Drone Map Creation and Vehicle Tracking on the Map

Oussama Jaffal

Semester Project

January 2025

Responsible
Prof. Nikolaos Geroliminis
EPFL / LUTS

Supervisor
Yura Tak
EPFL / LUTS

Urban Transport Systems Laboratory (LUTS)

Contents

1 Introduction **3**

2 Literature Review **4**
2.1 Image Stitching 4
2.2 Vehicle Tracking 4
2.3 Road Inpainting 4

3 Methodology **5**
3.1 Map Creation 5
3.1.1 Image Selection 5
3.1.2 Two Images Stitching 6
3.1.3 Multiple Image Stitching: Adding the Next Frame 8
3.1.4 Challenges in Stitching 11
3.2 Vehicle Tracking 12
3.2.1 Vehicle Tracking from Drone Video Footage 12
3.2.2 Mapping Tracking Trajectories to the Final Map 13
3.2.3 Post-Processing of the Computed Trajectories 15
3.2.4 Challenges in Vehicle Tracking 16

4 Results and Discussion **17**
4.1 Map Generation 17
4.1.1 Two Way Stitching vs One Way Stitching 17
4.1.2 Critical Assessment of the Generated Map 18
4.2 Vehicle Tracking 21
4.2.1 Visualization of the Mapped Trajectories 21
4.2.2 Vehicle Tracking Assessment 22

5 Conclusion **22**

1 Introduction

The ability to map road networks and monitor vehicle movements is critical for urban planning, traffic management, and the development of autonomous navigation systems. Recent advancements in drone technology and computer vision have made it possible to capture detailed aerial imagery and extract valuable information from it. This project aims to utilize bird's-eye view drone footage to create a high-resolution map of a scanned area and track vehicle trajectories within it.

The workflow involves stitching consecutive frames from drone footage to generate a seamless map of the observed region. Vehicle detection and tracking algorithms are then applied to identify and monitor vehicles in the footage, with their trajectories overlaid on the generated map. The integrated system provides a comprehensive visualization of road activity, offering insights into traffic flow and vehicle behavior.

This report details the methodology, challenges, and results of this project. The proposed approach demonstrates the feasibility and reliability of using drone-based systems for road mapping and traffic analysis, with potential applications in smart city initiatives and transportation research.

2 Literature Review

This section reviews the foundational methodologies and state-of-the-art tools used in image stitching, vehicle tracking, and image inpainting, which were integrated into the project workflow.

2.1 Image Stitching

Image stitching is a critical task in computer vision, enabling the creation of panoramic views or large-scale maps from overlapping images. Traditional approaches, such as those based on the Scale-Invariant Feature Transform (SIFT) [1], rely on detecting and matching keypoints between images, followed by homography estimation to align them. While effective, these methods often struggle in the presence of noise, motion blur, or varying illumination conditions.

LightGlue, a modern feature-matching library, builds upon these traditional methods by leveraging neural networks to dynamically refine matches between frames [2]. Unlike static feature-matching techniques, LightGlue uses transformer-based architectures to improve robustness against challenging environmental factors, making it particularly suitable for drone footage. However, prior work emphasizes the need for preprocessing steps, such as ensuring sufficient frame overlap, to avoid stitching artifacts and avoiding images with motion blur [3]. This project implemented these recommendations by filtering frames with inadequate overlap before applying LightGlue.

2.2 Vehicle Tracking

Vehicle tracking has been extensively studied within the domain of multi-object tracking (MOT). Early MOT frameworks, such as SORT [4], utilize simple motion models for tracking objects detected in consecutive frames. More advanced approaches, such as DeepSORT [5], incorporate appearance-based features for re-identification. However, these methods often struggle in scenarios involving occlusion or dynamic environments.

BoT-SORT [6] represents a significant advancement in MOT by combining detection-based tracking with appearance descriptors for re-identification and bounding box refinement. This makes it particularly effective in tracking vehicles in complex urban environments captured by drones. Prior research highlights its robustness in high-density traffic scenarios and its ability to maintain track consistency despite frequent occlusions. In this project, BoT-SORT was applied directly to drone video footage, and its outputs were mapped onto the stitched map for trajectory visualization.

2.3 Road Inpainting

Image inpainting has evolved significantly, with deep learning-based methods outperforming traditional patch-based approaches. Simple LaMa, a framework based on LaMa [7], is a lightweight yet powerful inpainting model that employs convolutional neural networks to fill occluded or removed regions in images. Unlike earlier techniques that rely on manual intervention, Simple LaMa ensures context-aware reconstruction with minimal supervision, making it ideal for applications such as removing moving objects from aerial imagery.

In this project, Simple LaMa was used to remove vehicles from selected drone frames before stitching, ensuring that the final map appears clean and free of visual clutter.

