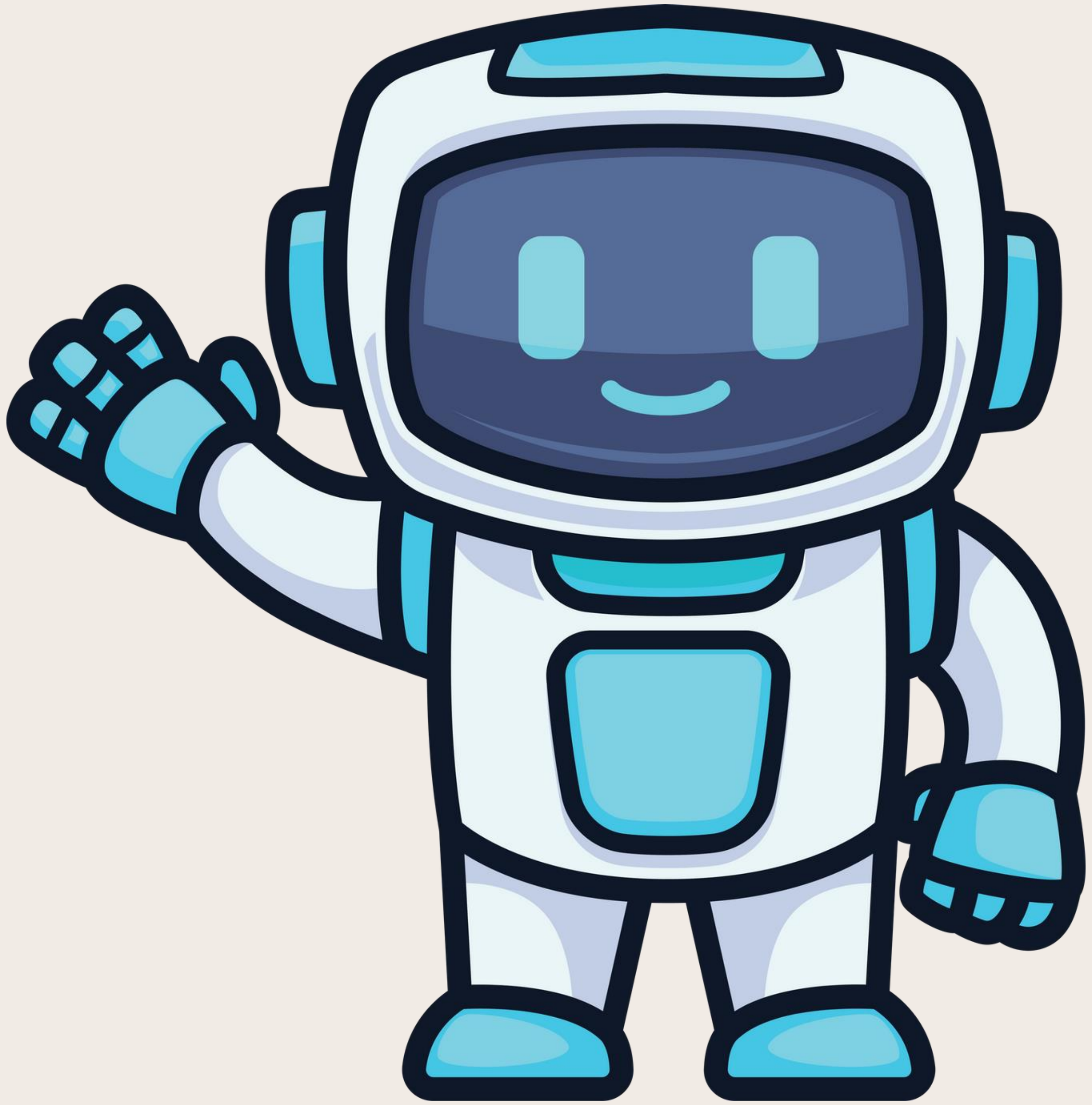
HAND GESTURE 3D POSE ESTIMATION AND RECOGNITION BASED ON EVENT CAMERA RECORDS

FINAL PRESENTATION ROBOTICS PERCEPTION



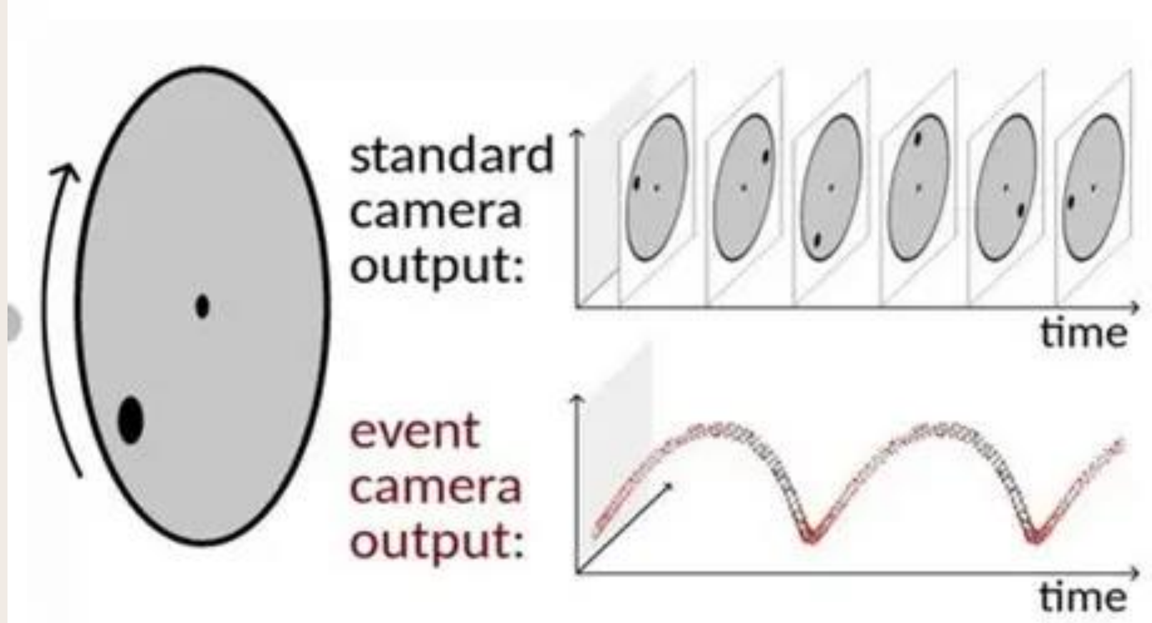
Presented by: **Shamma Alblooshi 100045387**

PROBLEM STATEMENT AND OBJECTIVES

INTRODUCTION

- Event-based vision mimics the human eye by capturing changes in the scene, offering advantages such as:
  - High temporal resolution: Records data at microsecond intervals.
  - Low latency: Enables real-time processing.
  - Power efficiency: Only processes changes, reducing computational overhead.

- Applications of event-based vision include robotics, augmented reality (AR), virtual reality (VR), and human-computer interaction.



3D Hand Gesture Posture and Recognition

- 3D hand gesture posture estimation involves determining the precise 3D positions of hand joints, enabling applications like:
