

# STREAM: A Universal State-Space Model for Sparse Geometric Data

Mark Schöne<sup>1\*</sup>   Yash Bhisikar<sup>2\*</sup>   Karan Bania<sup>2\*</sup>   Khaleelulla Khan Nazeer<sup>1</sup>  
 Christian Mayr<sup>1,3,4</sup>   Anand Subramoney<sup>3</sup>  
 David Kappel<sup>4</sup>

<sup>1</sup>TUD Dresden University of Technology   <sup>2</sup>BITS Pilani  
<sup>3</sup>Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI)  
<sup>4</sup>Centre for Tactile Internet (CeTI) with Human-in-the-Loop  
<sup>5</sup>Royal Holloway, University of London   <sup>6</sup>Bielefeld University

{mark.schoene, khaleelulla.khan, christian.mayr}@tu-dresden.de  
 {f20210483, f20212582}@goa.bits-pilani.ac.in  
 anand.subramoney@rhul.ac.uk  
 david.kappel@uni-bielefeld.de

## Abstract

Handling sparse and unstructured geometric data, such as point clouds or event-based vision, is a pressing challenge in the field of machine vision. Recently, sequence models such as Transformers and state-space models entered the domain of geometric data. These methods require specialized preprocessing to create a sequential view of a set of points. Furthermore, prior works involving sequence models iterate geometric data with either uniform or learned step sizes, implicitly relying on the model to infer the underlying geometric structure. In this work, we propose to encode geometric structure explicitly into the parameterization of a state-space model. State-space models are based on linear dynamics governed by a one-dimensional variable such as time or a spatial coordinate. We exploit this dynamic variable to inject relative differences of coordinates into the step size of the state-space model. The resulting geometric operation computes interactions between all pairs of  $N$  points in  $\mathcal{O}(N)$  steps. Our model deploys the Mamba selective state-space model with a modified CUDA kernel to efficiently map sparse geometric data to modern hardware. The resulting sequence model, which we call *STREAM*, achieves competitive results on a range of benchmarks from point-cloud classification to event-based vision and audio classification. *STREAM* demonstrates a powerful inductive bias for sparse geometric data by improving the PointMamba baseline when trained from scratch on

the *ModelNet40* and *ScanObjectNN* point cloud analysis datasets. It further achieves, for the first time, 100% test accuracy on all 11 classes of the *DVS128 Gestures* dataset.

## 1. Introduction

Computer vision computes relationships between data points in spatial coordinates  $X, Y$  (in  $\mathbb{R}^2$ ) or  $X, Y, Z$  (in  $\mathbb{R}^3$ ), or spatio-temporal coordinates ( $\mathbb{R}^3$  or  $\mathbb{R}^4$ ), where one of the dimensions denotes time  $t$ . Convolutional neural networks based on structured, uniformly spaced and local linear operations successfully address this problem for classical camera recordings such as images or videos, which are themselves structured and discrete. Many modalities of recent interest, however, are neither structured nor uniformly spaced. Sensors such as Light Detection and Ranging (LiDAR) or event-based cameras [21, 28] sample signals based on sparse processes, resulting in sparse geometric data with irregularly spaced coordinates. Point cloud analysis was the central research field for sparse geometric data over the past decade [13]. While early works followed voxel-based approaches [25, 50], point-based methods dominate the research landscape today [4, 5, 46, 48]. Meanwhile, event-based cameras [21, 28] raised considerable interest in the computer vision community. Although these cameras record sparse geometric data, most works collapse the sparse stream of events into frames [28, 56, 57], potentially losing unique properties of these cameras such as low latency and high dynamic range. The structural similarity be-

\*These authors contributed equally to this work.

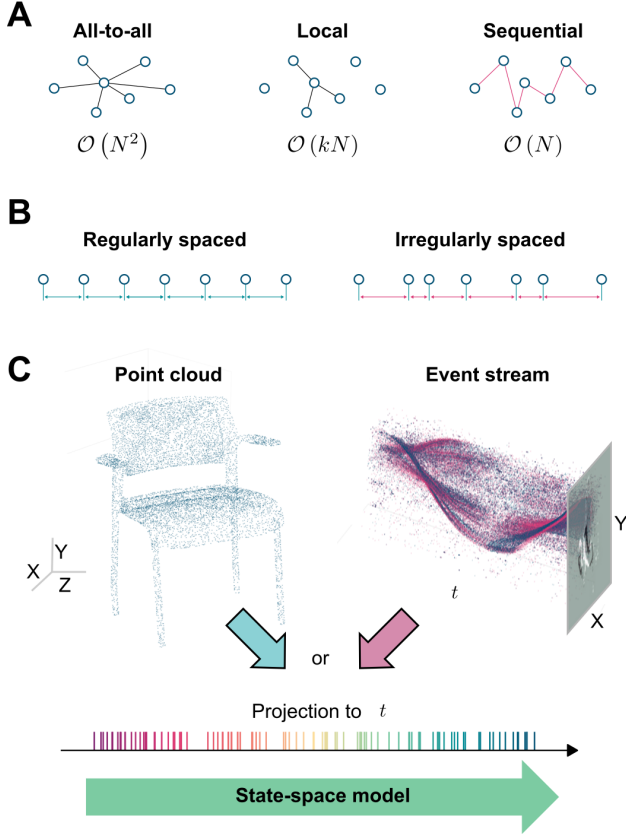


Figure 1. A unified view on point cloud and event stream modeling. **A.** Adjacency structure of point based methods **B.** The coordinates of sparse geometric data are *irregularly spaced* **C.** Point clouds and event streams are ordered by spatial and temporal axes, respectively, before the state-space model.

tween point clouds and event streams encourages methodological transfer, especially from the domain of point cloud analysis to event-based vision [35, 45]. Our work demonstrates the reverse: The inherently temporal state-space model formulation presented in sec. 3.2 improves event-based vision on the DVS128-Gesture dataset [1] to 100 %, and at the same time provides a valuable inductive bias for geometric data such as point clouds as demonstrated on the ModelNet40 and ScanObjectNN datasets [43, 51].

We show that sparse geometric data with irregularly spaced coordinates as shown in fig. 1B can be naturally integrated in the state-space model formalism. In contrast to prior sequence models such as PointMamba [20] or EventMamba [36], we *explicitly* encode geometric structure into the state-space model parameterization by injecting relative differences of coordinates as step sizes.

**Contributions.** This paper makes the following main innovations over the state-of-the-art:

- We propose a unified framework for modeling sparse geometric data with state-space models with irregularly

spaced step sizes. The resulting model, STREAM, handles sparse geometric data such as point clouds *and* event-based vision.

- Our state-space model formulation demonstrates a valuable inductive bias for geometric structure by improving the PointMamba [20] baseline by up to 2.84 % when trained from scratch on the ScanObjectNN dataset [43].
- We show compelling results for event-based vision, processed as a stream of events with our purely recurrent neural network without the usage of frames or spatial convolutions. For the first time, we demonstrate 100 % classification accuracy on all 11 classes of the DVS128-Gestures dataset [1].
- We provide an efficient CUDA implementation for integrating irregular step sizes based on the selective-scan implementation of [9].

## 2. Related Work

### 2.1. Point Cloud Analysis

Early point-based methods such as PointNets [4, 31] directly pass the point cloud through a Multi-layer Perceptron (MLP) and introduce a permutation invariance of the set of points via pooling operations, *implicitly* integrating spatial information. In contrast, methods for *explicitly* integrating spatial information parameterize linear operators such as convolutions based on relative differences between point coordinates. The convolutional neural networks based on irregularly spaced operators presented in [17, 46, 48, 49] outperform PointNets on a range of point cloud analysis tasks. These works parameterize the convolution kernel with MLPs evaluated at the relative differences between point coordinates. To break the  $\mathcal{O}(N^2)$  complexity of computing pairwise interactions for all  $N$  points, locality constraints are added to scale to larger point sets (see fig. 1A). As shown in fig. 2A, our method also explicitly integrates spatial information in this sense, but with a kernel parameterized by a state-space model (SSM) instead of a MLP (see equation (3)). This parameterization enables the complete computation to be performed in  $\mathcal{O}(N)$  steps.

