ICCV HONOLULU HAWAII OCT 19-23, 2025

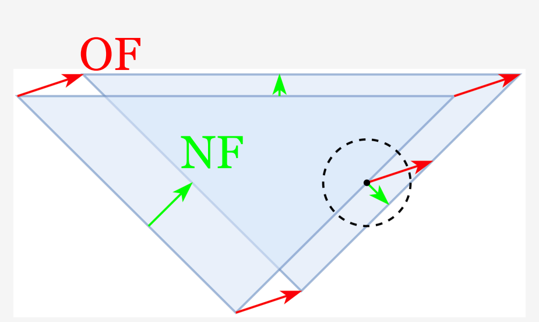
Learning Normal Flow Directly From Events

Dehao Yuan[1,2], Levi Burner[1], Jiayi Wu[1], Minghui Liu[1], Jingxi Chen[1], Yiannis Aloimonos[1], Cornelia Fermüller[1]
[1]University of Maryland, College Park [2]Capital One dhyuan99@gmail.com

UMIACS Capital One PRG

Motivation & Summary

	optical flow	normal flow
dimensionality	2-dimension	1-dimension
aperture problem?	Yes	No
estimated from	global region	local region



- Normal flow can be more reliably estimated than optical flow, because normal flow can be estimated from **local receptive field**.
- Unlike optical flow estimator that requires CNN-RNN, a normal flow estimator only requires **mapping local events to normal flow prediction**.
- How about losing 1-dimensional motion information? Don't worry! We established **egomotion** and **motion segmentation** algorithms with normal flow inputs!

