# II. Trajectory Data Collection Methodology

In the experiments, we applied the proposed vehicle trajectory extraction method to 70 minutes of drone videos. The video data were collected by two 4K drone cameras from 6:05–7:15 p.m. (from dusk to night) on Wednesday (November 16, 2022) over an 800 ft long segment of Park St in Madison, Wisconsin. A total of 4946 vehicle records were recorded. The period of our recording also includes the formation and dissipation of the evening peak.

The roadway includes two bus lanes (depicted in green in Fig). In pursuit of a more accurate vehicle simulation, vehicles are classified into two categories, 'large' and 'small,' based on their respective lengths. While these categories employ the same calibration model, the parameters within the model vary accordingly. The bus lanes on University Ave and W Johnson St were in use at the time of recording.

A aerial view of a road intersection

Description automatically generated

**Fig 1** The study area along Park St

The vehicle trajectory extraction and cleaning process mainly consists of two parts. The first part is Vehicle trajectory extraction. With the given video taken by different drones, Step 1 stabilizes all frames by matching the feature points, Then, Step 2 merges the frames from different drones and gets the full scope of the target range. Step 3 detects and tracks the vehicle using locally trained YOLOv8 [8] and DeepSORT [9], then gets the initial trajectory data of all vehicles in pixel coordinate and transfers the coordinate trajectory to Universal Transverse Mercator (UTM) Grid System coordinate trajectory.

The second part is trajectory data processing, Step 1 removes position offsets and then smooth the trajectory, and Step 2 calculates the vehicle speed, acceleration, and position. In the end, the method outputs the extracted trajectory dataset.

## A. Trajectory Extraction

This section describes how we extract vehicle trajectories from the recorded videos using drones.

### Step 1: Frame Stabilize

During drone flights, inevitable camera displacements lead to video shaking and perspective changes. To extract vehicle trajectories under a unified coordinate system, all frames in the drone-captured video must first be stabilized to align within a consistent reference frame. We utilized the Speeded Up Robust Features (SURF) to extract feature points [10] and Fast Library for Approximate Nearest Neighbors (FLANN) feature points matching algorithm [11] to match feature points. These matched points were then used to compute the relative position relationship between frames, transforming them into a unified coordinate system.

Two primary issues commonly arise in this process: First, the position and altitude changes of the drone impact the camera's angle, causing discrepancies in the relationship between feature points. Second, dynamic lighting conditions, such as sunlight, streetlights, and vehicle headlights, interfere with feature point extraction, especially under low-light or rapidly changing illumination scenarios.

To address these challenges, we implemented a dynamic reference frame adaptation strategy combined with a lighting mask scheme. The dynamic reference frame adaptation strategy dynamically updates the reference frame during processing to mitigate cumulative errors and account for lighting changes. Specifically, when lighting differences between the current processed frame and the reference frame reach a predefined threshold, or when the frame-to-frame mean squared error exceeds acceptable limits, a new reference frame is selected. The new reference frame is chosen from previously stabilized frames to ensure compatibility with the current frame’s conditions, allowing for more reliable alignment. Compared to traditional methods, this strategy reduces MSE by 22.7% and improves feature point matching rates by up to 18%.

To further mitigate interference from non-road lighting sources, we developed a lighting mask scheme. Using the DeepLabv3+ segmentation model, non-road regions were dynamically identified and masked by converting these areas to black. This process effectively suppresses the influence of light sources such as streetlights and building lights on feature point extraction. The combination of dynamic reference frame adaptation and lighting masking significantly enhances the robustness and accuracy of trajectory extraction across diverse environments.

This unified framework addresses the two common challenges—drone displacements and lighting variations，ensures applicability to a wide range of scenarios.

### Step 2: Frames Merging

To merge videos recorded by different drone cameras, we utilize the mapping relationships of feature points within each drone's field of view. As illustrated in Figure 2, this process creates a more extensive image, facilitating subsequent trajectory extraction.

A screenshot of a computer screen

Description automatically generated

**Fig 2** Illustration of the frame merging process

Initially, each camera requires calibration to establish the transformation relationship between pixel coordinates and UTM coordinates , which are obtained from GPS data. This relationship between these two coordinate systems is , where is the camera intrinsic matrix, is the rotation matrix, is the translation vector, and is a scaling factor.

The transformation process from UTM coordination to Pixel coordination is: , where represents the pixel coordinates of the feature points of Camera . represents the actual UTM coordinates of the feature points of Camera . represents the mapping relationship from UTM coordinates to pixel coordinates of Camera . Next, with Camera as the reference coordinate system, the pixel coordinates of the feature points within the field of view of other cameras in the pixel coordinate system of Camera are , where is the pixel coordinates of the feature points within the field of view of Camera , transformed into the same pixel coordinate system as Camera based on the mapping relationship from UTM coordinates to pixel coordinates of Camera . Lastly, using and , we transform images from other cameras through projection to overlap with the image of Camera , thus accomplishing the stitching process.