3 Methodology

3.1 Map Creation

The image stitching process involves organizing a sequence of images and aligning each one to ensure a smooth transition between frames. Each image is adjusted in terms of position and angle before being combined with the previously stitched images. This process continues iteratively, with each new image merging into the growing stitched map. Once all images have been processed, the final map is created and saved as a single seamless image, ensuring a continuous and coherent representation of the observed area.

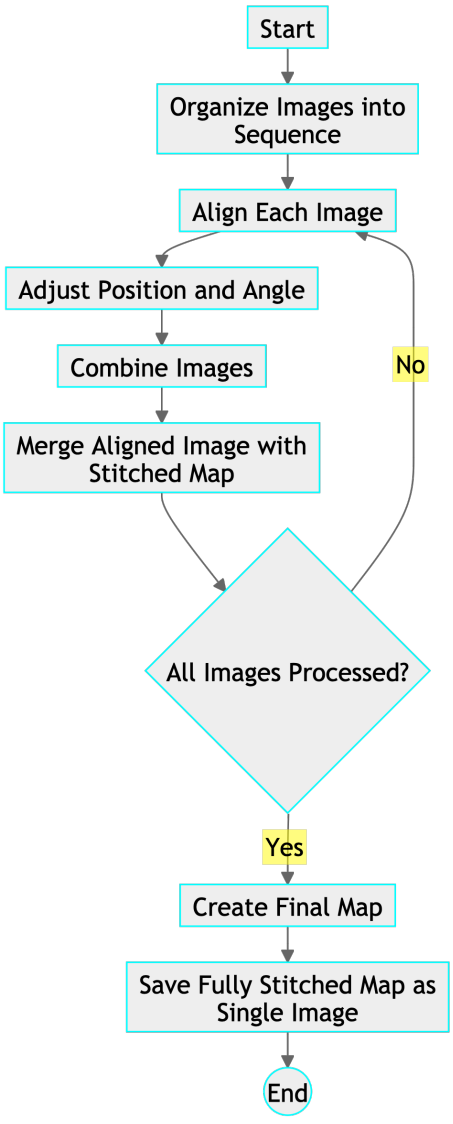

Figure 1: Diagram for Map Creation

3.1.1 Image Selection

In this step, frames are selected from the original drone video footage based on GPS coordinates extracted from the `DJI_0763.SRT` file. The process begins by calculating the distance in the real world between each next frame of the video footage and the first frame of the video. Once the

accumulated distance reaches a predefined threshold, the next candidate frame is checked for blurriness. If the frame is sufficiently clear, it is selected; otherwise, it is discarded. This process continues iteratively between the last selected frame and the potential next frame until all frames in the video have been evaluated, ensuring that only relevant and high-quality images are chosen for further processing.

3.1.2 Two Images Stitching

The image stitching process involves feature extraction and matching between two selected frames using the LightGlue library. The key features are detected in both frames and the corresponding points are matched to establish a transformation between them. This allows for accurate alignment of images, ensuring seamless stitching of consecutive frames.

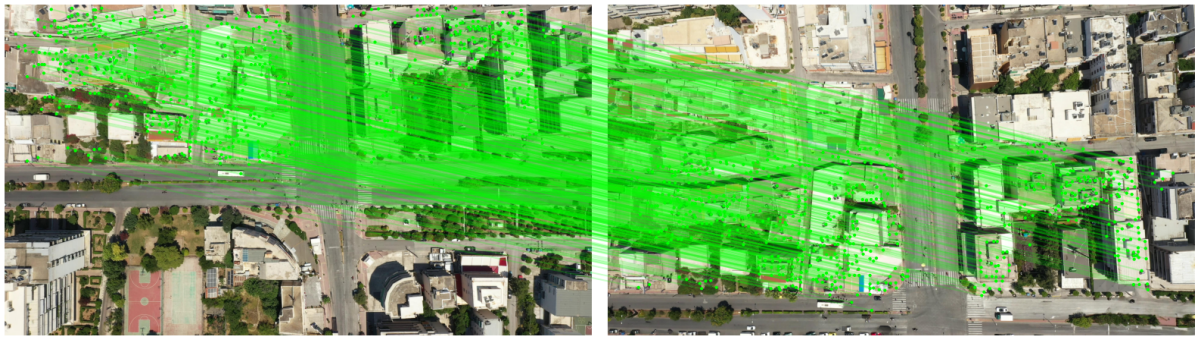

Figure 2: Feature Matching using LightGlue library

The resultant image is created by combining the two transformed images using the **RANSAC** method to compute the necessary transformation. However, due to geometric transformations, the transformed image may extend beyond the original frame, requiring padding to accommodate both images within a unified coordinate system, ensuring that both images fit into a common canvas by following these steps:

1. **Extract Image Dimensions:** The function retrieves the width and height of both input images (source and destination).
2. **Compute Transformed Corners:**
    - The four corners of the source image are defined.
    - These corners are transformed using the **homography matrix** to determine their new positions in the stitched space.
    - The destination image's corners are also included to determine the full bounding area needed for both images.
3. **Determine the Bounding Box:**
    - The minimum and maximum (x, y) coordinates from both images are computed to define the **bounding box** that can contain both images without cropping.
    - If any of the transformed coordinates are negative, a **translation shift** is applied to ensure all pixels are in the positive coordinate space.