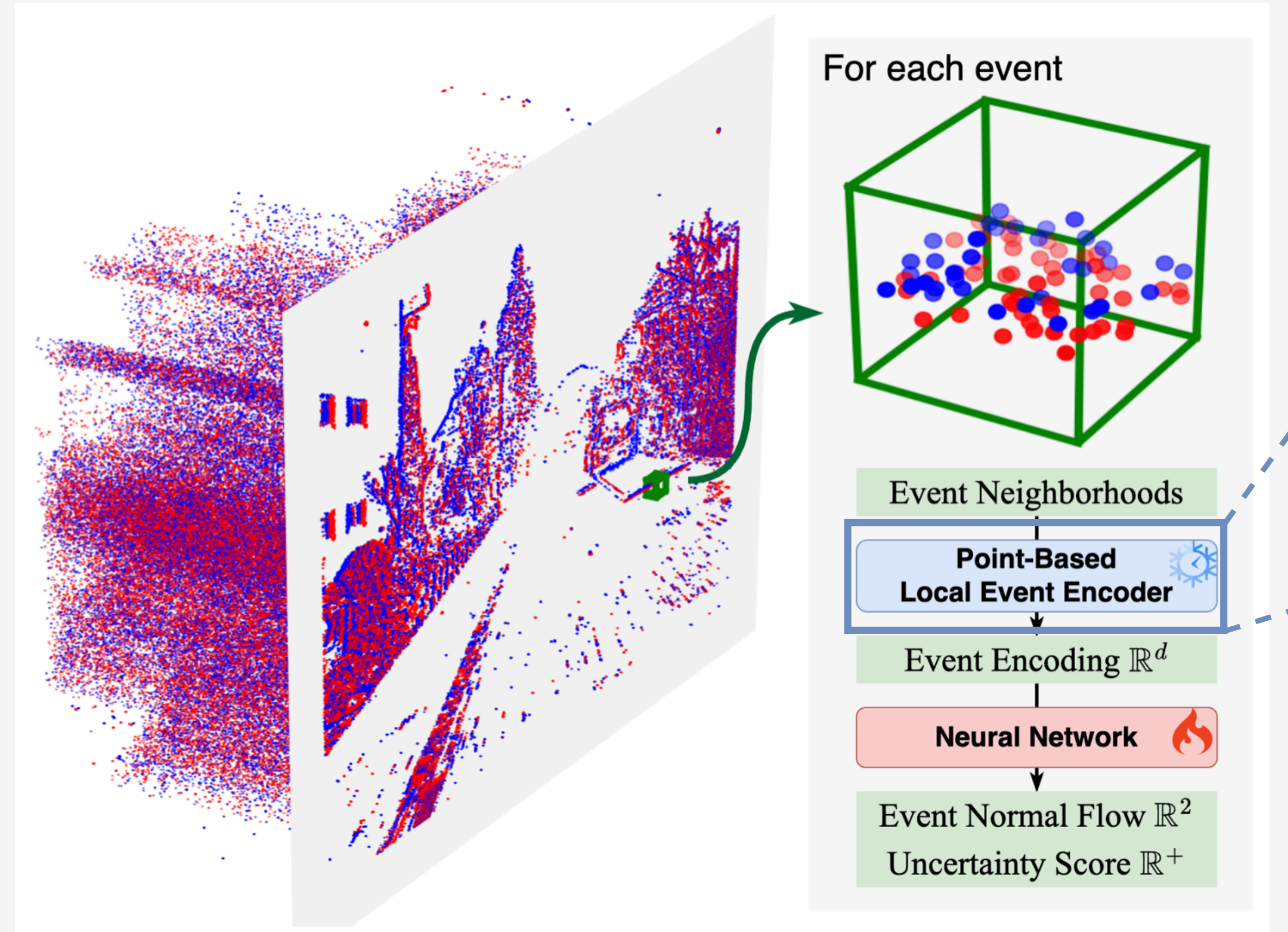

For each event
Event Neighborhoods
Point-Based Local Event Encoder
Event Encoding $\mathbb{R}^d$
Neural Network
Event Normal Flow $\mathbb{R}^2$
Uncertainty Score $\mathbb{R}^+$

Figure 1: Our normal flow estimator simply maps local events to a normal flow prediction, and a coupled uncertainty score.

