compare the full CaSPR architecture to two baselines on the task of reconstructing a partial point cloud sequence.

The baselines represent one alternative to achieve spatial continuity, and one to achieve temporal continuity. The *CaSPR-Atlas* baseline is the full CaSPR architecture as described, but replaces the Reconstruction CNF with an AtlasNet [26] decoder. We use the same decoder as the original AtlasNet. This decoder contains 64 MLPs, each responsible for transforming a patch to the partial visible surface at a desired timestep. Each MLP contains 4 hidden layers $(1600, 1600, 800, 400)$ with Tanh activation functions. This version of CaSPR is still trained with the auxiliary canonicalization task ($\mathcal{L}_c$ loss), but the reconstruction loss is now a Chamfer distance since AtlasNet does not support likelihood evaluations like a CNF. We use group normalization [82] instead of batch normalization within the decoder to improve performance with small batch sizes.

The *PointFlow* [83] baseline uses their deterministic autoencoder architecture. This follows the autoencoding evaluations from the original paper and uses a PointNet-like encoder to extract a shape feature, which conditions a CNF decoder. This version of the model is trained only with the reconstruction likelihood objective from the CNF, and does not use the various losses associated with the VAE formulation of their architecture. To make it a fair comparison, we increase the size of the shape feature bottleneck to 1600. The CNF decoder uses a dynamics MLP with 3 hidden layers of size $(512, 512, 512)$, just like CaSPR. Also like CaSPR, we train *PointFlow* with a learning rate of 1e-4, which we found to decrease the complexity of dynamics and therefore training time.