More recently, sequence models demonstrated favorable results over convolutional methods [5, 20, 26, 52]. The point cloud is flattened into a sequence and then processed by transformer or state-space model based backbones. This formulation inherits many strong scaling properties of sequence models including masked pre-training [52] or generative pre-training [5] objectives. Transformers, however, suffer from  $\mathcal{O}(N^2)$  complexity in the number of points  $N$ , limiting their application to larger point clouds. State-space models, in contrast, have sequential  $\mathcal{O}(N)$  complexity for inference and  $\mathcal{O}(N \log N)$  complexity for parallel training [10, 11]. At the same time, SSMs can learn long distant relationships between inputs including convolutional

operators with rational transfer functions [10, 30]. The recently introduced Mamba state-space model [9] sparked widespread discourse, and was quickly adapted in many domains, including point cloud analysis. Works like PointMamba [20] or Point Cloud Mamba (PCM) [55] deploy Mamba in established point cloud frameworks such as the masked autoencoder developed in [26]. By relying on the default Mamba architecture, spatial information is integrated *implicitly* in these works similar to PointNets. In contrast, we show in sec. 3.2 that spatial information can be integrated *explicitly* via the SSM parameterization. This formulation allows us to remove specialized preprocessing methods to represent point clouds as sequences [20, 55], and simply order by the spatial coordinates  $X$ ,  $Y$ , or  $Z$ .

## 2.2. Event-based Vision

Most state-of-the-art event-based vision methods construct frames from the asynchronous event-stream by binning events in regularly spaced time intervals [6, 28, 42, 56, 57]. Few methods operate on the sparse unstructured event streams directly. AEGNN [37] computes spatio-temporal features based on subsampled event streams with graph neural networks. Their method admits an event-based inference method where only nodes that correspond to incoming events are updated asynchronously. EventMamba [36] is tightly related to the point cloud analysis models discussed in sec. 2.1. They process subsampled raw event streams similar to [37, 39] with a combination of a point feature extractor and Mamba. Its high similarity with PointMamba [20] places EventMamba in the category of methods *implicitly* integrating spatio-temporal information. Conceptually closer to our model are filtering methods that process the stream of events recursively while *explicitly integrating spatio-temporal information*. Early methods like HOTS [18] create a spatial representation of the  $(X, Y)$ -plane of an event-based camera by integrating exact timestamps with exponentially moving averages. Similar to our method, EventNet [39] recursively iterates the event stream. In contrast to our state-space model, their recurrent network parameterization does not inherit stability guarantees from the theory of linear systems and lacks an efficient parallelization along the time dimension during training, limiting the scalability to longer streams. Event-SSM [38] proposed a discretization method for simplified state-space layers (S5) [40] to operate directly on asynchronous event streams. Their model excels at spiking audio classification, but falls behind the state-of-the-art in event-based vision. Concurrently to our work, S7 [41] explores improvements of the S5 architecture and Event-SSM through a more stable parameterization and input dependent state-space model parameters. In contrast, STREAM can improve sequence-based point cloud analysis (see sec. 4.1), and therefore drive the convergence between event-based vision and point cloud

analysis forward.

Related from the perspective of discretizing continuous kernels is the TENNs framework [27]. They construct convolutions over finite time horizons with orthogonal polynomials that can be discretized on irregularly spaced timestamps. In practice, they choose regularly spaced frames, however, and show compelling results on event-based vision benchmarks. In comparison, STREAM parameterizes infinite time horizons via the state-space model.

## 3. STREAM

In this section, we present our method for efficiently modeling sparse sets of  $N$  coordinates  $\{\mathbf{x}_i\}_{i=0}^N$ , where  $\mathbf{x}_i = (x_i^{(1)}, \dots, x_i^{(d)}) \in \mathbb{R}^d$ . This general formulation includes data in 3D space, with  $\mathbf{x} = (X, Y, Z)$ , as well as the streams recorded by event-based cameras. In the latter case one of the dimensions (e.g.  $x^{(1)}$ ) is interpreted as event time while others ( $x^{(2)}, x^{(3)}$ ) denote spatial locations (see fig. 1C). Furthermore, higher dimensions ( $d > 3$ ) can be used to denote data that includes other dimensions, such as color intensity, etc. STREAM is illustrated in fig. 2A-C.

**Notation.** We denote tensor-valued variables and functions such as multi-dimensional coordinates  $\mathbf{x} \in \mathbb{R}^d$  or matrices  $\mathbf{A} \in \mathbb{R}^{m \times m}$  in bold characters. Scalar variables such as the coordinates  $x^{(1)}, \dots, x^{(d)}$  of a tensor  $\mathbf{x}$  or the entries  $A_{ij}$  of a matrix  $\mathbf{A}$  are denoted in regular characters.

### 3.1. Problem Statement

Consider a set  $\{\mathbf{x}_i\}_{i=0}^N$  of points sparsely distributed in space. We assign a multi-dimensional representation vector  $\mathbf{u}_i$  to each coordinate  $\mathbf{x}_i$  as common practice in machine learning. Such input signals can be formulated mathematically as sums of Dirac delta pulses evaluated at the coordinates  $\{\mathbf{x}_i\}_{i=0}^N$

$$\mathbf{u}(\mathbf{x}) = \sum_{i=0}^N \delta(\mathbf{x} - \mathbf{x}_i) \mathbf{u}_i, \quad (1)$$

where  $\mathbf{u}_i \in \mathbb{R}^n$  is the representation at  $\mathbf{x}_i$ . To reason about spatial relationships between these pulses, we define an interaction kernel  $\Phi(\mathbf{x}, \mathbf{x}')$  that models pairwise interactions. A complete view of all possible interactions with the point  $\mathbf{x}_k$  is given by

$$\mathbf{y}(\mathbf{x}_k) = \int \Phi(\mathbf{x}_k, \mathbf{x}') \mathbf{u}(\mathbf{x}') d\mathbf{x}' \quad (2)$$

$$= \sum_{i=0}^N \Phi(\mathbf{x}_k, \mathbf{x}_i) \mathbf{u}_i. \quad (3)$$

This formulation contains the familiar convolution operator as a special case when  $\Phi(\mathbf{x}, \mathbf{x}') \equiv \Phi(\mathbf{x} - \mathbf{x}')$ . Equation