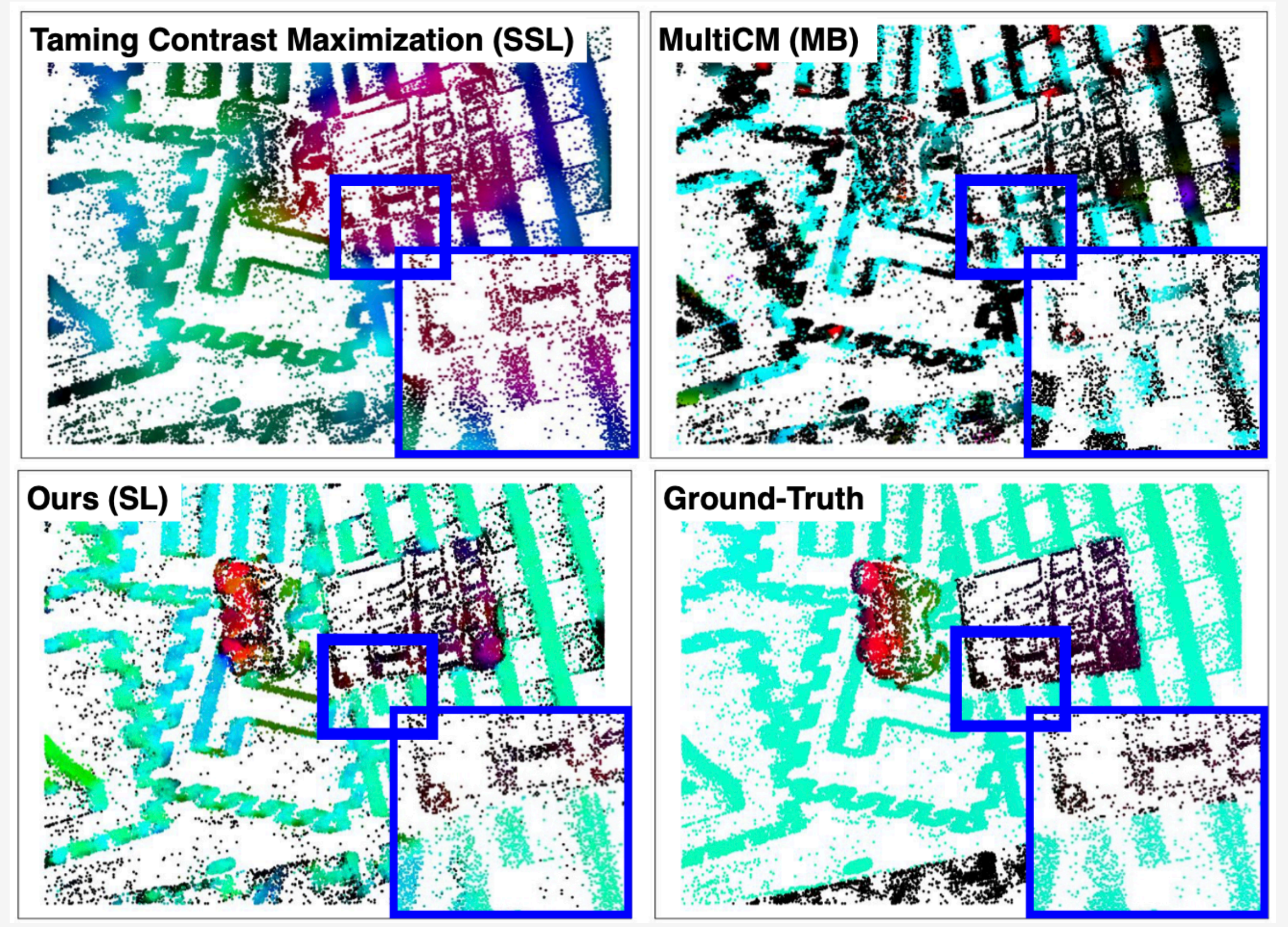
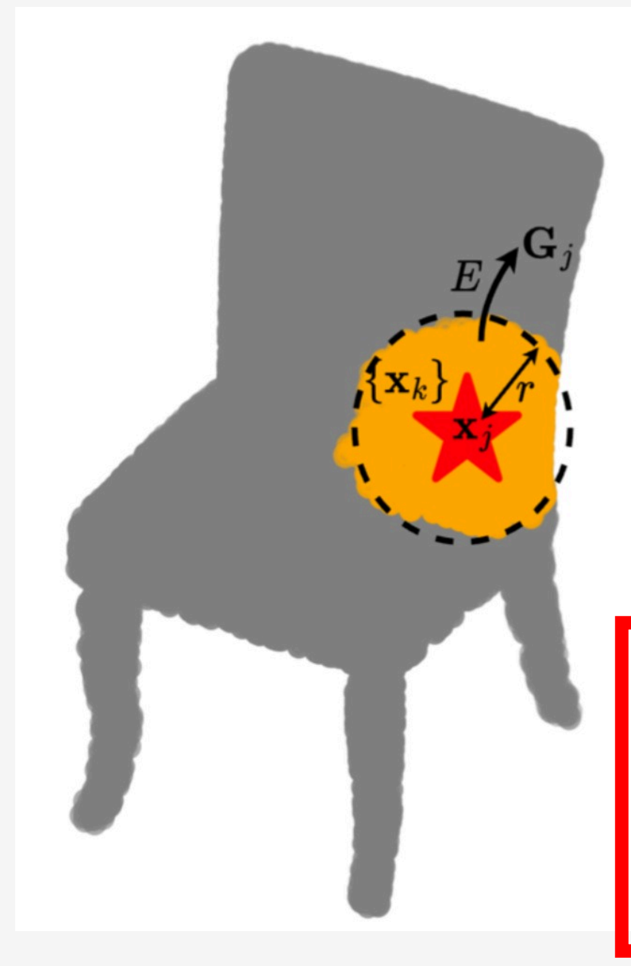


Figure 2: Our estimator produces accurate and sharp predictions in the presence of independently moving objects

VecKM Local Event Encoder

We use a **scalable** local point cloud encoder **VecKM**.



Local Point Cloud Encoding

- Input: point cloud $\mathbf{X}_{n\times 3}$.
- For each point $\mathbf{x}_j$, compute centerized local point cloud $\mathfrak{N}(\mathbf{x}_j) = \{\mathbf{x}_k - \mathbf{x}_j : ||\mathbf{x}_k - \mathbf{x}_j|| < r\}$.
- Output: $\mathbf{G}_{n\times d}$, where $\mathbf{G}_j = E(\mathfrak{N}(\mathbf{x}_j))$.
- E is called local point cloud encoder.