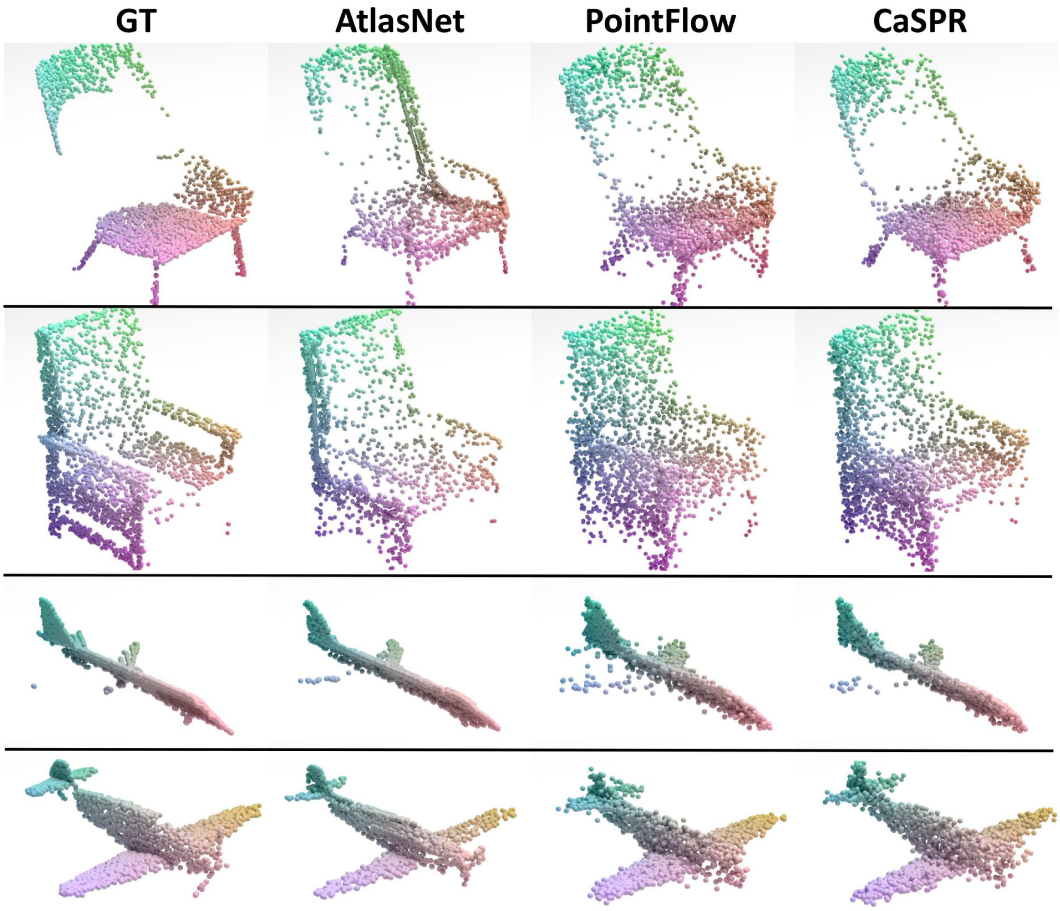

Figure E1: Reconstruction performance of the *CaSPR-Atlas* and *PointFlow* baselines compared to the full CaSPR model. Each row shows a frame from a different 10-step rigid motion sequence.

The *PointFlow* baseline operates on **single already-canonical partial point cloud frames**, while CaSPR and *CaSPR-Atlas* take in raw world-space sequences of partial point clouds. To reconstruct a sequence, *PointFlow* can easily reconstruct the observed (canonical) frames by simply autoencoding each frame independently. However, to allow reconstruction of intermediate unobserved steps, we must use linear interpolation in the shape feature space from surrounding observed frames, as described in the main paper.

Median reconstruction errors are presented in Tab. 2 of the main paper. Mean and standard deviation are shown here in Tab. E3. Generally, the CaSPR variants have a higher standard deviation than PointFlow. This is likely because CaSPR methods must canonicalize the input in addition to reconstructing it, so any errors in this first step may compound in the reconstruction causing some occasional high errors. A qualitative comparison is shown in Fig. E1. The *CaSPR-Atlas* baseline has perhaps deceivingly poor EMD errors. As discussed in the main paper, the patch-based approach has difficulty reconstructing the true point distribution of the partial view and may cause some areas to be much more dense or sparse than they should (see chairs in Fig. E1). Because EMD requires a bijection, these overly dense areas are paired with distant points causing large errors. However, qualitative and CD results suggest the approach has some advantages: the reconstructed point cloud tends to be less noisy and capture local detail better than its CNF-based counterparts.

Additional results of the full CaSPR model reconstructing 10-frame input sequences of rigid, partial point clouds are shown in Fig. E2. Please see the caption for details. Note that the shown T-NOCS predictions are using TPointNet++ trained jointly within the full CaSPR model rather than individually as in the “Canonicalization" experiments.