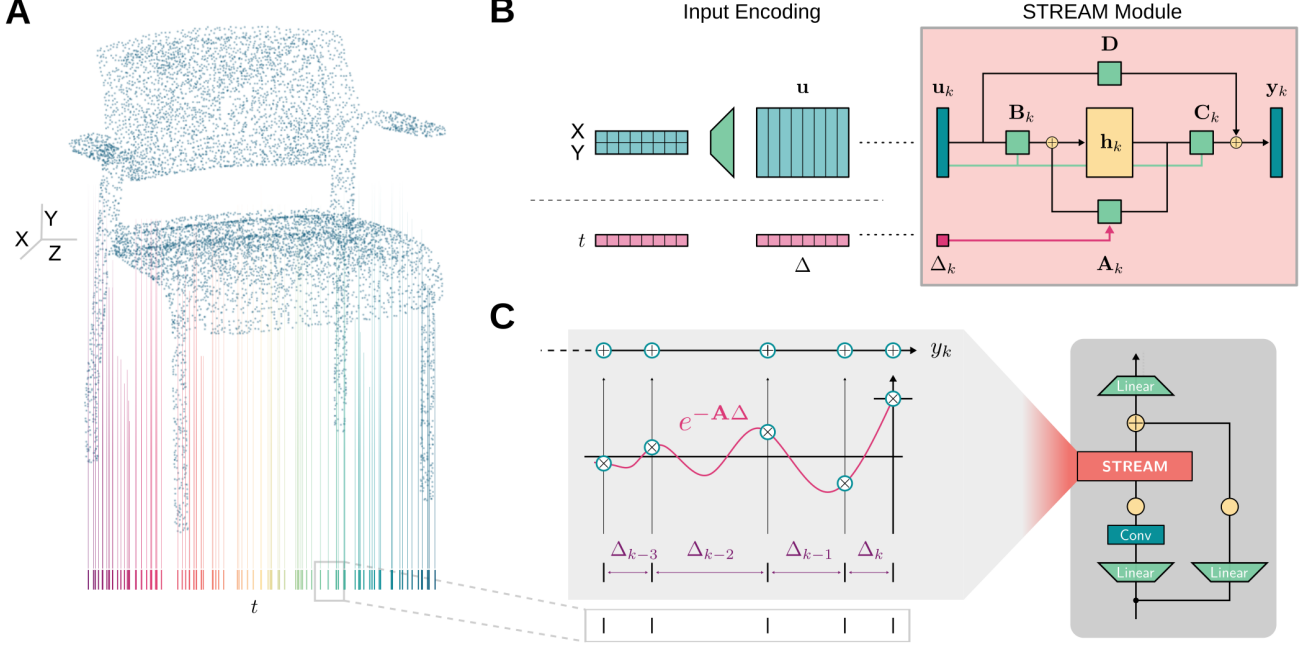


Figure 2. STREAM encodes geometric information into the SSM parameters. **A.** Point cloud input. **B.** The STREAM module converts relative differences in coordinates into the  $\Delta_k$  scale of the SSM. **C.** A STREAM module integrates pairwise spatial relationships based on an exponentially oscillating kernel.

(3) highlights that the computation of  $\mathbf{y}(\mathbf{x}_i)$  at the coordinates  $\{\mathbf{x}_i\}_{i=0}^N$  requires evaluating  $\Phi(\mathbf{x}_i, \mathbf{x}_j)$  only at pairs of these coordinates. In the general case, this operation requires  $\mathcal{O}(N^2)$  steps to compute. As shown in fig. 1A, this complexity can be reduced to  $\mathcal{O}(kN)$  by focusing on local neighborhoods of  $k$  points, a technique broadly used in the literature of point convolutions [46, 48]. In the following sections, we present a method for recursively integrating  $N$  sparsely sampled points in  $\mathcal{O}(N)$  time without restricting to local neighborhoods.

### 3.2. Integrating Sparse Coordinates with State-space Models

State-space models (SSMs) have demonstrated outstanding performance on long-range dependency modeling tasks, while remaining efficient for training and inference [9, 11]. SSMs project the input through a linear recursive kernel to states  $\mathbf{h}(t)$ . The independent variable  $t$  is usually interpreted as, and aligned with, time in the input signal domain. The linear nature of SSMs allows the exact integration of sparse sets of points with irregularly spaced coordinates, as we will show below. A linear time-varying SSM acting on a scalar input function  $u(t)$  and producing a scalar output  $y(t)$  is defined through

$$\dot{\mathbf{h}}(t) = \mathbf{A}(t) \mathbf{h}(t) + \mathbf{B}(t) u(t) \quad (4)$$

$$y(t) = \mathbf{C}(t) \mathbf{h}(t), \quad (5)$$

with  $u(t), y(t) \in \mathbb{R}$ , states  $\mathbf{h}(t) \in \mathbb{R}^m$  and parameters  $\mathbf{A}(t) \in \mathbb{R}^{m \times m}$ ,  $\mathbf{B}(t) \in \mathbb{R}^{m \times 1}$ ,  $\mathbf{C}(t) \in \mathbb{R}^{1 \times m}$ . For the sake of simplicity, we assume that the parameters are represented by piecewise constant functions, i.e.  $\mathbf{A}_i = \mathbf{A}(t_i)$ ,  $\mathbf{B}_i = \mathbf{B}(t_i)$ ,  $\mathbf{C}_i = \mathbf{C}(t_i)$  are constant on the intervals  $(t_{i-1}, t_i]$ . We show in appendix 6 that solving the linear system in equations (4) and (5) for the input in equation (1) yields the kernel function

$$\Phi(t_k, t_i) = \mathbf{C}_k \left( \prod_{j=i+1}^k \exp(\mathbf{A}_j \Delta_j) \right) \mathbf{B}_i, \quad (6)$$

where  $\Delta_j = t_j - t_{j-1}$ . Importantly, the output  $y(t)$  of the state-space model at  $t_0, \dots, t_N$  can be computed with the recursive formula

$$\mathbf{h}_k = \mathbf{h}(t_k) = e^{\mathbf{A}_k \Delta_k} \mathbf{h}_{k-1} + \mathbf{B}_k u_k \quad (7)$$

$$y_k = y(t_k) = \mathbf{C}_k \mathbf{h}_k. \quad (8)$$

Equations (6) and (7) highlight how relative differences  $\Delta_j = t_j - t_{j-1}$  in the coordinate  $t$  explicitly parameterize our pairwise interaction kernel  $\Phi$ . An example of the interaction defined by equation (6) can be found in Fig. 2C. For complex  $\mathbf{A} \in \mathbb{C}$ , the exponentially oscillating kernel  $e^{\mathbf{A} \Delta}$  integrates the history of signals  $u_i$  occurring in irregularly spaced intervals  $\Delta_i$  with  $i < k$ . With this parameterization, our method differs from other Mamba based works, where



$\Delta$  is a function of  $u_i$  that doesn't explicitly take the dynamics of  $t$  into account. A complete derivation can be found in appendix 6.

The recursive formulation in equation (7) is a linear recurrent neural network, whose state-to-state transition matrix  $e^{\mathbf{A}_k \Delta_k}$  is parameterized by the differences  $\Delta_k = t_k - t_{k-1}$  in the irregularly spaced coordinates  $t_i$ . As such, our method positions itself among the methods discussed in sec. 2.1 that *explicitly integrate spatial information* through the parameterization of a linear operator with the coordinates.

The output sequence  $y(t_0), \dots, y(t_N)$  can be computed sequentially in  $\mathcal{O}(N)$  time. This allows *efficient inference on streams* of pulses recorded by sensors such as event-based cameras or LiDAR. At the same time, linear recursive equations admit an *efficient parallelization* via the `Scan` primitive available in CUDA [3]. On sufficiently many processors, the output sequence can be computed in  $\mathcal{O}(\log N)$  time, which allows massive parallelization on very long sequences of up to 1 million inputs [9]. We show in appendix 7 how our formulation maps to the scan primitive.

### 3.3. Selective state-space models and STREAM

We develop our method based on the selective state-space model also known as Mamba [9], which was designed for regularly spaced modalities, and language modeling in particular. In accordance with this purpose, the integration time steps  $\Delta_i$  and the matrices  $\mathbf{B}_i, \mathbf{C}_i$  are functions of the input signal  $u_i$ , and do not carry explicit information about spatial relationships. Formally, Mamba is parameterized by

$$\mathbf{A}_i \equiv \mathbf{A} \quad \forall i \quad \Delta_i = \Psi(\text{Linear}(u_i)) \quad (9)$$

$$\mathbf{B}_i = \Delta_i \text{Linear}(u_i) \quad \mathbf{C}_i = \text{Linear}(u_i), \quad (10)$$

where  $\mathbf{A}$  is a learned diagonal matrix, and  $\Psi(u) = \ln(1 + e^u)$  is the softplus function. This single-input  $u_i$  single-output  $y_i$  SSM is then repeated  $n$  times and wrapped by linear transformations to create a multi-input  $\mathbf{u}_i$  multi-output  $\mathbf{y}_i$  model as shown in fig. 2B. Note that the total state size is  $nm$  in the multi-input multi-output case.

