

Data-Driven Feature Tracking for Aerial Imagery

Michael Brady, Harshil Sharma, Harsh Patel, Vinay Chaudhari

Team 6, Deep Learning Final Project

github: https://github.com/xxender13/DL_Final_Project_Team6/tree/main

Abstract—This project develops a deep learning model to identify and track features between frames in aerial video sequences. By leveraging event cameras, we implement structure-from-motion (SfM) algorithms for accurate object pose estimation and 3D structure reconstruction. The proposed approach is evaluated on MultiFlow and EDS datasets, demonstrating robust performance under noisy conditions and refined accuracy through COLMAP optimization.

I. INTRODUCTION

Feature tracking in aerial imagery is critical for navigation, mapping, and object reconstruction. Traditional cameras face challenges such as high latency and computational inefficiency, particularly in dynamic environments. Event cameras offer an innovative solution, capturing asynchronous events with low latency and enabling robust tracking in real-time applications. This project aims to utilize event-driven data to enhance 3D modeling and pose estimation in aerial imagery.

II. RELATED WORK

Numerous studies have explored event cameras for high-speed vision tasks.

- **Low-Latency Automotive Vision with Event Cameras** [1]: Demonstrates reduced latency and improved object detection but faces challenges in data fusion.
- **Temporal Feature Markers for Event Cameras** [2]: Employs strobe LEDs for marker tracking, achieving high accuracy despite lighting issues.
- **BlinkTrack** [3]: Combines event and RGB data for high-frequency tracking using a differentiable Kalman filter.
- **Enhancing Robustness in Asynchronous Feature Tracking** [4]: Merges event and frame data for improved accuracy in dynamic scenes.
- **Data-driven Feature Tracking for Event Cameras** [5]: Introduces a frame attention module for robust feature tracking.

III. DATA

The datasets used include:

- **MultiFlow**: Provides asynchronous event streams, facilitating the generation of structured tracks for model training.
- **EDS Dataset**: Supplies pose data, enabling fine-tuning and validation of the trained model.
- **Augmented Lab Dataset**: Introduces varying levels of noise to test the model's resilience under degraded conditions.

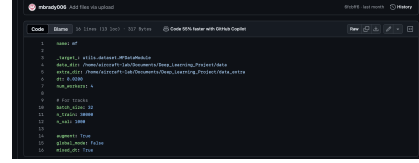


Fig. 1. MultiFlow Dataset structure and organization.

IV. METHODS

A. Event Representation and Track Generation

MultiFlow data was processed to generate event representations, capturing spatio-temporal features. Tracks were constructed to ensure temporal continuity and consistency.

```
508 v def on_validation_epoch_end(self):
509     if self.pose_mode is False:
510         # 1.2 event counts
511         with plt.style.context('ggplot'):
512             fig = plt.figure()
513             # Convert epe_12_hist to a numpy array of floats
514             self.epe_12_hist = np.array(self.epe_12_hist, dtype=float)
515             # counts
516             x, counts = np.unique(self.epe_12_hist, return_counts=True)
517             y = np.cumsum(counts) / np.sum(counts)
518             ax = fig.add_subplot()
519             ax.plot(x, y)
520             ax.set_xlabel('EPE (val)')
521             ax.set_ylabel('Proportion')
522             self.logger.experiment.add_figure(
523                 "12_cumsum/val", fig, self.global_step
524             )
525             plt.close("all")
526
527     # Convert epe_12_hist to float32 before logging the histogram
528     self.logger.experiment.add_histogram(
529         "EPE_hist/val", self.epe_12_hist.astype(np.float32), self.global_step
530     )
531
532     # Log metrics
533     self.log("EPE_median/val", np.median(self.epe_12_hist))
534     self.log("EPE_median/val", np.median(self.track_error_hist))
535     self.log("EPE_mean/val", np.mean(self.track_error_hist))
536     self.log("EPE_mean/val", np.mean(self.epe_12_hist))
537     self.log("EPE_std/val", np.std(self.track_error_hist))
538     self.log("EPE_std/val", np.std(self.epe_12_hist))
539     self.log("FA_median/val", np.median(self.feature_apr_hist))
```

Fig. 2. Template script for event-based data preprocessing.

B. Model Training

The generated tracks were used to train a deep learning model optimized for feature extraction and pose estimation. Hyperparameter tuning was conducted to balance precision and computational efficiency.

```
1  hydra:
2  run:
3  dir: training_runs/${data_name}/${model_name}/${experiment}/${now:%Y-%m-%d_%H%M%S}
4
5  # Composing nested config with default
6  experiment: open_source
7  train_name: whitestack_custom_v8
8
9  representation: time_surfaces_v2_8
10  epoch_size: 32
11
12  debug: False
13  n_iter: 2
14  logging: True
15
16  # Do not forget to set the learning rate for supervised or for pose finetuning in config/optimize.yaml
17  defaults:
18  - data: pose_data # (or, pose_edu, pose_rc)
19  - model: correlation_unscaled
20  - training: pose_finetuning_train_edu # (supervised_train, pose_finetuning_train_edu, pose_finetuning_train_rc)
21
22  # Pytorch lightning trainer's argument
23  trainer:
24  benchmark: True
25  log_every_n_steps: 8
26  max_epochs: 10000
27  num_processes: 1
28  num_sanity_val_steps: 1
```

Fig. 3. Training default settings for baseline experiments.

```

1 import logging
2 import os
3
4 import hydra
5 import pytorch_lightning as pl
6 import torch
7 import sys
8
9 sys.path.append('/home/aircraft-lab/Documents/Deep_Learning_Project/deep_ev_tracker/')
10
11 from utils.callbacks import IncreaseSequenceLengthCallback
12 from utils.utils import *
13 from utils.dataset import *
14
15 logger = logging.getLogger(__name__)

```

Fig. 4. Changes in training performance with varying parameters.