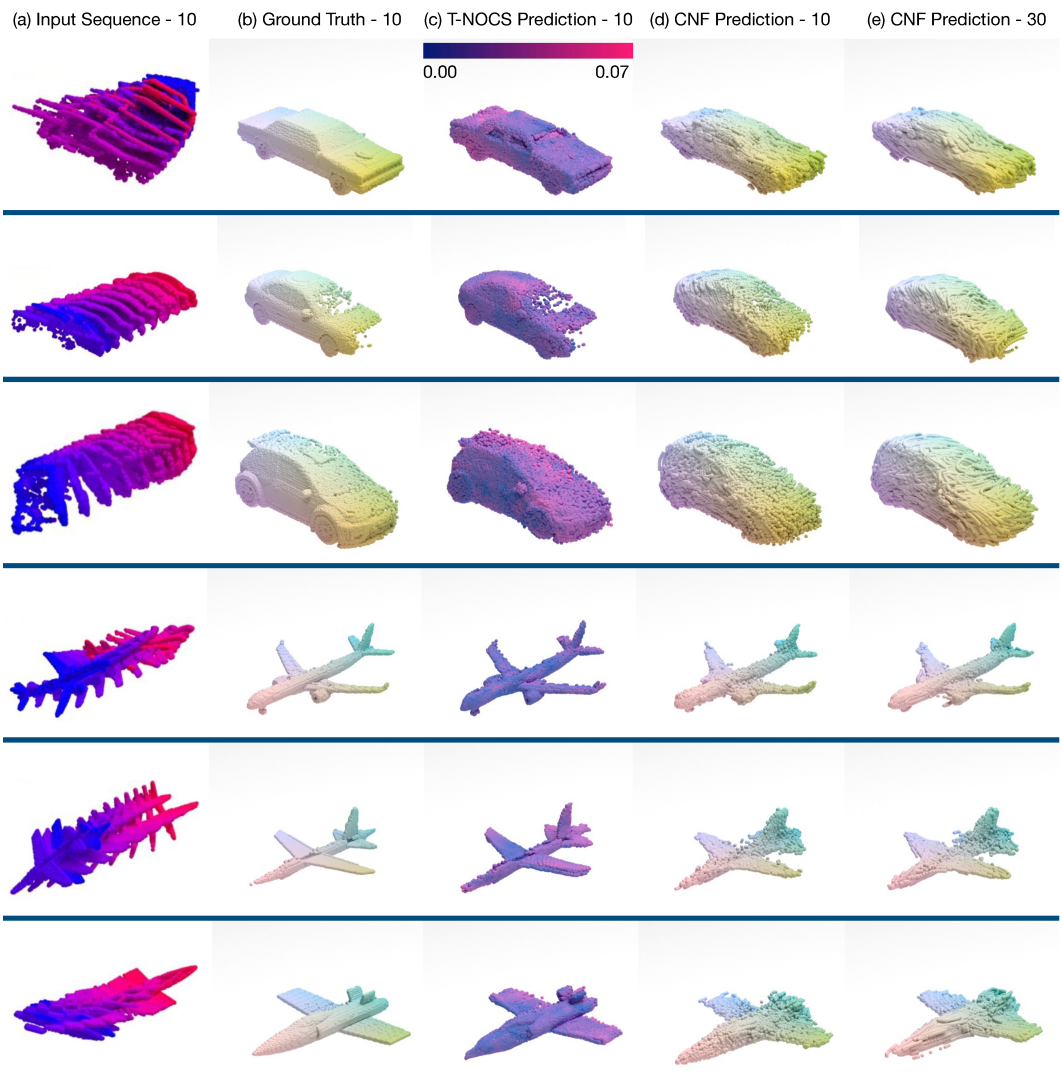

Figure E2: Canonicalization, aggregation, and dense reconstruction of rigid motion sequences by the full CaSPR model. Each sequence shows (a) the 10 observed raw partial point cloud frames given as input to CaSPR, (b) the GT partial reconstruction based on the observed frames, (c) the partial reconstruction achieved by aggregating T-NOCS predictions from TPointNet++ with color mapped to spatial error, (d) the aggregated prediction after reconstructing the 10 observed frames with the CNF, and (e) the aggregated prediction when interpolating 30 frames using the CNF.

Rigid Spatiotemporal Interpolation Additional results of the full CaSPR architecture reconstructing a sparse, partial input sequence are shown in Fig. E3. In each sequence, the model is given 3 frames with 512 points (with GT canonical point cloud shown as *Sparse GT*) and reconstructs any number of densely sampled steps (10 are shown as *Dense Sampling*, each with 2048 points).

Rigid Pose Estimation We solve for object pose in a post-processing step that leverages the world-canonical correspondences given by the output of TPointNet++. We use the full TPointNet++ architecture trained as in the "Canonicalization" evaluation above. For each frame independently, we run RANSAC [22] using 4 points to perform the fitting and with an inlier threshold of 0.015.

Table E4: Pose estimation performance showing mean and (standard deviation). Supplements Tab. 3 in the main paper.