4. **Apply Translation & Warping:**
    - A **translation matrix** is constructed to shift both images so that they fit within the bounding box.
    - The **source image** is warped using the updated transformation matrix that includes translation.
    - The **destination image** is warped separately using only the translation matrix, ensuring it remains in its original form but aligned with the new padded space.

Figure 3: Source Image

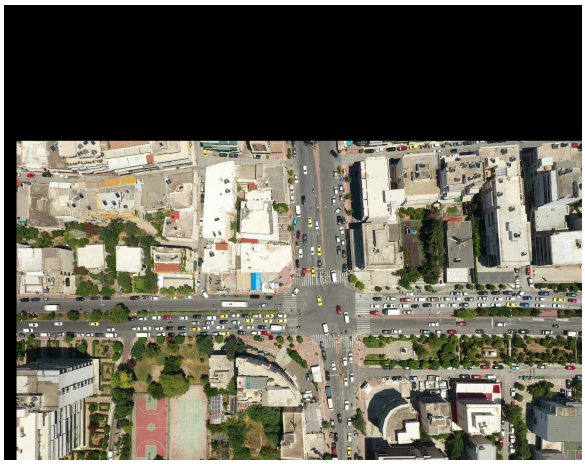

Figure 4: Destination Image

5. **Merge the source and destination images** The final step is to merge the two images by creating a mask of non zero value pixels in the transformed source image and apply that mask on the destination image.

Figure 5: Merging two selected frames

This process ensures that both images align correctly in the final output while preventing any part of the transformed image from being cropped due to negative coordinates.

3.1.3 Multiple Image Stitching: Adding the Next Frame

Instead of matching each new frame to the entire panorama, the code matches it *only* to the last transformed frame (the most recently added image). Once the new frame is aligned with this "last frame," it is integrated into the panorama through a series of carefully applied transformations and coordinate updates. Below is a step-by-step outline of how this process works, including the usage of the `warp_perspective_padded` function and the involved matrices.

1. **Match Features with the Last Transformed Frame.**
   - Let I_n be the "last transformed frame" already placed in the panorama, and I_{n+1} be the new incoming frame.
   - A feature extractor and matcher identify corresponding keypoints between I_n and I_{n+1}.
   - This localized matching (rather than matching I_{n+1} with the full panorama) is drastically more accurate, because I_n overlaps with I_{n+1} in the region of interest.
2. **Compute the Homography H.**
   - Using matched keypoints (x_n, y_n) in I_n and (x_{n+1}, y_{n+1}) in I_{n+1},

$$H = \text{cv.findHomography}(\,(x_{n+1}, y_{n+1}),\ (x_n, y_n)\,, \text{RANSAC}).$$

   - This homography H maps pixel coordinates from I_{n+1} directly to the coordinate system of I_n.
   - If an in-plane rotation is also tested beforehand, the best rotation angle is chosen, and the image is pre-rotated before homography estimation.
3. **Chain of Transformations.**
   - Recall that I_n itself was previously mapped into the *panorama coordinate system* via an earlier transform matrix T_n.
   - Thus, to position I_{n+1} into the *same* global coordinates, one can conceptually combine H with T_n (e.g., $T_{n+1} = T_n \cdot H$).
   - The code accomplishes this by updating corner coordinates and applying translations to keep track of each frame's placement in the growing panorama.
4. `warp_perspective_padded` **Function.**
   - Given an image I and a 3×3 transform matrix M, `warp_perspective_padded` warps I onto a *larger padded canvas* so that no part of the warped image is cut off, even if the transformed corners become negative or extend beyond image bounds.
   - It returns:
     - `dst_padded`: The padded background image (e.g., the existing panorama) large enough to accommodate the newly warped frame.
     - `warped_image`: The image I *after* applying M.
     - `anchorX`, `anchorY`: Offsets that indicate how far I had to be shifted on the padded canvas.

- These offsets and the resulting `warped_image` are crucial for *correctly* overlaying the new frame onto the panorama.