  - Gesture-based interaction in virtual reality (VR) and augmented reality (AR).
  - Real-time control in robotics and gaming.
  - Sign language interpretation for improved accessibility.

Challenges with existing systems:

Motion Blur and Latency:

- Traditional vision systems struggle with motion blur during fast hand movements.
- Latency in processing full video frames impacts real-time applications.

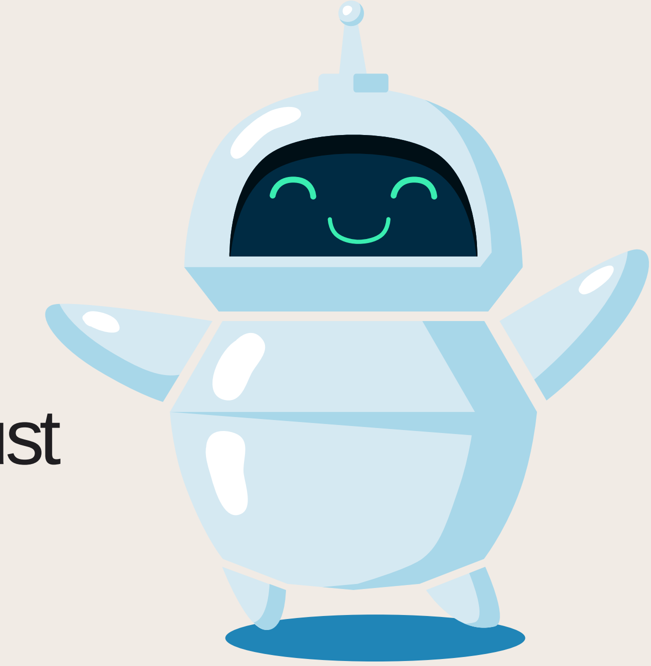
Computational Cost:

- Dense image data requires significant computational resources, which is unsuitable for low-power devices.

Event cameras overcome these issues, providing high-resolution temporal data, which is crucial for accurate and real-time 3D hand posture recognition.

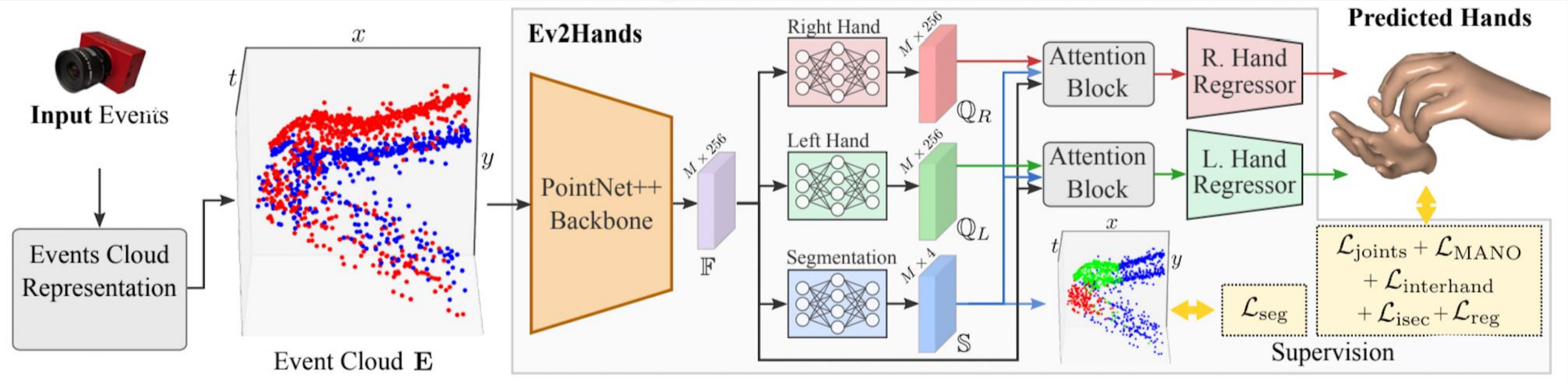
Proposed Solution:

- Utilize the Ev2Hands framework, which combines:
  - Event-based data streams: Captures only changes in the scene.
  - Pre-trained 3D hand pose models: Efficiently estimate joint positions.
  - Hierarchical feature extraction: Learns local and global features for robust predictions.



METHODOLOGY

FULL FRAMEWORK



CONVERTING EVENT DATA TO EVENT CLOUD REPRESENTATION

- **Input**:
  - Raw event stream data from the event camera (contains spatiotemporal information: [x, y, t, p], where p is polarity).

$$\mathbf{e}_i = (x_i, y_i, t_i, p_i)$$

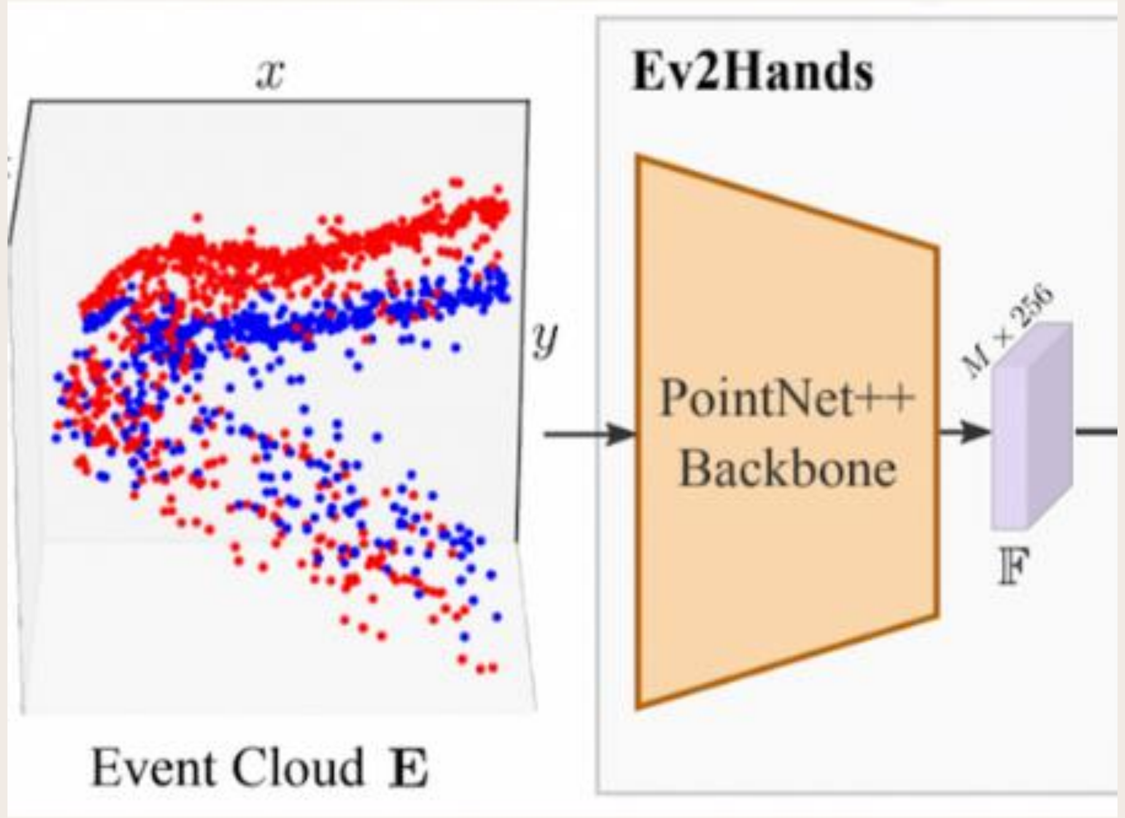
  - High temporal resolution data for capturing dynamic hand movements.
- **Process**:
  - Convert the event data into a structured **event cloud representation**, which is a set of points in a 3D space:

$$\mathbf{E}_k = (x_k, y_k, t_k, P_k, N_k)$$

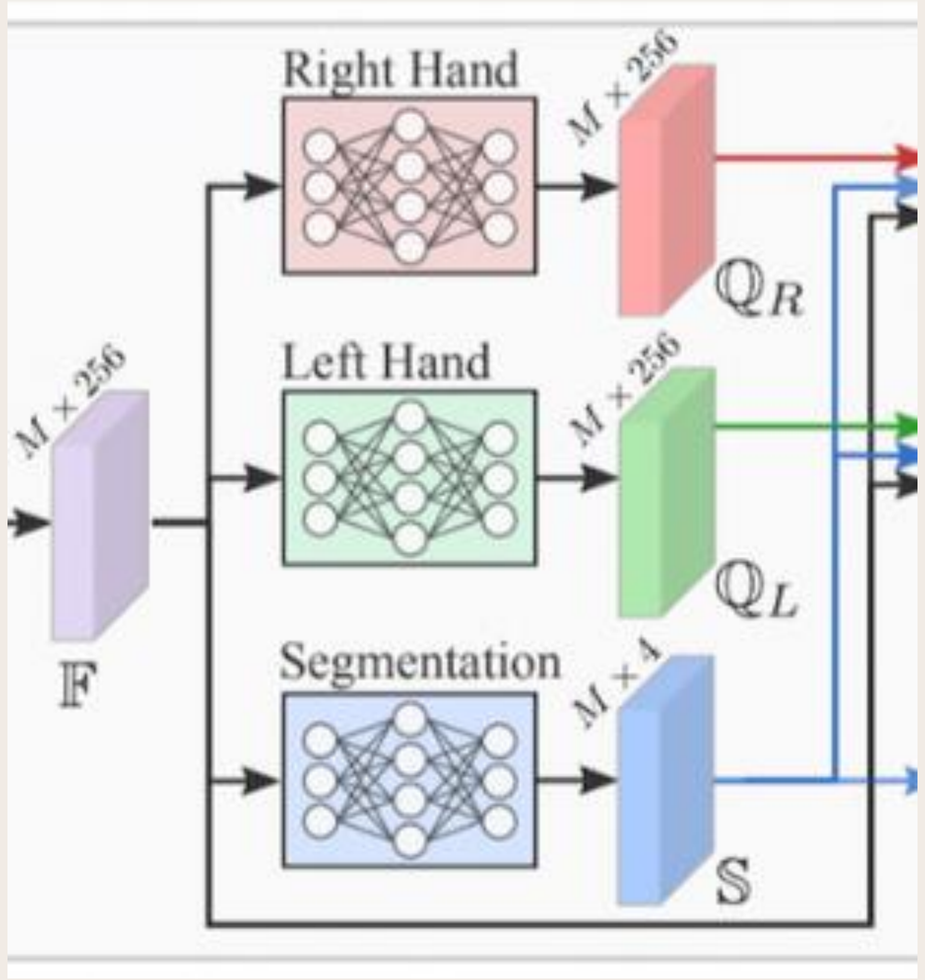
    - x, y: pixel coordinates
    - t: average time of the combined events
    - P, N: number of positive and negative events in the time interval considered
- **Output**:
  - A normalized **event cloud** that represents the spatiotemporal and polarity information of the input events.
  - Enables efficient processing using deep learning frameworks like PointNet++.

PROCESSING EVENT CLOUD WITH POINTNET++

- **Input**:
  - The normalized event cloud from the previous step.
- **Process**:
  - **PointNet++** architecture processes the event cloud hierarchically:
    - **Sampling**: Uses Farthest Point Sampling (FPS) to reduce the number of points while maintaining a uniform distribution.
    - **Grouping**: Applies ball query to group nearby points into local regions.
    - **Feature Learning**: Uses MLPs and max pooling to extract local and global features from the grouped points.
- **Output**:
  - **Event Feature Representation**:
    - A feature vector capturing the spatial and temporal relationships in the event cloud.
    - Encodes hand gesture dynamics for further processing.