We propose to *explicitly* represent the irregularly spaced pulse timings  $t_i$  in the Mamba parameterization according to equation (7). Therefore, we explicitly set  $\Delta_i$  to  $(t_i - t_{i-1}) \Psi(\delta)$ , where  $\delta$  is a trainable time scale parameter, in contrast to equation (9). We further decouple the coefficient of  $\mathbf{B}_i$  in equation (10) from the time differences  $\Delta_i$  to avoid zero coefficients in case of overlapping pulses  $t_i - t_{i-1} = 0$ . With these adjustments, our model is defined by

$$\mathbf{A}_i \equiv \mathbf{A} \quad \forall i \quad \Delta_i = (t_i - t_{i-1}) \Psi(\delta) \quad (11)$$

$$\mathbf{B}_i = \Gamma_i \text{Linear}(u_i) \quad \mathbf{C}_i = \text{Linear}(u_i) \quad (12)$$

$$\Gamma_i = \Psi(\text{Linear}(u_i)), \quad (13)$$

where  $\delta \in \mathbb{R}^n$  is a learnable parameter. We call our resulting state-space model parameterization STREAM for Spatio-Temporal Recursive Encoding of Asynchronous Modalities.

### 3.4. Modeling Point Clouds with STREAM

The explicit coordinate  $t$  in the STREAM module is one-dimensional and strictly ordered. For the case of event streams, this ordering corresponds to temporal order from past to future. Point clouds fit into this framework by ordering the set of points w.r.t. one of the spatial coordinates from left to right. As shown in fig. 2A,B, we select one of the coordinates to be integrated explicitly. Inserting the  $X$  coordinate for  $t$  in equation (11) without loss of generality and setting  $\mathbf{u}_k = \Theta(X_k, Y_k, Z_k)$ , where  $\Theta: \mathbb{R}^3 \rightarrow \mathbb{R}^n$  is a point wise encoder, yields our STREAM formulation for point cloud analysis. To achieve state-of-the-art results, we sort the point set by all three spatial coordinates respectively and feed the concatenation of the three resulting sequences to STREAM (see sec. 4.1.1, fig. 3).

## 4. Experiments

We demonstrate empirically that the explicitly parameterization of the state-space model with coordinate differences  $\Delta_i$  presented in sections 3.2 and 3.3 is a strong abstraction that exploits the sparse geometry of both point clouds and event-based vision. When trained from scratch, our method improves the PointMamba [20] baseline by up to 2.8 % on the ScanObjectNN dataset [43]. At the same time, our method sets a new state-of-the-art for event-based gesture recognition on the DVS128-Gestures dataset [1].

### 4.1. STREAM for Point Cloud Analysis

#### 4.1.1 Point Cloud Model

Our goal is to evaluate the explicit SSM parameterization defined in sec. 3.3. Therefore, we align our architecture close to PointMamba [20], which uses the default Mamba module [9]. As visualized in fig. 3A, we sample  $N$  points and sort this point set by the  $X$ ,  $Y$ , and  $Z$  coordinates respectively. The vector representations extracted from the point coordinates individually are scaled and shifted by learned parameters for the three sorted sequences separately to indicate the sorting dimension. The resulting sequence of  $3N$  points is concatenated along the sequence dimension to form the input to our model. This preprocessing simplifies the ordering of the point set by avoiding the two space-filling Hilbert-curves applied by PointMamba at the cost of increasing the input sequence length from  $2N$  to  $3N$ .

Similar to [5, 20, 26], our model is composed of two stages. The input sequence is grouped into sets of 32 points that are independently processed by a STREAM module to represent local information. A backbone of 12 STREAM

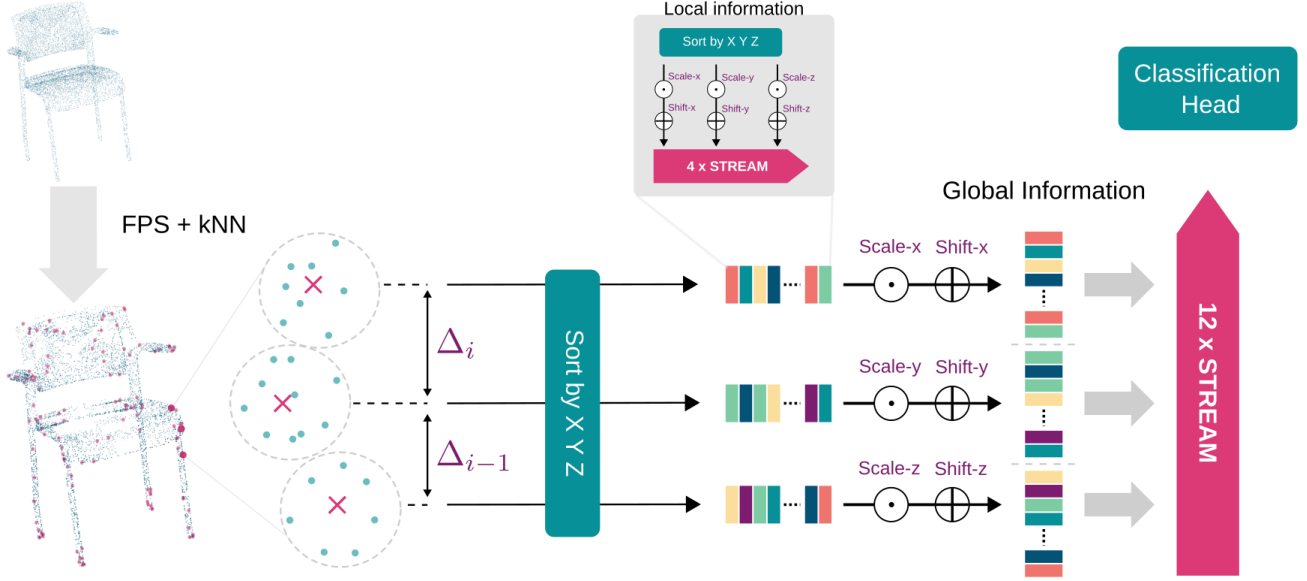


Figure 3. Point cloud processing pipeline inspired by PointMamba [20]. FPS: Farthest point sampling, kNN: k-Nearest Neighbors.

layers operates on the features extracted from these groups to agglomerate global information.

#### 4.1.2 Point Cloud Results

We evaluate our method on two standardized point cloud classification benchmarks. The ModelNet40 dataset [51] contains 12 311 clean point clouds sampled from 3D CAD objects. A more realistic dataset of physically scanned indoor scenes is given by ScanObjectNN [43]. ScanObjectNN contains about 15 000 scans from 2902 unique objects with varying difficulty. In both cases, we train our method with the same hyperparameters as PointMamba [20].

We report the best overall classification accuracy, i.e. the average over all samples, obtained from 5 different random seeds. Tab. 1 shows that replacing the Mamba module in PointMamba with a STREAM module improves the performance of models trained from scratch across all instances on ScanObjectNN. In particular, the hardest instance PB-T50-RS benefits from the exact integration of spatial information and improves PointMamba from 82.48 % to 85.32 % for the best out of 5 random seeds. With a mean and standard deviation of  $(84.4 \pm 0.7) \%$ , this poses a significant improvement.

On ModelNet40, STREAM improves the overall accuracy of PointMamba by 0.3 % from 92.4 % to 92.7 % as reported in tab. 2. Despite the smaller gain compared to ScanObjectNN, the low variance on this dataset indicates a significant improvement on ModelNet40 as well. The mean and standard deviation of our model is  $(92.6 \pm 0.1) \%$ .

Our results show that the inductive bias of explicitly encoding spatial relationships in the SSM parameterization is particularly useful when training models from scratch.

#### 4.2. STREAM for Event Streams

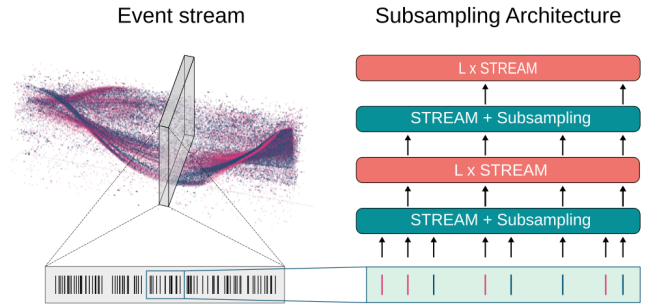


Figure 4. Hierarchical subsampling architectures with two stages to handle long event streams.