Figure 6: Merging second and third selected frames

5. **Applying Translations and Updating Corners.**

   - After warping I_{n+1}, the code typically applies an additional *translation matrix* to shift the entire stitched panorama if the bounding region becomes too large or negative indices occur.
   - The corner coordinates for each frame (stored in `image_corners_df`) are updated by:

$$\texttt{corners}_{I_{n+1}} \leftarrow \text{cv.perspectiveTransform}(\texttt{corners},\ H)\ +\ [\texttt{anchorX}, \texttt{anchorY}].$$

   - By keeping a running log of these corners, the system knows where each image resides in the panorama coordinate system.

Figure 7: Source Image

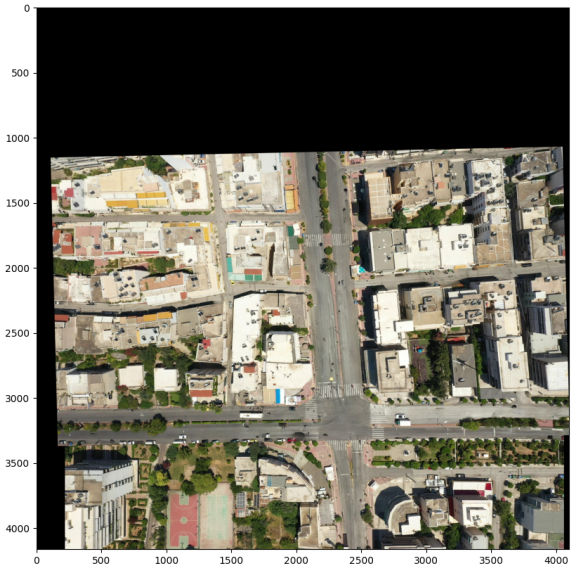

Figure 8: Destination Image

6. **Overlaying the New Warp onto the Panorama.**

   - A simple masking step fills in the non-zero (valid) pixels of the newly *warped* I_{n+1} onto the padded destination.
   - This integrated result is saved (e.g., as `aligned_image.jpg`), serving as the basis (anchor) for the next iteration.

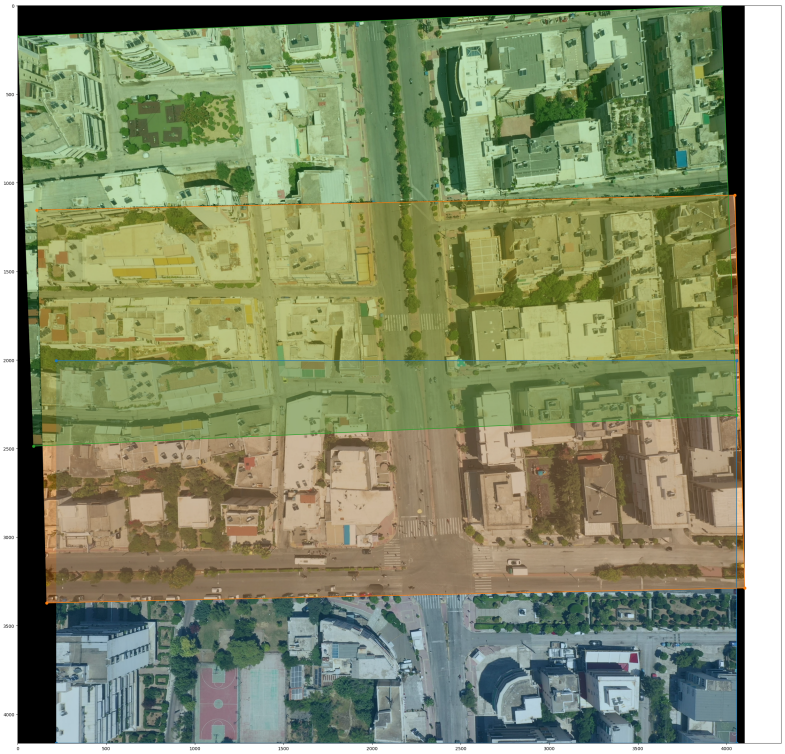

Figure 9: Merging three selected frames

In Summary: *Partial* feature matching against only the last transformed frame is used for efficiency and clarity. The homography H aligns the new frame with the coordinates of the last frame, and through accumulated transforms, the new frame is placed correctly in the global panorama space. The `warp_perspective_padded` function ensures no content is lost during warping, while final translations adjust for any offset in the global coordinate system. Finally, updated corner coordinates keep track of each image's position, enabling accurate and iterative expansion of the panorama.

3.1.4 Challenges in Stitching

Rotated Frames One major issue in our method was that when rotating the selected frames to align them with the previous selected frame, a portion of the image data was inevitably cropped out, which meant some of the data will be missing in the final map.
This problem only became apparent later in the project when the final map was successfully generated. An attempt to perform rotation without cropping failed to have satisfactory stitching results, primarily because the image sizes varied from frame to frame. The algorithm relied on having consistent dimensions for all frames, and the inconsistency hindered proper alignment.