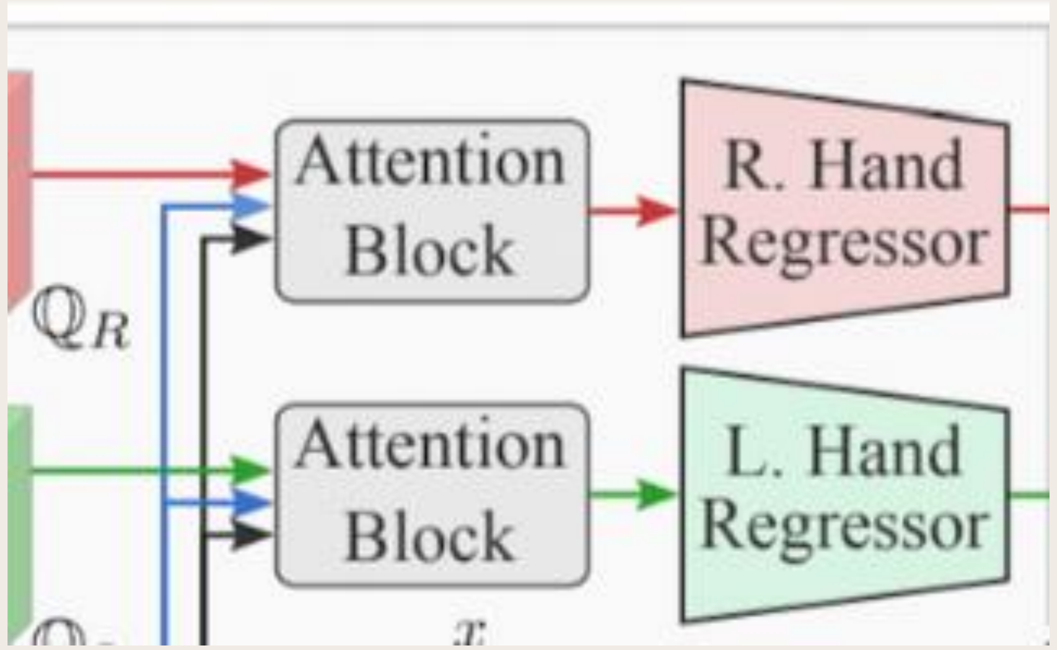


BRANCHING THE EVENT FEATURE REPRESENTATION



- **Input**:
  - Event features extracted from PointNet++.
- **Process**:
  - The event feature vector is passed through three parallel branches:
    i. **Left Hand Branch**:
       - Predicts features specific to the left hand.
       - Outputs: QL (feature vector for the left hand).
    ii. **Right Hand Branch**:
       - Predicts features specific to the right hand.
       - Outputs: QR (feature vector for the right hand).
    iii. **Segmentation Branch**:
       - Processes the event features to predict class logits for each point in the event cloud.
       - Outputs: S , which is a segmentation map
- **Outputs**:
  - Three feature vectors: QL, QR, and S, representing left hand, right hand, and segmenattion, respectively.

ENHANCING FEATURES WITH ATTENTION



- **Input**:
  - Feature vectors: QL (left hand), QR (right hand), S
- **Process**:
  - **Attention Block**:
    - Enhances the left (QL) and right (QR) hand features using attention mechanisms.
    - Computes the relationship between QL, QR, and S to focus on important spatial-temporal patterns.
    - The attention block adjusts the feature vectors to highlight critical regions and suppress irrelevant information.

$$\text{Attention}(\mathbb{Q}_{(\cdot)}, \mathbb{S}, \mathbb{F}) = \mathbb{F}\left(\text{Softmax}\left(\frac{\mathbb{Q}_{(\cdot)}^T \mathbb{S}}{\sqrt{d_s}}\right)\right)$$

- **Output**:
  - Enhanced feature vectors for the left and right hands: QL' and QR'.

REGRESSING HAND POSES

- **Input**:
  - Enhanced feature vectors QL' and QR'.
- **Process**:
  - **Regressor**:
    - A fully connected neural network that maps the enhanced features to 3D hand poses (joint positions).
  - **Loss Function**:
    - Combines multiple loss components to train the model effectively:
      - **Regression Loss**: Minimizes the error between predicted and ground truth joint positions.
      - **Temporal Consistency Loss**: Ensures smooth transitions between consecutive frames.
- **Output**:
  - Predicted 3D joint positions for the left and right hands.

GENERATING 3D HAND POSE PREDICTIONS

- **Input**:
  - Output of the regressor: Predicted 3D joint positions.
- **Process**:
  - The predicted joint positions are scaled and transformed to match the physical hand pose.
- **Output**:
  - Final 3D hand pose predictions:
    - Visualized as a set of 3D joint locations.
    - Enables applications like gesture recognition and virtual interaction.