Method	Category	Trans Err	Rot Err(°)	Point Err
RPM-Net	Cars	**0.0071** (0.0102)	**2.1677** (10.0952)	**0.0087** (0.0146)
CaSPR		0.0116 (0.0245)	4.9597 (22.8311)	0.0203 (0.0645)
RPM-Net	Chairs	**0.0029** (0.0031)	**0.6212** (3.3530)	**0.0042** (0.0078)
CaSPR		0.0094 (0.0127)	3.0264 (9.5897)	0.0152 (0.0367)
RPM-Net	Airplanes	**0.0050** (0.0076)	**2.2703** (16.2945)	**0.0070** (0.0190)
CaSPR		0.0083 (0.0144)	3.6740 (16.9152)	0.0144 (0.0456)

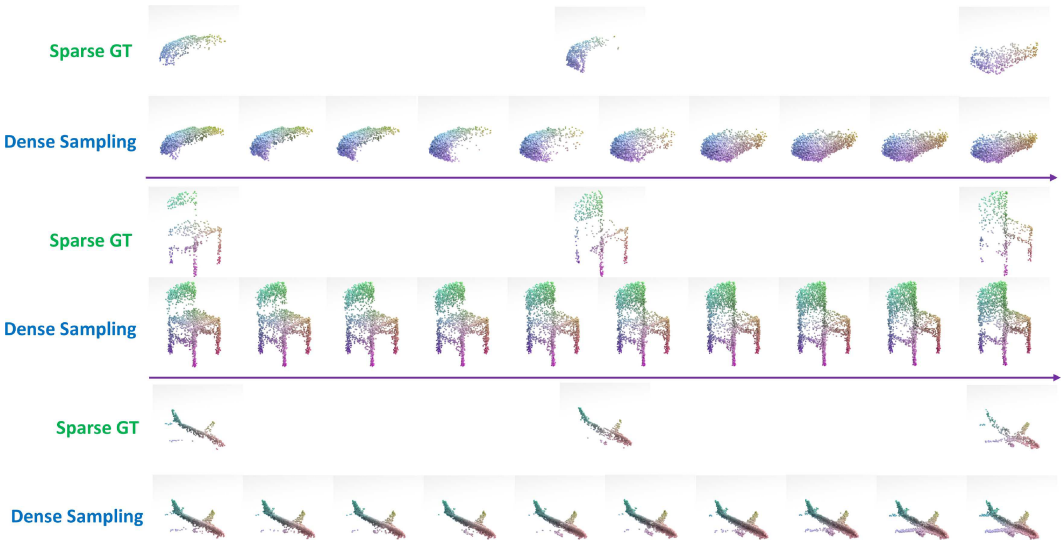

Figure E3: Examples of spatiotemporal interpolation to reconstruct sparse, partial input sequences. The sparse GT canonical point cloud for each sequence is shown in the top row; the dense CaSPR reconstruction using the CNF is shown in the bottom row.

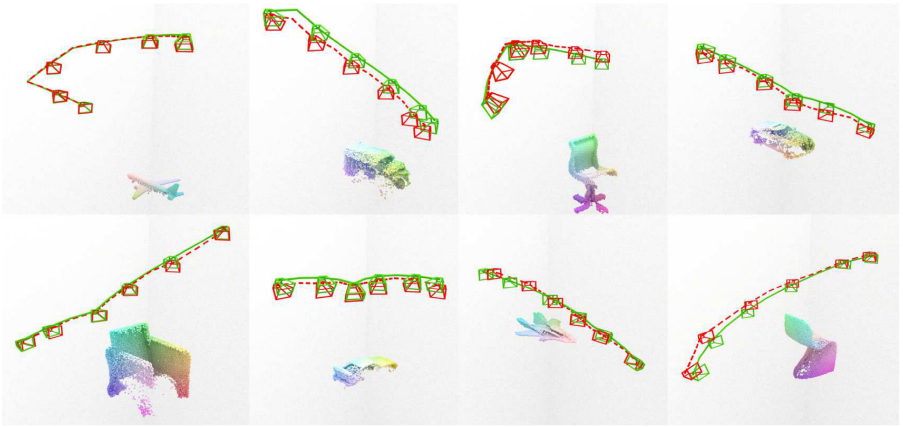

Figure E4: Additional camera pose estimation results. Ground truth trajectories are shown in solid green and the CaSPR prediction in dashed red.