Modified Homography Matrice In our method, we modify the homography matrix by retaining only its first two rows and fixing the third row to $[0\,0\,1]$. In a standard homography, the third row may introduce a scaling factor, which can distort the transformed images. However, because the images used in our stitching pipeline all have the same dimensions, any rescaling is unnecessary. By setting the last row to $[0\,0\,1]$, we ensure that no additional scaling or distortion occurs during the transformation. This results in more accurate alignment and a better stitched panorama compared to using the unmodified homography matrix.

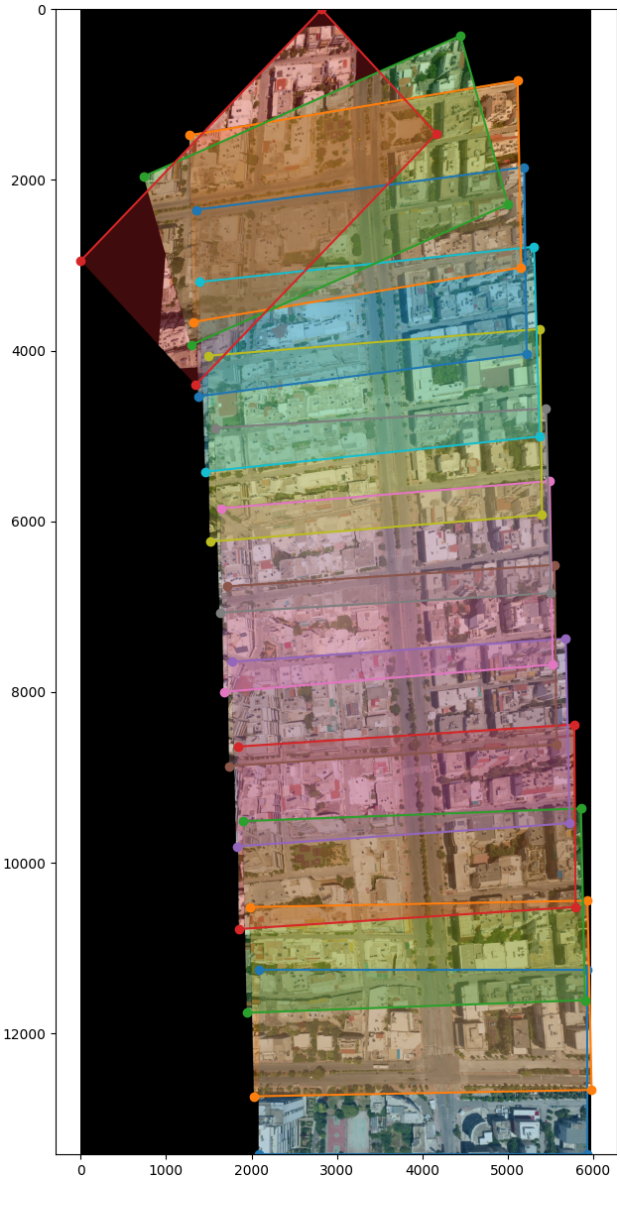

Figure 10: Stitching with Modified Homography

Figure 11: Stitching with Unmodified Homography

Comparison of homography approaches: The modified homography (left) eliminates rescaling and distortion by fixing the last row to $[0\,0\,1]$, while the unmodified version (right) can introduce distortions due to scaling factors.

3.2 Vehicle Tracking

3.2.1 Vehicle Tracking from Drone Video Footage

In our approach, we perform vehicle tracking on aerial footage captured by a drone, offering a bird's-eye view of the scene. The tracking module leverages the BotSort algorithm, which

is initialized using a pre-trained ReID model. The main steps of the tracking procedure are as follows:

1. **Video and Detection Input:** For each frame in the input video, the corresponding detection file (generated by a vehicle detection module) is read. These files contain the coordinates of the detected vehicles in the form of bounding boxes along with a confidence score.

2. **Tracker Update and Visualization:** The detections are converted into the required format and passed to BotSort, which associates detections across frames to produce unique track IDs. The tracking results are overlaid on the current video frame to visualize the trajectories.

3. **Result Storage:** For every frame, the tracker saves the results in a text file. Each file is named using the frame index (e.g., `track_fr_0001.txt`) and contains, for each tracked vehicle, the following information:

    - **Track ID:** A unique identifier assigned to the tracked vehicle.
    - **Bounding Box Center Coordinates:** The computed (x, y) coordinates representing the center of the detection.
    - **Detection Confidence:** Although not used in the current application, the confidence score (obtained via matching detections with the tracker's output using an IoU metric) is included for potential future use. Confidence score generated by the BotSort module is also possible, however not implemented here.