##### 4.2.1 Event Stream Model

In contrast to most prior works, our method directly operates on the stream of events as visualized in fig. 4. The stream is encoded as a sequence of tokens  $(t_i, \mathbf{u}_i)$ , where each pixel of the event-based camera is uniquely encoded by a separate token, similar to how text is represented as tokens in modern language modeling [33]. This sequence is directly processed by a STREAM module. To accommodate large streams of up to 131 072 events per sample in GPU memory, the output of the first STREAM module is

Methods	Backbone	Param. (M) ↓	OBJ-BG ↑	OBJ-ONLY ↑	PB-T50-RS ↑
Supervised Learning Only					
PointMLP [24]	-	13.2	85.4	-	85.4
RepSurf-U [34]	-	1.5	-	-	84.3
ADS [15]	-	-	-	-	87.5
Point-MAE [26]	Transformer	22.1	86.75	86.92	80.78
PCM [55]	Mamba	34.2	-	-	88.1
PointMamba [20]	Mamba	12.3	88.30	87.78	82.48
Ours	STREAM	12.3	90.02 (+1.72)	88.64 (+0.86)	85.32 (+2.84)
Training from Pre-training (Single-Modal)					
Point-BERT [52]	Transformer	22.1	87.43	88.12	83.07
MaskPoint [23]	Transformer	22.1	89.30	88.10	84.30
Point-MAE [26]	Transformer	22.1	92.77	91.22	89.04
PointMamba [20]	Mamba	12.3	94.32	92.60	89.31

Table 1. Object classification on ScanObjectNN [43]. We report overall accuracy (OA) in percent and indicate the **improvement** over the PointMamba baseline through using our STREAM module instead of the default Mamba module.

Methods	Param. (M) ↓	OA (%) ↑
Supervised Learning Only		
PointNet [4]	3.5	89.2
PointNet++ [31]	1.5	90.7
PointCNN [19]	0.6	92.2
DGCNN [29]	1.8	92.9
PointNeXt [32]	1.4	92.9
PCT [12]	2.9	93.2
OctFormer [44]	4.0	92.7
PCM-Tiny [55]	6.9	93.1
Point-MAE [26]	22.1	92.3
PointMamba [20]	12.3	92.4
STREAM (ours)	12.3	92.7 (+0.3)
with Self-supervised pre-training		
Point-BERT [52]	22.1	92.7
MaskPoint [23]	22.1	92.6
Point-M2AE [54]	12.8	93.4
Point-MAE [26]	22.1	93.2
PointGPT-S [5]	29.2	93.3
ACT [8]	22.1	93.6
PointMamba [20]	12.3	93.6

Table 2. Classification on ModelNet40. We report overall accuracy (OA) in percent and indicate the **improvement** over the PointMamba baseline through using our STREAM module instead of the default Mamba module.

subsampled by a factor of 8 or 16 depending on the task. This way, every event is processed by the neural network at least once directly, in contrast to previous methods that

Methods	Param. (M) ↓	Acc (%) ↑
Frame-based		
ACE-BET [22]	-	98.9
ExACT [56]	-	98.9
SpikMamba [6]	0.2	99.0
EventMamba [36]	0.3	99.2
TENNs-PLEIADES [27]	0.2	100.0*
Event-based		
Event-SSM [38]	5.0	97.7
S7 [41]	4.1	99.2
STREAM (ours)	0.2 + 1.0 <sup>†</sup>	100.0

Table 3. Comparison of STREAM to the state-of-the-art on the DVS128-Gestures dataset [1]. \*evaluated on 10 out of 11 classes, omitting the "other" class. <sup>†</sup>0.2 M parameters for STREAM, and 1.0 M parameters for encoding the input stream as tokens.

subsampled the raw stream before presenting it to a neural network [37, 39]. To create a hierarchical architecture, the sequence is subsampled a second time after half the number of layers. The state dimension is increased by a factor of 2 upon the second subsampling. Our hierarchical subsampling architecture is visualized in fig. 4. We apply average pooling along the sequence dimension before computing the class labels. To improve generalization, we implement a set of geometric data augmentations and a variant of CutMix [53] that directly mixes event streams [38].

Methods	Param. (M) ↓	Acc (%) ↑
Frame-based		
Bittar and Garner [2]	0.1	71.7
Bittar and Garner [2]	3.9	77.4
Hammouamri et al. [14]	0.7	79.8
Hammouamri et al. [14]	2.5	80.7
Event-based		
Event-SSM [38]	0.1	85.3
Event-SSM [38]	0.6	88.4
S7 [41]	0.6	88.2
STREAM (ours)	0.2	86.3

Table 4. Comparison of STREAM to the state-of-the-art on the Spiking Speech Commands dataset [7].

#### 4.2.2 Event Stream Results

We evaluate STREAM on two popular event-based datasets. The DVS128 Gestures dataset [1] consists of 1342 recordings of 10 distinct classes of hand gestures and an additional class of arbitrary other gestures. While this dataset features only a relatively small number of samples for 11 hand gesture categories recorded from 29 subjects, the total number of events in this dataset adds up to about 390 M events. Our model has an initial model dimension of  $n = 32$  and a state-space dimension of  $m = 32$ , which is expanded by a factor of 2 upon subsampling. We deploy a total of 6 STREAM blocks, which by the notation of fig. 4 corresponds to  $L = 2$ . We train with a batch size of 32 on sequences of 65 536 events such that the total input sequences created by CutMix are up to 131 072 events long, and subsample by a factor of 16. Evaluation is conducted on the full sequences of up to 1.5 M events. Remarkably, our fully event-based model sets a new state-of-the-art on the DVS128-Gestures dataset as reported in tab. 3, improving over both frame-based and event-based references. While [27] reported 100 % accuracy on 10 out of the 11 classes, omitting the ‘others’ class, we for the first time present a model that can reach a maximum accuracy of 100 % on all 11 classes of the dataset.

In addition to event-based vision, we evaluate STREAM on an event-based audio classification task. The Spiking Speech Commands dataset [7] contains more than 100 000 samples converted from the original Speech Commands dataset [47] with a median number of 8100 events per sample. The model dimension is again  $n = 32$  with a smaller state-space dimension of  $m = 4$  compared to the vision model. We deploy 8 STREAM blocks ( $L = 3$  in fig. 4), and set the subsampling factor to 8. With this configuration, we report a classification accuracy of 86.3 % in tab. 4, which settles between the small and large models reported in [38].

#### 4.3. Ablation Study

We compare our STREAM module against the Mamba module [9] in tab. 5. The central difference is the explicit integration of spatio-temporal information in the recurrent operator  $e^{A\Delta_k}$  with  $\Delta_k = t_k - t_{k-1}$ . We additionally experiment with all combinations of applying linear transformations activated by Softplus functions to renormalize the  $\Delta_k$  and  $\Gamma_k$  parameters in equations (11) and (12), respectively. While the fairly small DVS128 Gestures dataset possesses high variance, STREAM consistently improves over Mamba. This effect is stronger expressed on the larger Spiking Speech Commands dataset, which is less affected by training noise due to the larger number of samples. Here, STREAM creates a significant margin to the Mamba baseline.

#### 5. Discussion

We have introduced STREAM, a sequence model for point cloud and event stream data, which achieves competitive results on a range of benchmarks from point-cloud classification to event-based vision. Prior efforts of unifying these modalities transferred point cloud models to event streams, or ignore the spatio-temporal structure of either modality by applying the default Mamba model to a sequential view of the data. In contrast, our model exploits the dynamics of state-space models, a temporal process at first sight, to encode spatial geometric information into the models parameterization. This inductive bias proved valuable as STREAM improves our reference model, PointMamba, when trained from scratch on point cloud datasets such as ModelNet40 and ScanObjectNN. By design, STREAM applies to spatio-temporal modalities such as event-based vision. We operate on the stream of events without collapsing events into frames and without using 2D convolutions. This processing paradigm achieves 100 % classification accuracy on all 11 classes of the event-based DVS128 Gestures dataset, a result that has so far only been achieved on the 10 predetermined classes by [27]. A practically relevant property of STREAM is that it allows asynchronous inference on streams of points or events recorded from LiDAR sensors or event-based cameras. We, therefore, expect STREAM to be well suited for sensor fusion of these modalities. Previous work has shown that SSMs can benefit significantly from self-supervised pre-training on larger datasets [20], which we will explore for our method in future research.

