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

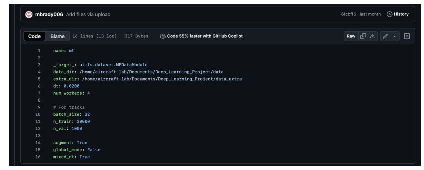
Numerous studies have explored event cameras for high-speed vision tasks.

- **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.



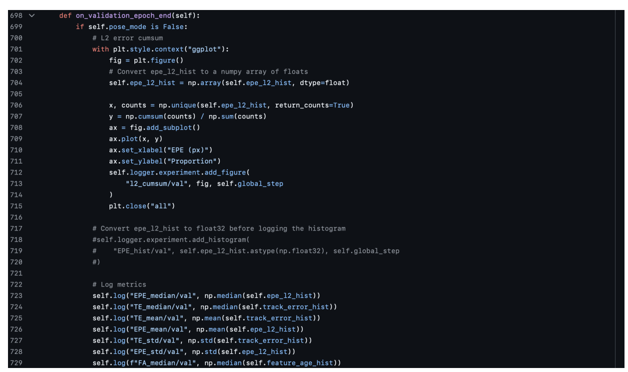

```yaml
name: mf

_target_: utils.dataset.MFDataModule
data_dir: /home/aircraft-lab/Documents/Deep_Learning_Project/data
extra_dir: /home/aircraft-lab/Documents/Deep_Learning_Project/data_extra
dt: 0.0200
num_workers: 4

# For tracks
batch_size: 32
n_train: 30000
n_val: 1000

augment: True
global_mode: False
mixed_dt: True
```

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.



```python
def on_validation_epoch_end(self):
    if self.pose_mode is False:
        # L2 error cumsum
        with plt.style.context("ggplot"):
            fig = plt.figure()
            # Convert epe_l2_hist to a numpy array of floats
            self.epe_l2_hist = np.array(self.epe_l2_hist, dtype=float)

            x, counts = np.unique(self.epe_l2_hist, return_counts=True)
            y = np.cumsum(counts) / np.sum(counts)
            ax = fig.add_subplot()
            ax.plot(x, y)
            ax.set_xlabel("EPE (px)")
            ax.set_ylabel("Proportion")
            self.logger.experiment.add_figure(
                "l2_cumsum/val", fig, self.global_step
            )
            plt.close("all")

        # Convert epe_l2_hist to float32 before logging the histogram
        #self.logger.experiment.add_histogram(
        #    "EPE_hist/val", self.epe_l2_hist.astype(np.float32), self.global_step
        #)

        # Log metrics
        self.log("EPE_median/val", np.median(self.epe_l2_hist))
        self.log("TE_median/val", np.median(self.track_error_hist))
        self.log("TE_mean/val", np.mean(self.track_error_hist))
        self.log("EPE_mean/val", np.mean(self.epe_l2_hist))
        self.log("TE_std/val", np.std(self.track_error_hist))
        self.log("EPE_std/val", np.std(self.epe_l2_hist))
        self.log(f"FA_median/val", np.median(self.feature_age_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.

```yaml
hydra:
  run:
    dir: training_runs/${data.name}/${model.name}/${experiment}/${now:%Y-%m-%d_%H%M%S}

# Composing nested config with default
experiment: open_source
track_name: shitomasi_custom_v5

representation: time_surfaces_v2_5
patch_size: 31

debug: False
n_vis: 2
logging: True

# Do not forget to set the learning rate for supervised or for pose finetuning in configs/optim/adam.yaml
defaults:
  - data: pose_eds # [mf, pose_eds, pose_ec]
  - model: correlation3_unscaled
  - training: pose_finetuning_train_eds # [supervised_train, pose_finetuning_train_ec, pose_finetuning_train_eds]

# Pytorch lightning trainer's argument
trainer:
  benchmark: True
  log_every_n_steps: 8
  max_epochs: 40000
  num_processes: 1
  num_sanity_val_steps: 1
```

Fig. 3. Training default settings for baseline experiments.

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.

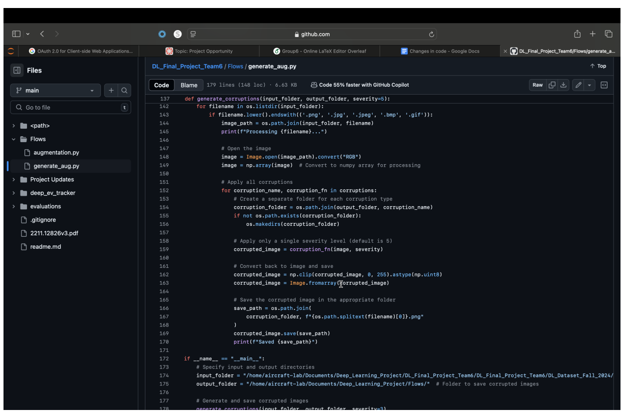


Fig. 5. Augmentation code for generating noisy datasets.

V. Experiments and Results

Fig. 6. Predictions on original tracks.

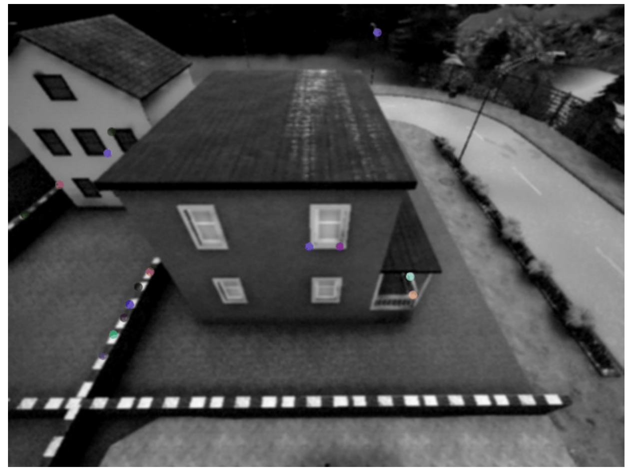


Fig. 7. Predictions on blurred tracks.

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.