This modular design enables a clear separation between detection and tracking, and the inclusion of the detection confidence facilitates further refinement of the tracking results in future applications.

3.2.2 Mapping Tracking Trajectories to the Final Map

After obtaining frame-level tracking results, the next step is to map these trajectories onto the final panoramic map. Note that the process for generating the panorama was detailed in 3.1; here we focus on the transformation of tracking data into the map's coordinate system.

Due to the large size of the panorama and the suboptimal performance of feature matching on full-scale images, we adopt a batch-based approach. The key steps are as follows:

1. **Batch Selection:** Frames are grouped into batches defined by two selected reference frames. These reference frames have associated corner coordinates (stored in a DataFrame) that were used in the panorama creation process.

2. **Cropped Map Extraction:** For each batch, the minimum and maximum coordinates of the two reference frames are used to extract a cropped region from the final map. This localized region is more manageable for robust feature matching.

Figure 12: Exctracted Zone of the two selected frames

3. **Frame Processing:** Within each batch, frames are selected at regular intervals (e.g., every 10th frame). A temporary folder is created for each batch.

4. **Feature Matching and Homography Calculation:** Each selected frame is rotated to align with the cropped map region, and feature matching is performed using a feature extractor and matcher (e.g., LightGlue). A homography transformation is then computed to map the frame coordinates to the cropped map.

Figure 13: Feature Matching of the Cropped Zone

5. **Global Coordinate Mapping:** Once the local transformation is determined, an additional translation—based on the minimum x and y values of the selected corner coordinates—is applied. This step transforms the local coordinates into the global coordinates of the full panorama.

6. **Trajectory Assembly:** The process is repeated for all batches, and the resulting vehicle trajectories are aggregated and stored (e.g., in a pickle file) for further analysis.

This two-stage transformation (first rotating and aligning with a cropped map, then translating into the global panorama) addresses the challenges of feature matching on large images and improves the accuracy of mapping vehicle trajectories from the drone video to the final panoramic map.

Comment : For computation reasons, the same rotation is applied to all frames within a batch. During the map generation phase, we precompute and store the optimal rotation angle for each frame added to the final map. When processing a batch, we use the recorded rotation angle of the reference frames to transform all frames within that batch accordingly. This guarantees that all frames in a batch undergo the same rotation, preventing inconsistencies and improving computation time.

3.2.3 Post-Processing of the Computed Trajectories

After obtaining the vehicle trajectories from the tracking stage, a post-processing step is performed to refine and classify these trajectories based on their movement characteristics. This post-processing involves three main computations:

- **Computation of Turn Angles:** For each trajectory, the angles between consecutive segments are computed. This is done by determining the angle (in degrees) between successive movement vectors. The resulting turn angles help identify abrupt directional changes in the trajectory.
- **Calculation of Bounding Radius:** The spatial extent of each trajectory is quantified by computing its bounding radius. First, the centroid (i.e., the average position) of all points in a trajectory is calculated. Then, the maximum distance from any trajectory point to this centroid is determined. This measure provides insight into how spread out the trajectory is.
- **Trajectory Classification:** Based on the computed turn angles and bounding radius, each trajectory is classified into one of three categories:
    1. **Label 0 (Good Trajectory):** The trajectory displays smooth motion with few abrupt turns. These trajectories are preserved as-is.
    2. **Label 1 (Small Anomaly):** The trajectory exhibits a moderate number of sharp turns but is confined within a small spatial radius. Such trajectories, often corresponding to parked or stopped vehicles, are represented by a single point (typically the centroid).

Figure 14: Small Anomaly Pre vs Post-Processing

3. **Label 2 (Big Anomaly):** The trajectory has a high fraction of abrupt turns and a large spatial dispersion. These are considered too erratic and are excluded from the final results.

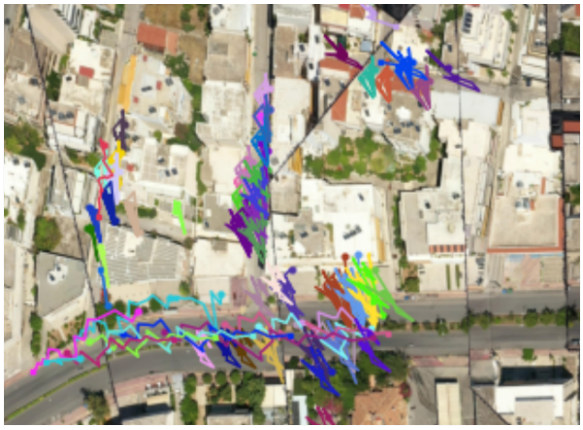

Figure 15: Big Anomaly Detection

In practice, the post-processing function computes the turn angles using cosine similarity between consecutive movement vectors, determines the bounding radius based on the maximum distance from the centroid, and then applies the above criteria to classify each trajectory. This classification not only helps in filtering out unreliable trajectories but also distinguishes between normal vehicle movements and those indicative of vehicles that are stationary or with big anomalies.