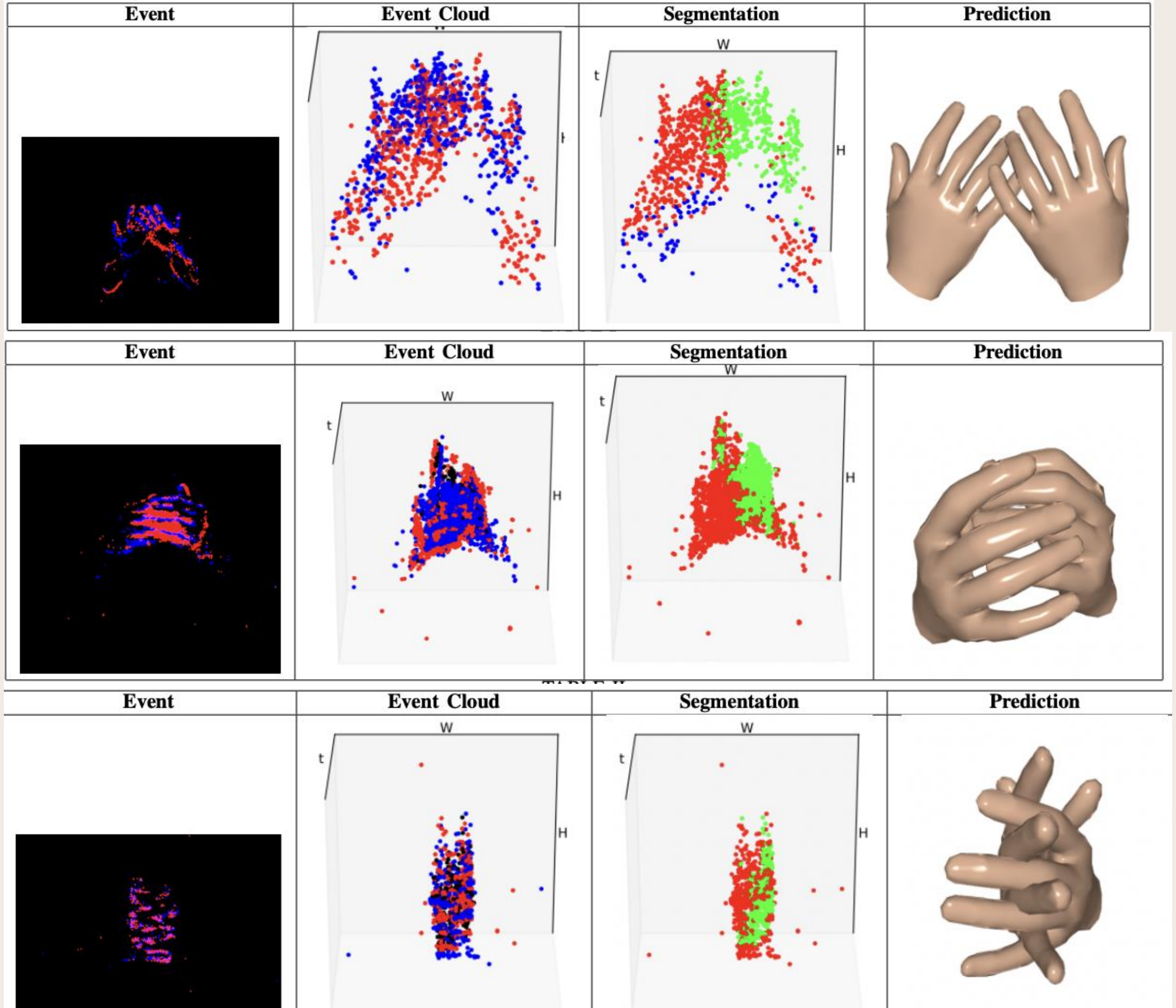
EXPERIMENTAL SETUP & DATASET COLLECTION

SETUP

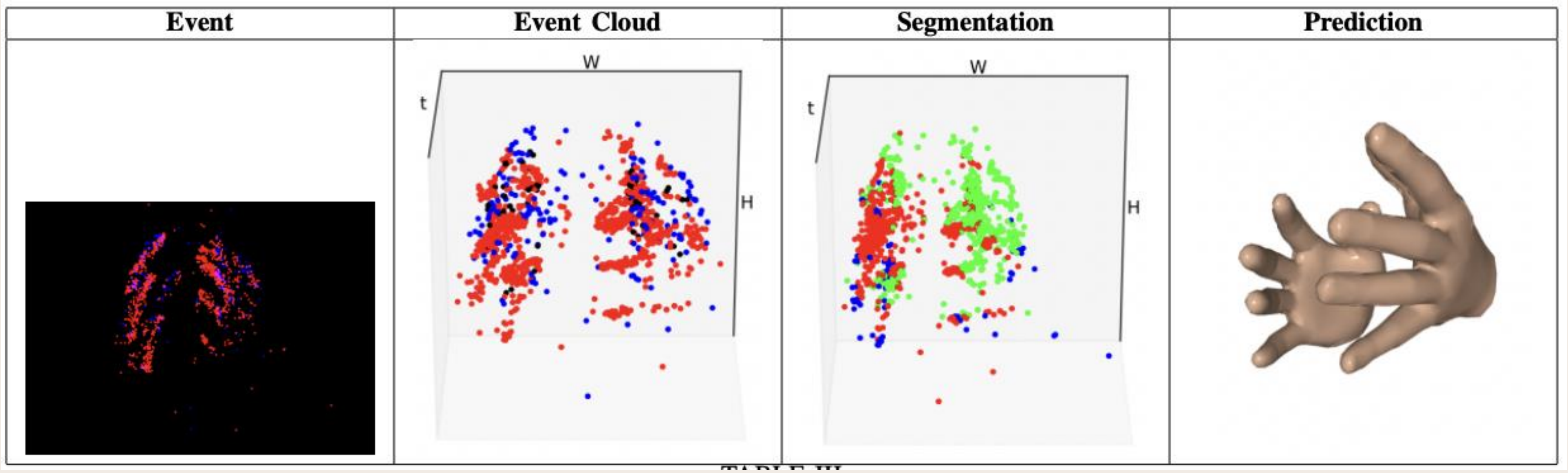
- The experimental setup consists of an event camera mounted securely on a tripod with a suction base for stability.
- The camera is connected to a laptop via a USB interface, facilitating data capture and processing in real-time.
- The output was given by the lab engineer Murad as ros bags
- Ros bags were used to be converted to mp4 videos