Naive Mini-PointNet does not work:

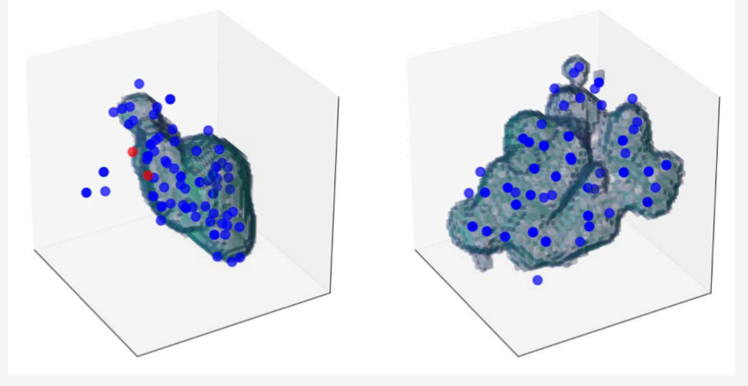
- For each point $\mathbf{x}_j$, **sample** K points from $\mathfrak{N}(\mathbf{x}_j)$. Get per-point neighborhood (n, K, 3).
- Apply PointNet on each neighborhood. (n, K, 3) $\to$ (n, K, d) $\to$ (n, d)

What VecKM does:

- Use **every** neighboring points $\mathfrak{N}(\mathbf{x}_j)$ to compute $\mathbf{G}_j$. **No Sampling**, so more **descriptive**.
- **Scalable** to large point cloud size. $n > 80000$ with 12 GB memory. This is essential for event cameras!

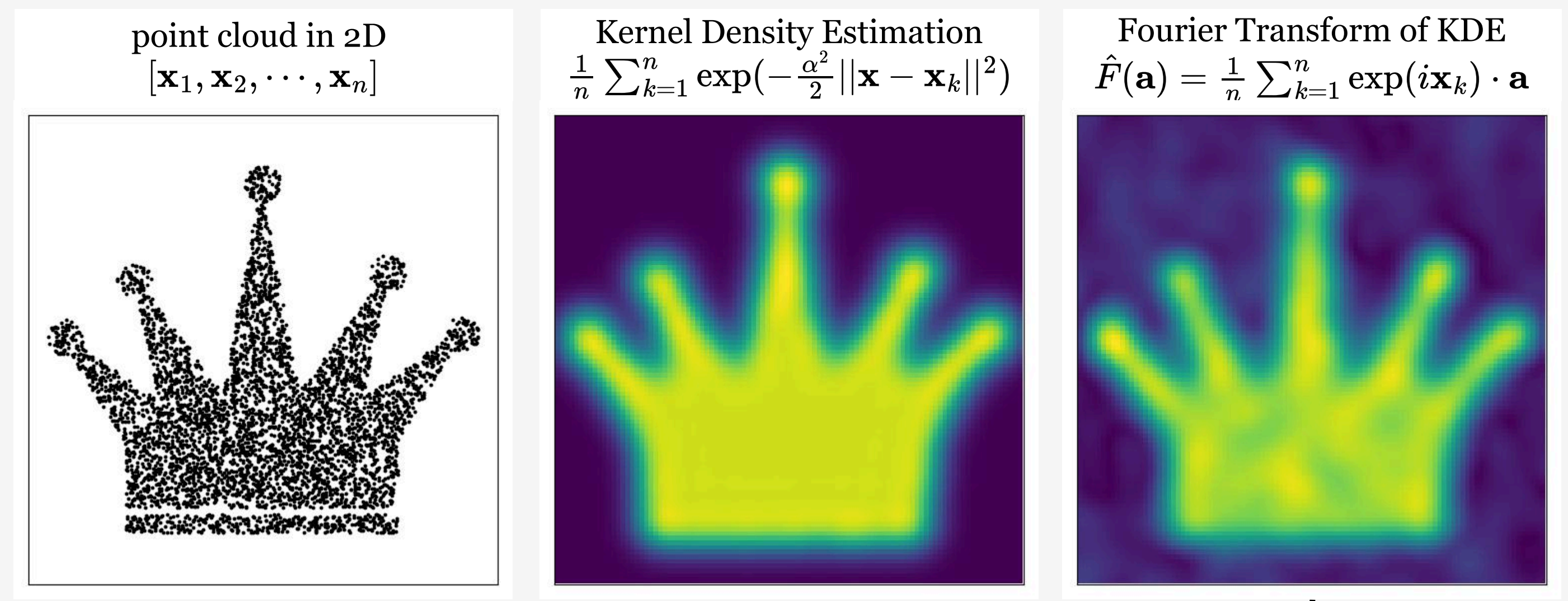
Input Point Cloud: $\mathbf{X}_{n\times 3} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]$
VecKM Initialize: $\mathbf{A}_{3\times d} \sim \mathcal{N}(0, \alpha^2)$.
VecKM Encoding $\mathbf{G}_{n\times d}$ is obtained by:

$$\mathcal{A}_{n\times d} = \exp(i\mathbf{X}_{n\times 3}\mathbf{A}_{3\times d})$$
$$\mathcal{J}_{n\times n} = \mathbb{I}(||\mathbf{x}_i - \mathbf{x}_j|| < r)$$
$$\mathbf{G}_{n\times d} = normalize\big((\mathcal{J} \times \mathcal{A})./\mathcal{A}\big)$$



Encoding quality check provided by VecKM. Blue and red dots are events. Gray regions are reconstructed event distribution from encoding.

Intuition behind VecKM


point cloud in 2D
$[\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]$
Kernel Density Estimation
$\frac{1}{n}\sum_{k=1}^{n}\exp(-\frac{\alpha^2}{2}||\mathbf{x} - \mathbf{x}_k||^2)$
Fourier Transform of KDE
$\hat{F}(\mathbf{a}) = \frac{1}{n}\sum_{k=1}^{n}\exp(i\mathbf{x}_k)\cdot\mathbf{a}$

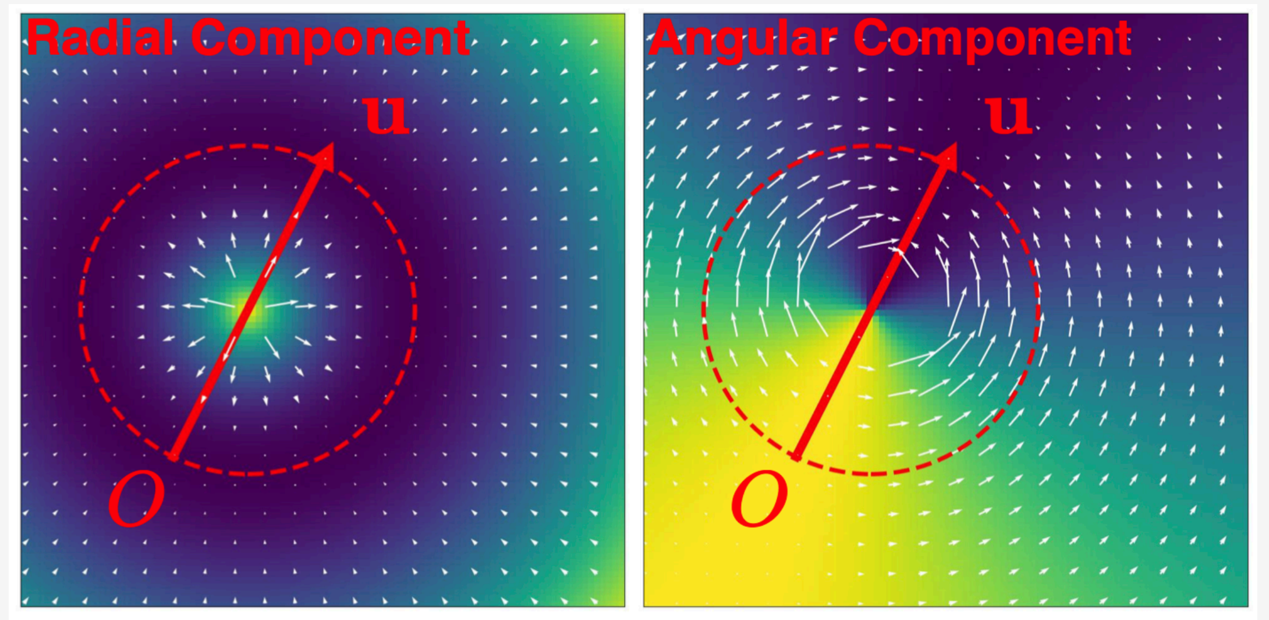
Sampling different **a** gives VecKM encoding.