3.2.4 Challenges in Vehicle Tracking

Vehicles shown in black regions In the vehicle tracking phase, the cropping issue from the stitching stage led to certain vehicle trajectories appearing in "black regions" on the final map. These vehicles, while tracked in the original video, were not shown on the road in the stitched panorama.

Untracked regions For computational efficiency, the same rotation was applied to all frames within each batch by using the same rotation transformation as the first selected frame in the batch. This approach proved problematic in scenarios where the camera underwent significant directional changes (e.g., alternating left and right rotations in small intervals of time), causing

two of the batches to fail in tracking properly. Although computing an optimal rotation for each frame individually could potentially mitigate these issues, such an approach would have required significantly more computational resources and processing time.

4 Results and Discussion

4.1 Map Generation

4.1.1 Two Way Stitching vs One Way Stitching

In our approach, we implement a two-way stitching pipeline to construct a high-quality panorama from drone-captured images. Instead of performing one-way stitching, where the images are sequentially combined and cumulative errors can propagate throughout the entire sequence, we first divide the image set into two halves. The first half is stitched in order until a midpoint is reached. At this point, we reset the coordinate system using the initial image's coordinates, thus reinitializing the error accumulation. This "reset" minimizes the propagation of small errors that otherwise compound in one-way stitching, leading to a more accurate and visually coherent panorama.

Figure 16: Two-Way Stitching Result

Figure 17: One-Way Stitching Result

Comparison of stitching results: The two-way stitching method effectively resets the propagation of errors between halves, whereas the one-way stitching accumulates errors continuously. We see the difference especially on the south region when comparing the two panoramas, the main road in the real world is orthogonal to the main road on the east region, but only in the two-way stitching result can we see that it still remains orthogonal, but not in the one-way stitching, where we can detect a distortion coming from the error propagation not being reset.

4.1.2 Critical Assessment of the Generated Map

In this stage, we measure the accuracy of the generated map by aligning and comparing two trajectories: one derived from the drone's GPS data and the other obtained via image stitching.

GPS and Stitching Trajectories: The GPS trajectory is computed by triangulating the drone's longitude, latitude, and altitude. This triangulation accounts for variations in the drone's height across frames. In parallel, the stitching trajectory is generated by following the centers of each selected frame, which were used in the panorama construction.

Rescaling to a Common Reference Frame: Since the GPS trajectory is measured in meters and the stitching trajectory in pixels, the two must be rescaled to a common reference frame. We achieve this by assuming that the distance between the first and second selected frames is equivalent in both trajectories. This assumption is justified by the minimal cumulative error in the initial segment. Consequently, both trajectories are normalized so that the first segment has unit length and are then centered at the origin.