We compare our approach to a recent method for robust rigid registration called RPM-Net [85]. This is an algorithm specially designed for pairwise point cloud registration that iteratively estimates the transformation parameters of possibly-partial point clouds by estimating soft correspondences in the inferred feature space. Because this method can only operate on pairs of point clouds, during training we give it the raw partial point cloud (1024 points) along with the corresponding GT canonical point cloud (1024 points with permuted ordering) as input. This contrasts with TPointNet++ that only receives the raw partial point cloud at each step, and must *predict* the canonical point cloud to establish correspondences. At test-time, we instead use 2048 points for the input point clouds, and the raw points and GT canonical points are randomly sampled so they are not in perfect correspondence.

Median errors appear in Tab. 3 of the main paper, but mean and standard deviation results are shown here in Tab. E4. Though TPointNet++ does not outperform RPM-Net on any shape category, the minimal gap in performance is impressive considering that TPointNet++ has to solve a much harder task (*pose estimation*) than RPM-Net, which receives both the world and canonical point clouds as input and iteratively solves the simpler *pairwise registration* task. As seen in Fig. E5, the qualitative difference between the two methods is nearly imperceivable. Additional qualitative camera pose estimation results from CaSPR are shown in Fig. E4.

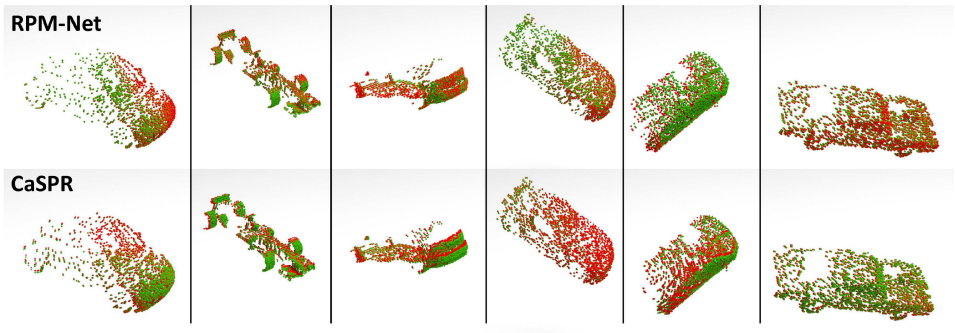

Figure E5: Rigid pose estimation comparison. Each column shows a frame from a different partial point cloud sequence. Predicted object pose (red points) is shown compared to the GT depth point cloud (green points) for each method. Both methods are very accurate.

Non-Rigid Reconstruction and Temporal Correspondences We compare CaSPR to Occupancy Flow (OFlow) [49] on the task of reconstructing Warping Cars sequences and estimating correspondences over time. Because OFlow uses an implicit occupancy shape representation, this dataset contains complete shapes with a clearly defined inside and out. The OFlow baseline uses the point cloud completion version of the model, which leverages a PointNet-ResNet architecture for both the spatial and temporal encoders. OFlow is trained with the reconstruction loss only (*i.e.* it does not explicitly use a correspondence loss).

Table E5: Deformable reconstruction and correspondences mean and (standard deviation). Supplements Tab. 4 in the main paper.

	Reconstruction		*Correspondences*	
Method	**CD**	**EMD**	**Dist** t_1	**Dist** t_{10}
OFlow	1.764 (0.913)	24.247 (14.74)	**0.011** (0.003)	**0.032** (0.007)
CaSPR	**0.992** (0.256)	**12.864** (5.856)	0.014 (0.002)	0.037 (0.009)

Both methods are trained on sequences of 10 frames with 512 points, and tested on sequences of 10 frames with 2048 points. Due to restrictions of the OFlow encoder, the points at each frame in the input sequence are in correspondence over time (note that this is **not** a requirement for CaSPR, which can accurately estimate temporal correspondence even if this is not the case as in most real-world applications) and we must use the same number of timesteps at training and test time. To reconstruct a sequence with OFlow, we reconstruct the mesh at the first time step based on the occupancy network predictions, then randomly sample 2048 points on this mesh and advect them forward in time with the predicted flow field. For CaSPR, we advect the latent feature forward in time to each desired timestep as per the usual, then reconstruct each frame with the CNF using the *same* Gaussian samples at each step to achieve temporal continuity.