In this study, we merge two video sets captured by two drones with overlapping fields of view. We manually selected 20 feature points verified their GPS coordinates using Google Maps, and transformed them into the UTM coordinate system. The chosen points were all situated on flat ground with negligible height differences. Among the 20 feature points, three are within the overlapping area of the two cameras' fields of view. These points served as benchmarks to validate the accuracy of the transformation process. The discrepancy between the pixel coordinates of the three feature points in the reference system of Drone 1 and their projected counterparts from Drone 2 was a mere 0.003%. This low error rate demonstrates the robustness and precision of our frame-merging process, ensuring its applicability to high-resolution drone-captured traffic videos.

### Step 3: Vehicle Detection and Tracking

To efficiently detect and track vehicles in drone-captured videos, we adopted the YOLOv8 [8] with a customized detection head tailored specifically for small object detection. This detection head has been optimized by retaining only the layers designed for small-scale object recognition while removing layers targeting large-scale objects. Compared to SSD-based approaches, which integrate additional deconvolution and feature fusion modules to enhance small object detection [12], YOLO-based solution achieves superior computational efficiency by directly simplifying the detection head structure. This modification not only reduces computational overhead but also enhances detection accuracy for small objects like vehicles in UAV videos.

Notably, the optimized detection head is designed to be modular and compatible with various versions of YOLO, including YOLOv10, and future iterations, ensuring long-term adaptability and scalability.

To further address the challenges posed by elongated objects or those with aspect ratio imbalances, a slicing mechanism was integrated into the detection pipeline. This mechanism divides each input frame into overlapping tiles, ensuring that small objects occupy a proportionally larger area within the cropped regions. By processing these tiles independently and merging the results post-detection, the method effectively improves the model’s sensitivity to small-scale vehicles.

Then, the DeepSORT algorithm [9] is employed to track vehicles across frames. DeepSORT is a state-of-the-art tracking algorithm that associates detections across frames using a combination of appearance features and motion cues. Specifically, DeepSORT assigns a unique ID to each detected bounding box and tracks the movement of the bounding boxes over time. After obtaining the pixel coordinates, according to the calibration results of the reference camera in Step 2, the UTM coordinates are reversed, and the preliminary vehicle trajectory is obtained.

## B. Trajectory Data Processing

### Step 1: Data Cleaning

Although the vehicle detection and tracking methods proposed earlier achieve high precision and recall, some inevitable issues remain due to the inherent challenges of drone-captured traffic videos. Specifically, **drifting points** can occur because of tracking inaccuracies, occlusions, or rapid vehicle movements in dynamic environments.

To identify drifting points, we employed a vector-based angular filtering approach. By analyzing the angle between consecutive direction vectors, and , where is the direction vector. We can precisely detect abrupt changes in trajectory caused by noise. Unlike traditional threshold-based outlier detection, our method dynamically adapts to local trajectory characteristics, ensuring robust detection even under noisy conditions. Based on the practical considerations, points that contribute to an angular shape within 30 degrees are removed. The removed points are reconstructed using the moving average method.

The second step addresses trajectory smoothing to correct for noise and ensure a more natural vehicle path. Several widely applied smoothing methods were adopted, including Kalman filtering, Spline smoothing, and moving average techniques. The 5-point moving average method stand out for two reasons. First, the implementation of the 5-point moving average is relatively straightforward and does not require complex parameter calibration, reducing the difficulty and uncertainty of preprocessing. Moreover, this method demonstrated significant advantages in processing speed, allowing for rapid smoothing of data points, which is crucial for applications involving large datasets. By calculating the average position within a sliding window of five data points and applying it to each data point within the trajectory, the 5-point moving average effectively reduces random noise while maintaining the overall shape and characteristics of the trajectory, shown in Figure 3.

A graph of a cleaning procedure

Description automatically generated with medium confidence

Fig 3 Drifting points along parts of a trajectory.

### Step 2: Determining Additional Characteristics

To facilitate subsequent calculations, we augmented the data with vehicle speed, acceleration, and lane information, shown in TABLE II. We employ the filtered trajectory data to compute the vehicle's speed and acceleration, which are respectively the first and second derivatives of the position to time. Vehicles are assigned to their respective lanes based on the position of their center point. Our proposed data adopt a naming convention akin to those utilized in simulation software such as SUMO and VISSIM. Intersections are denoted as "nodes," and the connecting roads are termed "edges." Each "edge" is subdivided into various "lanes." During our data processing, the position data for each vehicle is stored under two features. The first feature refers to the "edge" on which the vehicle is currently traveling, encompassing all lanes and intersections present on that "edge." The second feature specifies the exact lane occupied by the vehicle. If the vehicle is located within an intersection, the lane feature defaults to 0.