More About VecKM

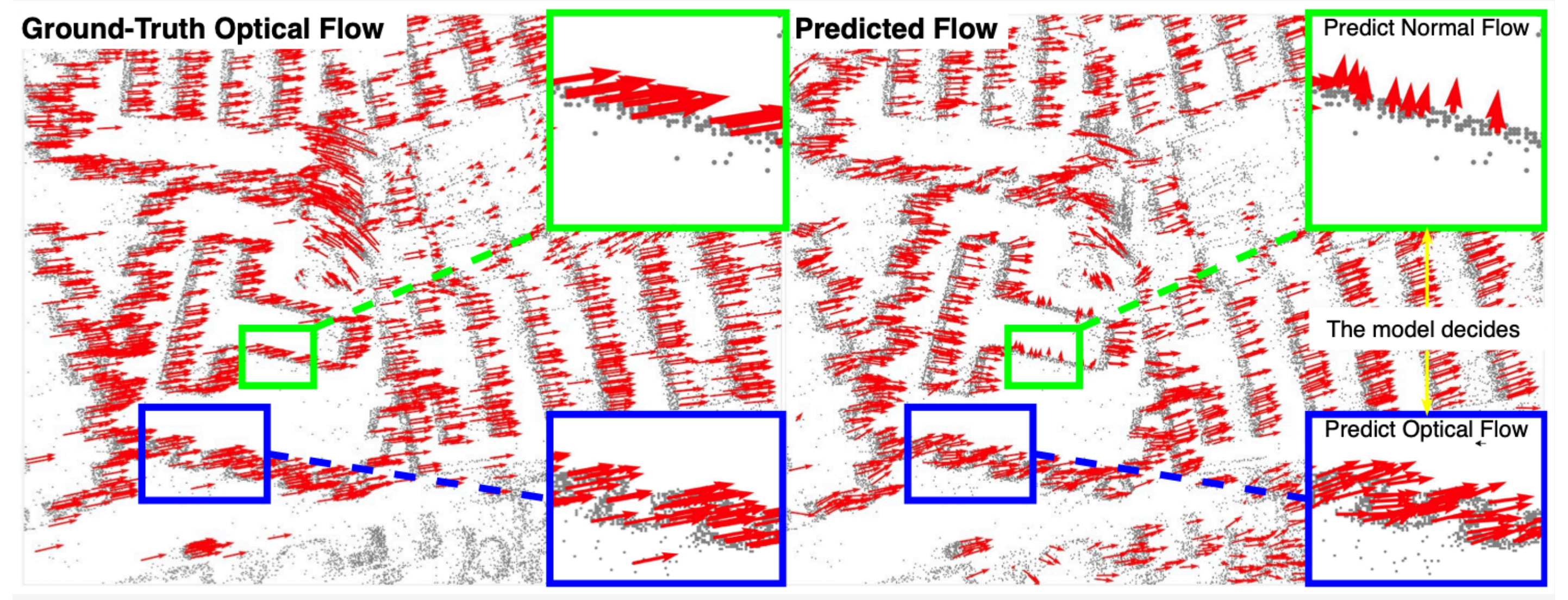
Yuan, Dehao, Cornelia Fermüller, Tahseen Rabbani, Furong Huang, and Yiannis Aloimonos. “A Linear Time and Space Local Point Cloud Geometry Encoder via Vectorized Kernel Mixture (VecKM).” ICML2024.
https://arxiv.org/abs/2404.01568
https://github.com/dhyuan99/VecKM

Or Just Google VecKM.

Normal Flow Loss Function



Satisfy normal flow constraint. If possible, predict optical flow.



The model by itself decides whether predicting optical flow or normal flow.