#### Acknowledgments

MS is supported with funds from Bosch-Forschungsförderung im Stifterverband. YB and KB were funded by the German Academic Exchange Service (DAAD) under the funding programme WISE (57698568). DK is funded by the German Federal Ministry of Education and Research



Model	DVS128 Gestures				Spiking Speech Commands			
	$t_k - t_{k-1}$	softplus + $\Delta$	Linear $\Gamma$	Validation Accuracy	$t_k - t_{k-1}$	softplus + $\Delta$	Linear $\Gamma$	Validation Accuracy
Mamba	✗	✓	✓	98.6± 0.7	✗	✓	✓	86.8± 0.1
STREAM (variants)	✓	✗	✗	99.2± 0.3	✓	✗	✗	87.9± 0.1
	✓	✗	✓	98.8± 0.2	✓	✗	✓	87.9± 0.3
	✓	✓	✗	98.8± 0.6	✓	✓	✗	87.8± 0.2
	✓	✓	✓	98.6± 1.2	✓	✓	✓	87.9± 0.3

Table 5. STREAM versus Mamba [9] on the DVS128 Gestures [1] and Spiking Speech Commands [7] datasets. STREAM directly feeds the time coordinate of irregularly spaced events as a parameter to the state-space model as described in sec. 3.3, which is indicated by column  $t_k - t_{k-1}$ . Different variants of STREAM might apply linear transformations and Softplus activations similar to Mamba to adjust the time scales of  $e^{\mathbf{A}\Delta_k}$  (columns  $\Delta$ ) or the factor of  $\Gamma_k$  in equation (12) (columns  $\Gamma$ ).

(BMBF) within the project EVENTS (16ME0733). KKN is funded by the German Federal Ministry of Education and Research (BMBF) within the KI-ASIC project (16ES0996). CM receives funding from the German Research Foundation (DFG, Deutsche Forschungsgemeinschaft) as part of Germany’s Excellence Strategy – EXC 2050/1 – Project ID 390696704 – Cluster of Excellence “Centre for Tactile Internet with Human-in-the-Loop” (CeTI) of Technische Universität Dresden. This work was partially funded by the German Federal Ministry of Education and Research (BMBF) and the free state of Saxony within the ScaDS.AI center of excellence for AI research. The authors gratefully acknowledge the computing time made available to them on the high-performance computer at the NHR Center of TU Dresden. This center is jointly supported by the Federal Ministry of Education and Research and the state governments participating in the NHR ([www.nhr-verein.de/unsere-partner](http://www.nhr-verein.de/unsere-partner)). The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. ([www.gauss-centre.eu](http://www.gauss-centre.eu)) for funding this project by providing computing time on the GCS Supercomputer JUWELS [16] at Jülich Supercomputing Centre (JSC).

## References

- [1] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, Jeff Kusnitz, Michael Debole, Steve Esser, Tobi Delbruck, Myron Flickner, and Dharmendra Modha. A low power, fully event-based gesture recognition system. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7388–7397, 2017. 2, 5, 7, 8, 9
- [2] Alexandre Bittar and Philip N. Garner. A surrogate gradient spiking baseline for speech command recognition. *Frontiers in Neuroscience*, 16, 2022. 8
- [3] Guy E Blelloch. Prefix sums and their applications, 1990. 5, 2
- [4] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, Los Alamitos, CA, USA, 2017. IEEE Computer Society. 1, 2, 7
- [5] Guangyan Chen, Meiling Wang, Yi Yang, Kai Yu, Li Yuan, and Yufeng Yue. Pointgpt: Auto-regressively generative pre-training from point clouds. In *Advances in Neural Information Processing Systems*, pages 29667–29679. Curran Associates, Inc., 2023. 1, 2, 5, 7
- [6] Jiaqi Chen, Yan Yang, Shizhuo Deng, Da Teng, and Liyuan Pan. Spikmamba: When snn meets mamba in event-based human action recognition, 2024. 3, 7, 1
- [7] Benjamin Cramer, Yannik Stradmann, Johannes Schemmel, and Friedemann Zenke. The heidelberg spiking data sets for the systematic evaluation of spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33(7):2744–2757, 2022. 8, 9
- [8] Runpei Dong, Zekun Qi, Linfeng Zhang, Junbo Zhang, Jianjian Sun, Zheng Ge, Li Yi, and Kaisheng Ma. Autoencoders as cross-modal teachers: Can pretrained 2d image transformers help 3d representation learning? In *The Eleventh International Conference on Learning Representations*, 2023. 7
- [9] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024. 2, 3, 4, 5, 8, 9, 1
- [10] Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state space layers. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 572–585, 2021. 2, 3
- [11] Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022. 2, 4, 1
- [12] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R. Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, 2021. 7
- [13] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point

- clouds: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(12):4338–4364, 2021. 1
- [14] Ilyass Hammouamri, Ismail Khalfaoui-Hassani, and Timothée Masquelier. Learning delays in spiking neural networks using dilated convolutions with learnable spacings. In *The Twelfth International Conference on Learning Representations*, 2024. 8
- [15] Cheng-Yao Hong, Yu-Ying Chou, and Tyng-Luh Liu. Attention discriminant sampling for point clouds. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14383–14394, 2023. 7
- [16] Jülich Supercomputing Centre. JUWELS Cluster and Booster: Exascale Pathfinder with Modular Supercomputing Architecture at Jülich Supercomputing Centre. *Journal of large-scale research facilities*, 7(A138), 2021. 9
- [17] Sanghyeon Kim and Eunbyung Park. SMPConv: Self-Moving Point Representations for Continuous Convolution. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10289–10299, 2023. ISSN: 2575-7075. 2
- [18] Xavier Lagorce, Garrick Orchard, Francesco Galluppi, Bertram E. Shi, and Ryad B. Benosman. Hots: A hierarchy of event-based time-surfaces for pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(7):1346–1359, 2017. 3
- [19] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018. 7
- [20] Dingkan Liang, Xin Zhou, Wei Xu, Xingkui Zhu, Zhikang Zou, Xiaoqing Ye, Xiao Tan, and Xiang Bai. Pointmamba: A simple state space model for point cloud analysis. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 2, 3, 5, 6, 7, 8, 1
- [21] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A  $128 \times 128$  120 db 15  $\mu$ s latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566–576, 2008. 1
- [22] Chang Liu, Xiaojuan Qi, Edmund Y. Lam, and Ngai Wong. Fast classification and action recognition with event-based imaging. *IEEE Access*, 10:55638–55649, 2022. 7
- [23] Haotian Liu, Mu Cai, and Yong Jae Lee. Masked discrimination for self-supervised learning on point clouds. *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 7
- [24] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking network design and local geometry in point cloud: A simple residual MLP framework. In *International Conference on Learning Representations*, 2022. 7
- [25] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015. 1
- [26] Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In *Computer Vision—ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part II*, pages 604–621. Springer, 2022. 2, 3, 5, 7
- [27] Yan Ru Pei and Olivier Coenen. TENNs-PLEIADES: Building Temporal Kernels with Orthogonal Polynomials, 2024. arXiv:2405.12179. 3, 7, 8
- [28] Etienne Perot, Pierre de Tournemire, Davide Nitti, Jonathan Masci, and Amos Sironi. Learning to detect objects with a 1 megapixel event camera. In *Advances in Neural Information Processing Systems*, pages 16639–16652. Curran Associates, Inc., 2020. 1, 3
- [29] Anh Viet Phan, Minh Le Nguyen, Yen Lam Hoang Nguyen, and Lam Thu Bui. Dgcnn: A convolutional neural network over large-scale labeled graphs. *Neural Networks*, 108:533–543, 2018. 7
- [30] Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Re. Hyena hierarchy: Towards larger convolutional language models. In *Proceedings of the 40th International Conference on Machine Learning*, pages 28043–28078. PMLR, 2023. 3
- [31] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017. 2, 7
- [32] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Abed Al Kader Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: revisiting pointnet++ with improved training and scaling strategies. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2024. Curran Associates Inc. 7
- [33] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 2019. 6
- [34] Haoxi Ran, Jun Liu, and Chengjie Wang. Surface Representation for Point Clouds. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18920–18930, Los Alamitos, CA, USA, 2022. IEEE Computer Society. 7
- [35] Hongwei Ren, Yue Zhou, Haotian FU, Yulong Huang, Renjing Xu, and Bojun Cheng. Ttpoint: A tensorized point cloud network for lightweight action recognition with event cameras. In *Proceedings of the 31st ACM International Conference on Multimedia*, page 8026–8034, New York, NY, USA, 2023. Association for Computing Machinery. 2
- [36] Hongwei Ren, Yue Zhou, Jiadong Zhu, Haotian Fu, Yulong Huang, Xiaopeng Lin, Yuetong Fang, Fei Ma, Hao Yu, and Bojun Cheng. Rethinking efficient and effective point-based networks for event camera classification and regression: Eventmamba, 2024. 2, 3, 7, 1
- [37] Simon Schaefer, Daniel Gehrig, and Davide Scaramuzza. Aegnn: Asynchronous event-based graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12371–12381, 2022. 3, 7
- [38] Mark Schöne, Neeraj Mohan Sushma, Jingyue Zhuge, Christian Mayr, Anand Subramoney, and David Kappel. Scalable