Median reconstruction and correspondence errors are reported in Tab. 4 of the main paper. Here we show the mean and standard deviations in Tab. E5. Reconstruction errors are measured at all 10 observed timesteps by randomly sampling 2048 ground truth points, while correspondence errors are measured at only the first and last steps using the procedure detailed in the main paper. Additional qualitative results are shown in Fig. E6.

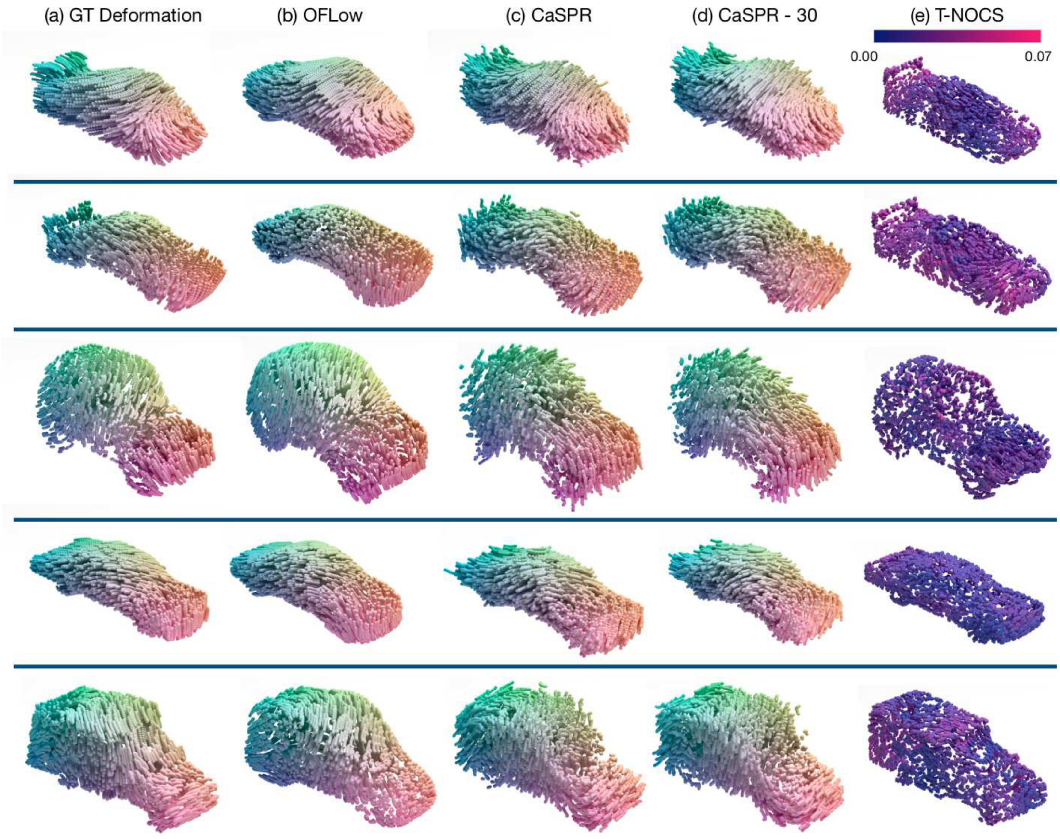

Figure E6: Reconstruction results on Warping Cars data. Each sequence is 10 steps in length and we show point trajectories over time for (a) the ground truth input sequence, (b) the reconstruction from Occupancy Flow, (c) the reconstruction at the 10 observed steps with CaSPR, (d) 30 interpolated steps with CaSPR, and (e) the T-NOCS prediction from TPointNet++.