Uncertainty Scores

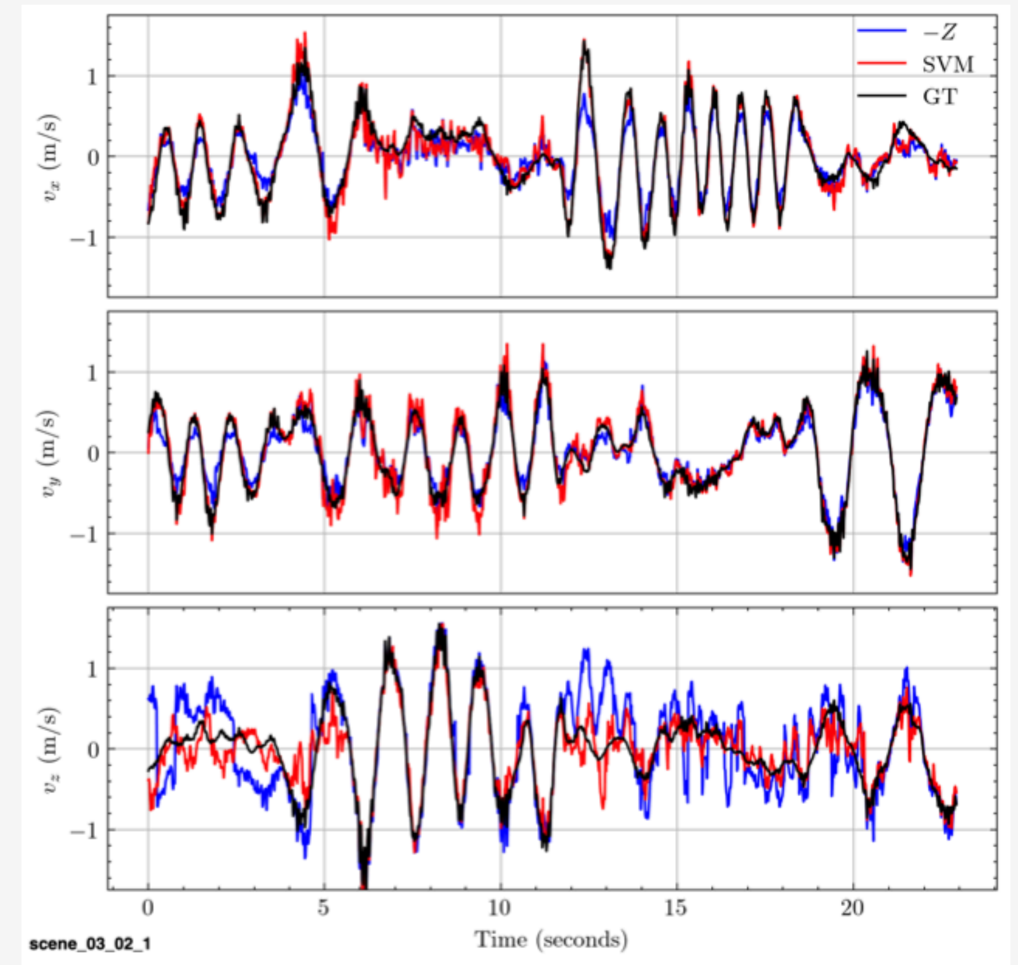
We have an **ensemble** strategy to output **prediction uncertainty**.

$$\hat{U} = estimator\left(X_{N\times 3}\begin{bmatrix}1 & \mathbf{0}^T \\ \mathbf{0} & R(\theta)\end{bmatrix}\right)R(\theta)^{-1}$$

Sample θ for K times and compute mean and std accordingly.