C. Pose Refinement with COLMAP

COLMAP was utilized to refine pose data, enhancing the accuracy of the structure-from-motion pipeline. These refinements significantly improved the robustness of the overall process.

D. Augmentation and Noise Testing

Augmentation methods were employed to introduce controlled noise for testing the robustness of the trained model. The script in Figure 5 outlines the augmentation logic.

```

237 def generate_corruption(input_folder, output_folder, severity):
238     for filename in os.listdir(input_folder):
239         if filename.endswith('.png') or filename.endswith('.jpg') or filename.endswith('.gif'):
240             image_path = os.path.join(input_folder, filename)
241             print(f'Processing {filename}...')
242
243             # Open the image
244             image = imageio.imread(image_path)
245             image = image.astype(float) / 255.0
246
247             # Apply all corruptions
248             for corruption_name, corruption_fn in corruptions:
249                 # Create a temporary folder for each corruption type
250                 corruption_folder = os.path.join(output_folder, corruption_name)
251                 if not os.path.exists(corruption_folder):
252                     os.makedirs(corruption_folder)
253
254                 # Apply only a single severity level (default is 1)
255                 corrupted_image = corruption_fn(image, severity)
256
257                 # Correct back to image and save
258                 corrected_image = np.clip(corrupted_image, 0, 255).astype(np.uint8)
259                 corrected_image = imageio.imwrite(os.path.join(corruption_folder, filename), corrected_image)
260
261             # Save the corrupted image in the appropriate folder
262             new_path = os.path.join(output_folder, filename)
263             corruption_folder = f'./{corruption_name}/{filename}'
264             corrected_image = imageio.imread(new_path)
265             print(f'Processed {filename}...')
266
267 if __name__ == '__main__':
268     # Specify input and output directories
269     input_folder = '/home/aircraft-lab/Documents/Deep_Learning_Project/01_Final_Project/Frame01_Final_Project/Frame01_000000_000000'
270     output_folder = '/home/aircraft-lab/Documents/Deep_Learning_Project/01_Final_Project/Frame01_Final_Project/Frame01_000000_000000'
271
272     # Generate and save corrupted images
273     generate_corruption(input_folder, output_folder, severity=1)

```

Fig. 5. Augmentation code for generating noisy datasets.

V. EXPERIMENTS AND RESULTS



Fig. 6. Predictions on original tracks.

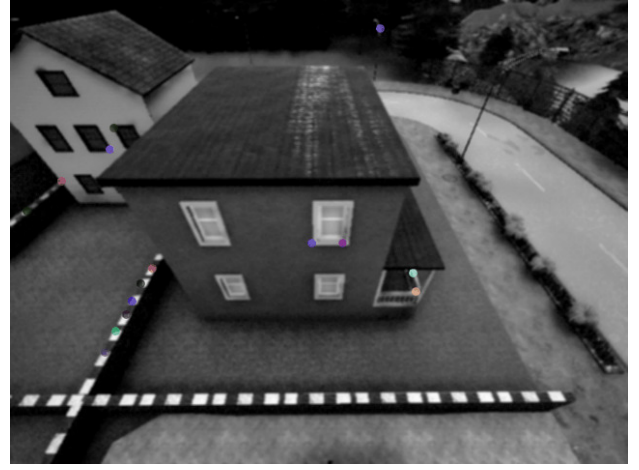


Fig. 7. Predictions on blurred tracks.

TABLE I
PERFORMANCE METRICS

| Condition | Feature Age | Expected Feature Age |
|---------------------|-------------|----------------------|
| Original Tracks | 0.0529 | 0.149 |
| Defocus Blur Tracks | 0.0521 | 0.146 |
| EDS Tracks | 0.576 | 0.472 |

VI. CONCLUSION

This project demonstrates the potential of event cameras for feature tracking and pose estimation in aerial imagery. By integrating MultiFlow tracks, COLMAP refinement, and noise-augmented testing, the proposed model can achieve robust performance under challenging conditions. With the test data, performance was limited due to the low frame rate of the simulated event camera data. A higher frame rate from the event camera may lead to better model performance. Future work will explore real-time applications and further noise-handling techniques.

REFERENCES

- [1] Low-Latency Automotive Vision with Event Cameras.
- [2] Temporal Feature Markers for Event Cameras.
- [3] BlinkTrack: Feature Tracking over 100 FPS via Events and Images.
- [4] Enhancing Robustness in Asynchronous Feature Tracking.
- [5] Data-driven Feature Tracking for Event Cameras.