- event-by-event processing of neuromorphic sensory signals with deep state-space models. In *ACM/IEEE International Conference on Neuromorphic Systems*. IEEE, 2024. 3, 7, 8
- [39] Yusuke Sekikawa, Kosuke Hara, and Hideo Saito. Eventnet: Asynchronous recursive event processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3, 7
- [40] Jimmy T.H. Smith, Andrew Warrington, and Scott Linderman. Simplified state space layers for sequence modeling. In *The Eleventh International Conference on Learning Representations*, 2023. 3, 1, 2
- [41] Taylan Soydan, Nikola Zubić, Nico Messikommer, Siddhartha Mishra, and Davide Scaramuzza. S7: Selective and simplified state space layers for sequence modeling, 2024. 3, 7, 8
- [42] Anand Subramoney, Khaleelulla Khan Nazeer, Mark Schöne, Christian Mayr, and David Kappel. Efficient recurrent architectures through activity sparsity and sparse back-propagation through time. In *The Eleventh International Conference on Learning Representations*, 2023. 3
- [43] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *International Conference on Computer Vision (ICCV)*, 2019. 2, 5, 6, 7
- [44] Peng-Shuai Wang. Octformer: Octree-based transformers for 3d point clouds. *ACM Trans. Graph.*, 42(4), 2023. 7
- [45] Qinyi Wang, Yexin Zhang, Junsong Yuan, and Yilong Lu. Space-time event clouds for gesture recognition: From rgb cameras to event cameras. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1826–1835, 2019. 2
- [46] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep Parametric Continuous Convolutional Neural Networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2589–2597, 2018. ISSN: 2575-7075. 1, 2, 4
- [47] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition, 2018. 8
- [48] Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep Convolutional Networks on 3D Point Clouds. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9613–9622, 2019. ISSN: 2575-7075. 1, 2, 4
- [49] Wenxuan Wu, Li Fuxin, and Qi Shan. PointConvFormer: Revenge of the Point-based Convolution. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21802–21813, 2023. ISSN: 2575-7075. 2
- [50] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1
- [51] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015. 2, 6
- [52] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-BERT: Pre-training 3D Point Cloud Transformers with Masked Point Modeling. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19291–19300, Los Alamitos, CA, USA, 2022. IEEE Computer Society. 2, 7
- [53] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 7
- [54] Renrui Zhang, Ziyu Guo, Rongyao Fang, Bin Zhao, Dong Wang, Yu Qiao, Hongsheng Li, and Peng Gao. Point-m2ae: multi-scale masked autoencoders for hierarchical point cloud pre-training. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2024. Curran Associates Inc. 7
- [55] Tao Zhang, Xiangtai Li, Haobo Yuan, Shunping Ji, and Shuicheng Yan. Point cloud mamba: Point cloud learning via state space model. *CoRR*, abs/2403.00762, 2024. 3, 7, 1
- [56] Jiazhou Zhou, Xu Zheng, Yuanhuiyi Lyu, and Lin Wang. Exact: Language-guided conceptual reasoning and uncertainty estimation for event-based action recognition and more. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18633–18643, 2024. 1, 3, 7
- [57] Nikola Zubic, Mathias Gehrig, and Davide Scaramuzza. State Space Models for Event Cameras. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5819–5828, Los Alamitos, CA, USA, 2024. IEEE Computer Society. 1, 3

# STREAM: A Universal State-Space Model for Sparse Geometric Data

## Supplementary Material

### 6. Derivations

This section provides complete derivations of equations (6) and (7). We restrict to the case of single-input single-output (SISO) state-space models with multi-dimensional state. Multi-input multi-output (MIMO) formulations like S4 [11] or Mamba [9] can be obtained by creating  $n$  parallel instances of the SISO model and mixing the input and output components with linear layers.

Our motivation was to reason about spatial relationships between input pulses. Therefore, we defined an interaction kernel  $\Phi(\mathbf{x}, \mathbf{x}')$  that models pairwise interactions. A complete view of all possible interactions with the point  $\mathbf{x}_k$  is given by

$$\mathbf{y}(\mathbf{x}_k) = \int \Phi(\mathbf{x}_k, \mathbf{x}') \mathbf{u}(\mathbf{x}') d\mathbf{x}' \quad (14)$$

$$= \sum_{i=0}^N \Phi(\mathbf{x}_k, \mathbf{x}_i) \mathbf{u}_i. \quad (15)$$

We will now derive a parameterization of the interaction kernel  $\Phi$  with a state-space model.

A linear time-varying state-space model acting on a scalar input function  $u(t)$  and producing a scalar output  $y(t)$  is defined through

$$\dot{\mathbf{h}}(t) = \mathbf{A}(t) \mathbf{h}(t) + \mathbf{B}(t) u(t) \quad (16)$$

$$y(t) = \mathbf{C}(t) \mathbf{h}(t), \quad (17)$$

with  $u(t), y(t) \in \mathbb{R}$ , states  $\mathbf{h}(t) \in \mathbb{R}^m$  and parameters  $\mathbf{A}(t) \in \mathbb{R}^{m \times m}, \mathbf{B}(t) \in \mathbb{R}^{m \times 1}, \mathbf{C}(t) \in \mathbb{R}^{1 \times m}$ .

Note that we stick to our notation introduced in sec. 3, denoting vectors and matrices with bold face and scalar values with regular face.

#### 6.1. Solution of the Linear State-space Model

Linear ordinary differential equations have a well known analytical solution. We refer the reader to standard calculus textbooks. As such, the linear dynamics of equation (16) for initial value  $h_0 = h(t_0)$  admit the analytical solution

$$\mathbf{h}(t) = h_0 + \int_{t_0}^t \exp\left(\int_{t'}^t \mathbf{A}(t'') dt''\right) \mathbf{B}(t') u(t') dt', \quad (18)$$

which can be checked by taking the derivative of  $\mathbf{h}$  and comparing it to equation (16). Comparing equations (18), (17) and (14), we read off the kernel

$$\Phi(t, t') = \mathbf{C}(t) \exp\left(\int_{t'}^t \mathbf{A}(t'') dt''\right) \mathbf{B}(t') \quad (19)$$

Note that equation (18) can be evaluated on all coordinates  $t > t_0$ . The two canonical options for discretizing the variables  $h$  and  $y$  are equidistant steps resulting in a regular grid, or using the input coordinates  $t_0, \dots, t_N$ . While most state-space model works discretize on equidistant steps, [40] shows on a toy task that the SSM formulation is capable of solving tasks with irregularly spaced steps as well.

Our work differs from other recent Mamba based models such as PointMamba [20], Point Cloud Mamba [55], EventMamba [36], or SpikMamba [6] by integrating the true timings of the inputs in the following sense.