Egomotion Estimation

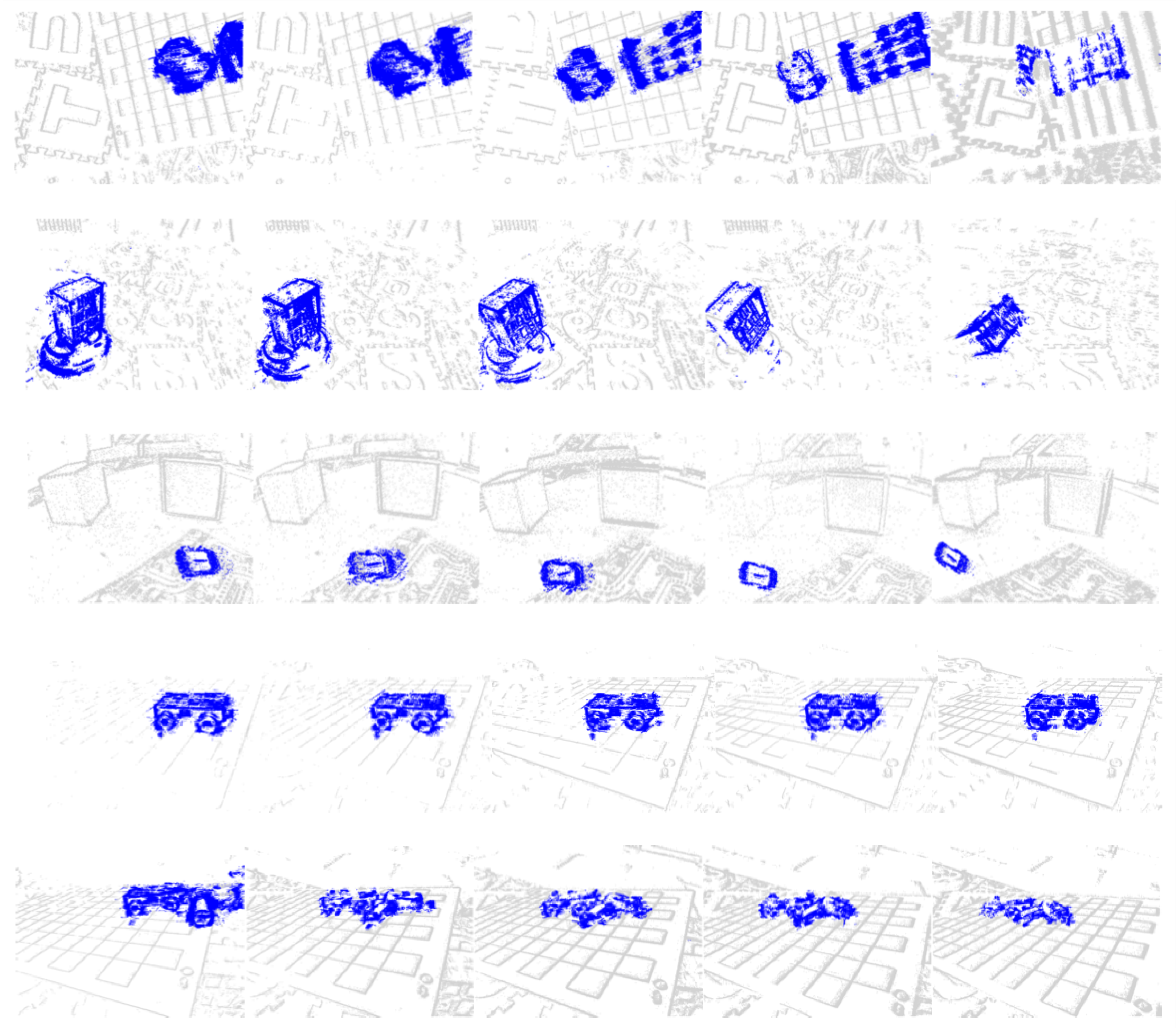
We derived an SVM-based egomotion solver with normal flow inputs.



accurate even on **very fast motions.**

Motion Segmentation

We derived a motion segmentation algorithm with normal flow inputs.



Hua, Zhiyuan, Dehao Yuan, and Cornelia Fermüller. "Motion Segmentation and Egomotion Estimation from Event-Based Normal Flow." arXiv preprint arXiv:2507.14500 (2025).
https://arxiv.org/abs/2507.14500

Paper Series and Codes

Yuan, Dehao, Cornelia Fermüller, Tahseen Rabbani, Furong Huang, and Yiannis Aloimonos. “A Linear Time and Space Local Point Cloud Geometry Encoder via Vectorized Kernel Mixture (VecKM).” ICML2024.
https://arxiv.org/abs/2404.01568
https://github.com/dhyuan99/VecKM

Yuan, Dehao, Levi Burner, Jiayi Wu, Minghui Liu, Jingxi Chen, Yiannis Aloimonos, and Cornelia Fermüller. "Learning normal flow directly from event neighborhoods." ICCV2025.
https://arxiv.org/abs/2412.11284
https://github.com/dhyuan99/VecKM_flow

Yuan, Dehao, and Cornelia Fermüller. "A Real-Time Event-Based Normal Flow Estimator." Arxiv2025.
https://arxiv.org/abs/2504.19417
https://github.com/dhyuan99/VecKM_flow_cpp

Hua, Zhiyuan, Dehao Yuan, and Cornelia Fermüller. "Motion Segmentation and Egomotion Estimation from Event-Based Normal Flow." Arxiv2025.
https://arxiv.org/abs/2507.14500