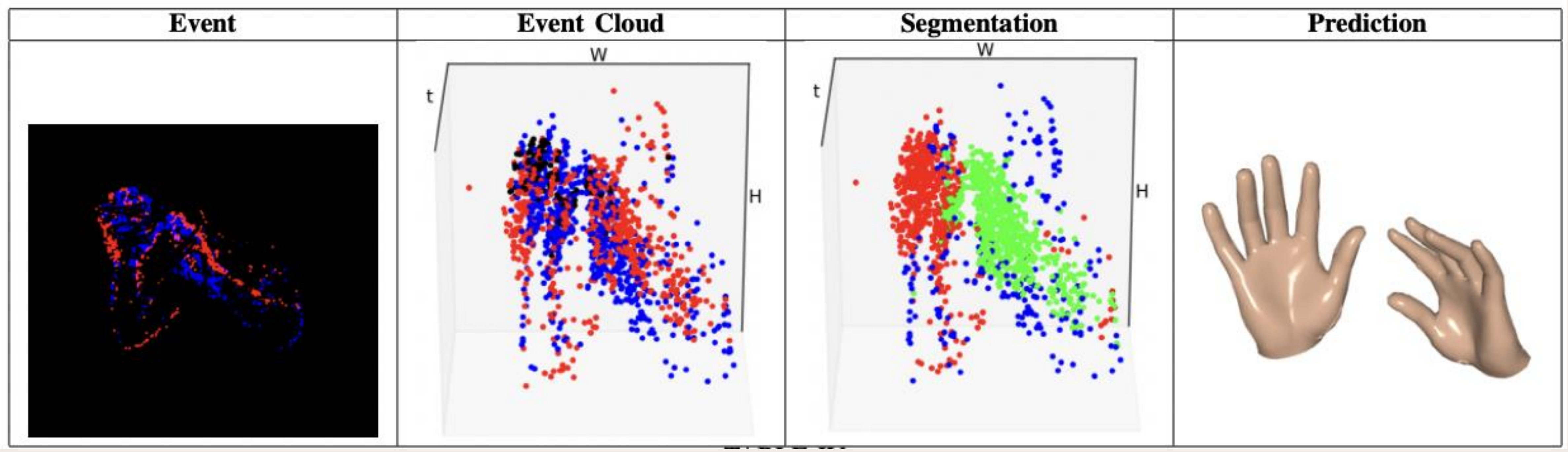
RESULTS

DIFFERENT HAND GESTURE EVENTS

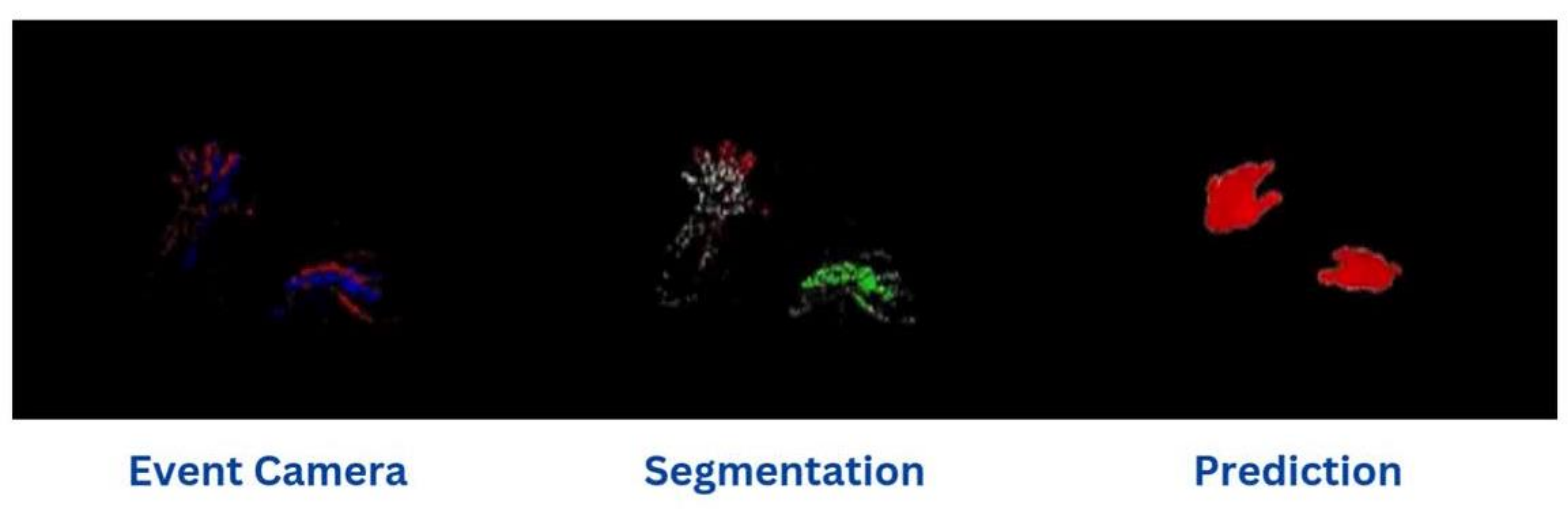


DIFFERENT HAND GESTURE EVENTS

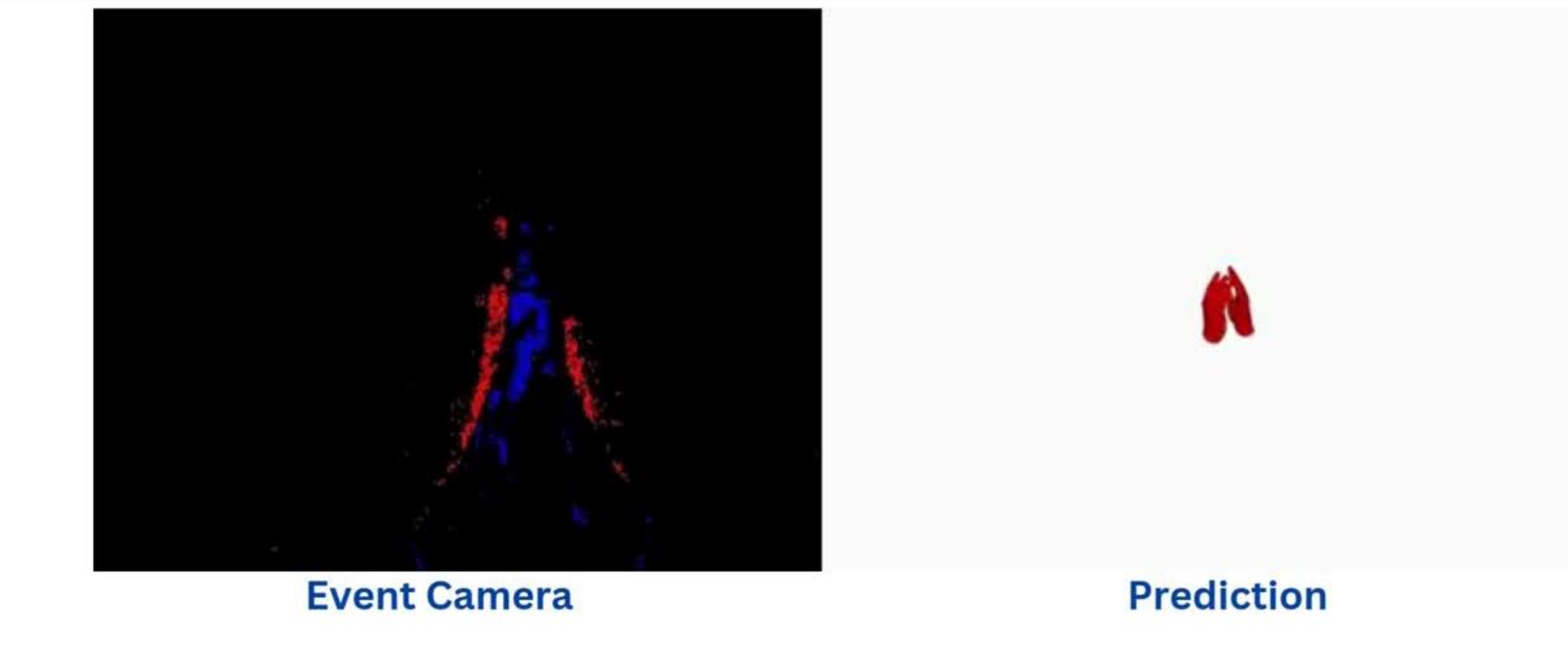




DEMO VIDEO 1



DEMO VIDEO 2



STRENGTHS

STRENGTHS

High Temporal Resolution

- Utilizes event cameras capable of capturing changes at microsecond intervals, making it highly effective for fast and dynamic hand gestures.

Robustness to Motion Blur

- Event cameras inherently avoid motion blur by capturing only changes, ensuring accurate hand pose estimation even during rapid hand movements.

Real-Time Performance

- The framework is optimized for low-latency applications, making it suitable for real-time hand gesture recognition

Handles complex scenarios

- Handles complex hand motions and interactions effectively, showcasing adaptability to real-world scenarios.

LIMITATIONS AND FUTURE WORK

LIMITATIONS AND FUTURE WORK

Fixed Camera Assmption

- The current approach assumes a stationary camera, which simplifies the problem but limits usability in dynamic or mobile setups.

Background Clutter

- Event data generated by moving objects or changes in the background can introduce noise, reducing segmentation accuracy.

Future work

Moving Camera Integration:

- Extend the framework to handle data from moving or portable event cameras, addressing challenges of background clutter and motion compensation.

RGB and Event Data Fusion:

- Combine event data with traditional RGB streams to leverage the strengths of both modalities:
  - Increased visual fidelity from RGB.
  - Low latency from event data.