We will use  $h_0 = 0$  for notational simplicity in the following. Discretizing equation (18) on the Dirac delta coded input (1) then yields

$$\begin{aligned} \mathbf{h}(t_k) &= \int_{t_0}^{t_k} \exp\left(\int_t^{t_k} \mathbf{A}(t') dt'\right) \mathbf{B}(t) \sum_{i=0}^N \delta(t - t_i) u_i dt \\ &= \sum_{i=0}^k \exp\left(\int_{t_i}^{t_k} \mathbf{A}(t) dt\right) \mathbf{B}(t_i) u_i. \end{aligned} \quad (20)$$

We decompose the integral from  $t_i$  to  $t_k$  into the integrals from  $t_{j-1}$  to  $t_j$  and sum over them

$$\mathbf{h}(t_k) = \sum_{i=0}^k \exp\left(\sum_{j=i+1}^k \int_{t_{j-1}}^{t_j} \mathbf{A}(t) dt\right) \mathbf{B}(t_i) u_i. \quad (21)$$

We will further assume that  $\mathbf{A}(t), \mathbf{B}(t), \mathbf{C}(t)$  are constant on the intervals  $(t_{j-1}, t_j]$  for  $j = i+1, \dots, k$ . Denoting  $\mathbf{A}(t_j) = \mathbf{A}_j, \mathbf{B}(t_j) = \mathbf{B}_j, \mathbf{C}(t_j) = \mathbf{C}_j, \mathbf{h}(t_k) = \mathbf{h}_k$ , and  $\Delta_j = t_j - t_{j-1}$  we get

$$\mathbf{h}(t_k) = \sum_{i=0}^k \exp\left(\sum_{j=i+1}^k \mathbf{A}_j \Delta_j\right) \mathbf{B}_i u_i \quad (22)$$

$$= \sum_{i=0}^k \left(\prod_{j=i+1}^k \exp(\mathbf{A}_j \Delta_j)\right) \mathbf{B}_i u_i. \quad (23)$$

Comparing to equation (15), we read off the discrete kernel function

$$\Phi(t_k, t_i) = \mathbf{C}_k \left(\prod_{j=i+1}^k \exp(\mathbf{A}_j \Delta_j)\right) \mathbf{B}_i. \quad (24)$$

**Proposition 1.** *The kernels parameterized by equation (24) contain convolution operations with rational kernels as a special case.*

*Proof.* Consider the linear time-invariant (LTI) case where  $\mathbf{A}, \mathbf{B}, \mathbf{C}$  are constant for all  $t$ . Then

$$\prod_{j=i+1}^k \exp(\mathbf{A}_j \Delta_j) = \exp(\mathbf{A}(t_k - t_i)) \quad (25)$$

The proposition follows from the fundamental result from linear systems theory that LTI state-space models can represent every rational transfer function.  $\square$

## 6.2. Recursive Computation of the Interaction Kernel

Let's expand equation (23) in a recursive form

$$\begin{aligned} \mathbf{h}_k &= \mathbf{h}(x_k) = \sum_{i=0}^k \left( \prod_{j=i+1}^k e^{\mathbf{A}_j \Delta_j} \right) \mathbf{B}_i u_i \\ &= \mathbf{B} \mathbf{u}_k + \sum_{i=0}^{k-1} \left( \prod_{j=i+1}^k e^{\mathbf{A}_j \Delta_j} \right) \mathbf{B}_i u_i \\ &= \mathbf{B} \mathbf{u}_k + \sum_{i=0}^{k-1} e^{\mathbf{A}_k \Delta_k} \left( \prod_{j=i+1}^{k-1} e^{\mathbf{A}_j \Delta_j} \right) \mathbf{B}_i u_i \\ &= \mathbf{B} \mathbf{u}_k + e^{\mathbf{A}_k \Delta_k} \sum_{i=0}^{k-1} \left( \prod_{j=i+1}^{k-1} e^{\mathbf{A}_j \Delta_j} \right) \mathbf{B}_i u_i \\ &= \mathbf{B} \mathbf{u}_k + e^{\mathbf{A}_k \Delta_k} \mathbf{h}_{k-1}. \end{aligned}$$

This concludes the derivation of our recurrent operator

$$\mathbf{h}_k = \mathbf{B} \mathbf{u}_k + e^{\mathbf{A}_k \Delta_k} \mathbf{h}_{k-1}. \quad (26)$$

In line with most recent SSM works, we choose  $\mathbf{A}_i \equiv \mathbf{A}$  as a diagonal matrix for all  $i$ .

**Remark 1.** Equation (26) allows asynchronous inference on streams of incoming coordinates in  $\mathcal{O}(1)$  time per coordinate, or  $\mathcal{O}(N)$  time for the full stream of coordinates.

## 7. Scan

Linear time-varying systems such as the one given by equation (26) resemble an associative operation, with well-known time complexity of  $\mathcal{O}(\log N)$  [3]. The goal of this section is not to provide a proof, but to give the reader a clear idea of how irregularly spaced sequences can be parallelized in the same way that regular SSMs can be parallelized with the Scan primitive [9, 40]. The presentation follows [3].

Consider the pair

$$\mathbf{c}_i = [\mathbf{a}_i, \mathbf{b}_i] \quad (27)$$

with the binary operator  $\bullet$  defined through

$$\mathbf{c}_i \bullet \mathbf{c}_j = [\mathbf{a}_i \cdot \mathbf{a}_j, \mathbf{a}_j \cdot \mathbf{b}_i + \mathbf{b}_j]. \quad (28)$$

As [3] shows, the operator  $\bullet$  is associative, i.e.

$$(\mathbf{c}_i \bullet \mathbf{c}_j) \bullet \mathbf{c}_k = \mathbf{c}_i \bullet (\mathbf{c}_j \bullet \mathbf{c}_k). \quad (29)$$

Associative operators can be parallelized to run in  $\mathcal{O}(\log N)$  time on sequence of length  $N$  given sufficiently many processors [3]. We use this primitive to parallelize our model. The pairs  $\mathbf{c}$  are initialized as

$$\mathbf{c}_0 = [e^{\mathbf{A} \Delta_0}, \mathbf{B}_0 \mathbf{u}_0] \quad \dots \quad \mathbf{c}_N = [e^{\mathbf{A} \Delta_N}, \mathbf{B}_N \mathbf{u}_N] \quad (30)$$

Fig. 5 shows an example of how the Scan primitive computes the entire recurrence in equation (26) in roughly  $2 \log N$  steps.



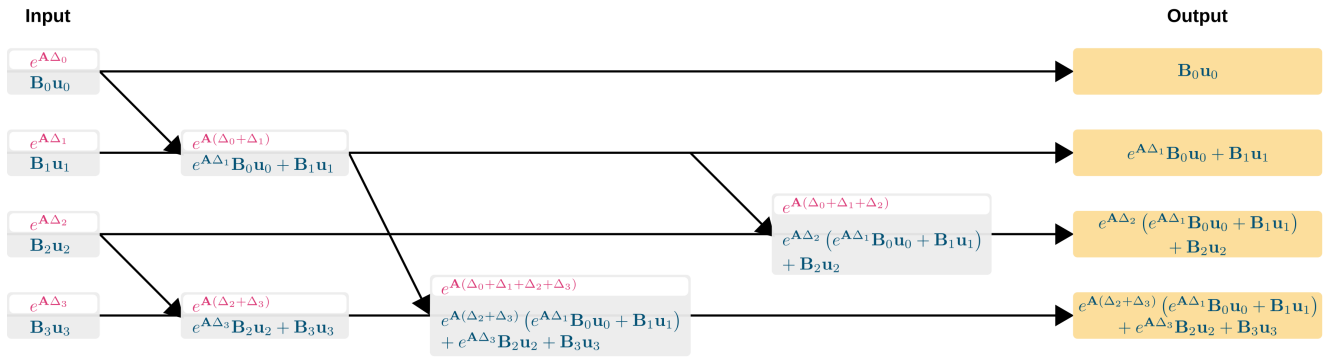


Figure 5. Caption