THANK YOU !

REFERENCES

[1] Riccardo Mangiaracina, Alessandro Perego, Arianna Seghezzi, and Angela Tumino. Innovative solutions to increase last-mile delivery efficiency in b2c e-commerce: a literature review. International Journal of Physical Distribution & Logistics Management, 49(9):901–920, 2019

[2] Ming Li, Jian Weng, Anjia Yang, Wei Lu, Yue Zhang, Lin Hou, Jia-Nan Liu, Yang Xiang, and Robert H. Deng. Crowdbc: A blockchain-based decentralized framework for crowdsourcing. IEEE Transactions on Parallel and Distributed Systems, 30(6):1251–1266, 2019.

[3] Maha Kadadha, Hadi Otrok, Rabeb Mizouni, Shakti Singh, and Anis Ouali. Sensechain: A blockchain-based crowdsensing framework for multiple requesters and multiple workers. Future Generation Computer Systems, 105:650–664, 2020.

[4] Seth Larweh Kodjiku, Tao Han, Yili Fang, Esther Stacy E.B Aggrey, Collins Sey, Kwame O. Asamoah, Linda Delali Fiasam, Evans Aidoo, and Xun Wang. Wqcrowd: Secure blockchain-based crowdsourcing framework with multi-tier worker quality evaluation. Journal of King Saud University - Computer and Information Sciences, 35(10):101843, 2023.

[5] Zhifang Liao, Jincheng Ai, Shaoqiang Liu, Yan Zhang, and Shengzong Liu. Blockchain-based mobile crowdsourcing model with task security and task assignment. Expert Systems with Applications, 211:118526, 2023.

[6] Qiliang Yang, Tao Wang, Wenbo Zhang, Bo Yang, Yong Yu, Haiyu Li, Jingyi Wang, and Zirui Qiao. Privcrowd: A secure blockchain-based crowdsourcing framework with fine-grained worker selection. Wireless Communications and Mobile Computing, 2021(1):3758782, 2021