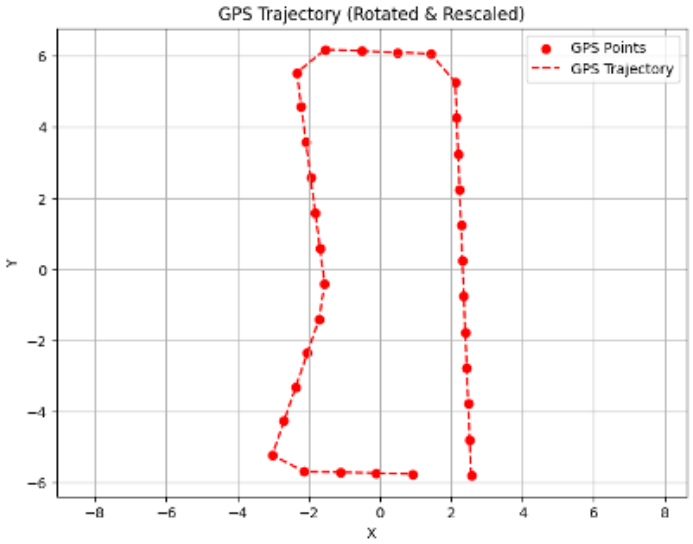

Figure 18: GPS Trajectory Rescaled

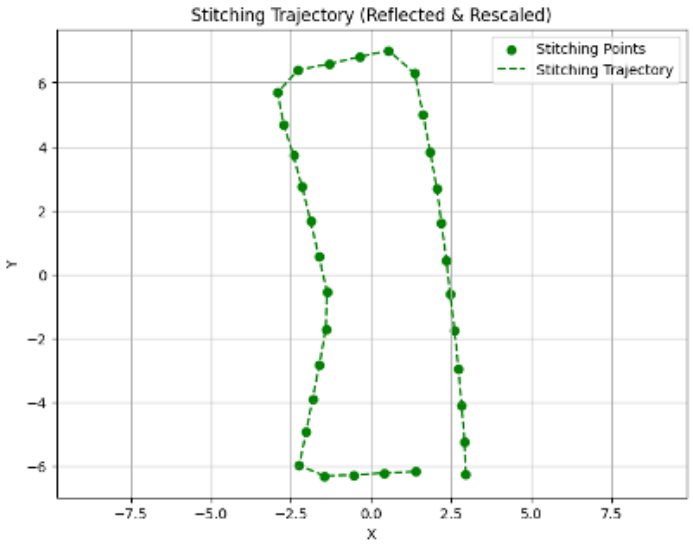

Figure 19: Stitching Trajectory Rescaled

PCA Transformation: To facilitate shape comparison between the rescaled trajectories, we apply Principal Component Analysis (PCA) to each trajectory. PCA reduces the dimensionality of the data, highlighting the primary movement trends in a common 2D space. This transformation allows us to visually and quantitatively compare the overall shapes of the trajectories.

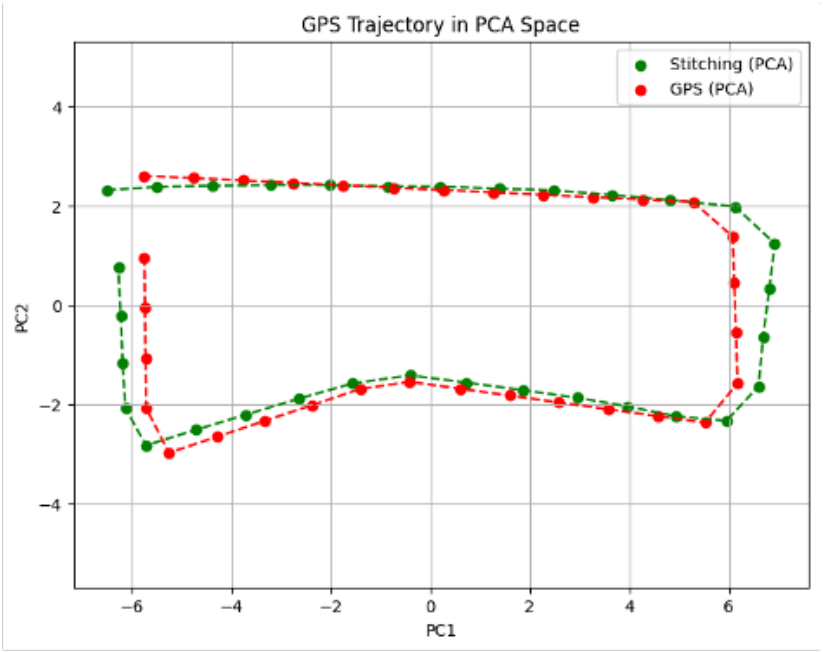

Figure 20: GPS and Stitching Trajectories in the PCA Space

Similarity Metrics: MED and RMSE

To quantitatively compare the GPS-derived trajectory with the stitching-derived trajectory, we employ two metrics: the Mean Euclidean Distance (MED) and the Root Mean Square Error (RMSE). Both metrics measure the deviation between corresponding points in the two trajectories.

Mean Euclidean Distance (MED): For two trajectories $T1$ and $T2$ containing N corresponding 2D points, the MED is computed as:

$$\text{MED} = \frac{1}{N} \sum_{i=1}^{N} \|T1_i - T2_i\|$$

where $\|T1_i - T2_i\|$ denotes the Euclidean distance between the ith points of $T1$ and $T2$.

Root Mean Square Error (RMSE): The RMSE is calculated by first computing the squared Euclidean distance for each pair of points, averaging these squared differences, and finally taking the square root:

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \|T1_i - T2_i\|^2}$$

This metric places a higher emphasis on larger deviations.

Similarity Percentage: To convert these error metrics into a similarity measure, the total length of a trajectory is first determined as the sum of Euclidean distances between consecutive points. The similarity percentage is then defined as:

$$\text{Similarity} = \max\left(0,\ 100 - \frac{\text{metric}}{\text{total length}} \times 100\right)$$

Results: In our evaluation, the similarity percentages computed from the MED and RMSE were:

- **MED-based Similarity:** 98.63%
- **RMSE-based Similarity:** 98.48%

These high similarity values suggest that the stitching-derived trajectories are very similar to the GPS trajectories, validating our stitching approach.

4.2 Vehicle Tracking

4.2.1 Visualization of the Mapped Trajectories

Figure 21: After Post-Processing

Figure 22: Before Post-Processing

Comment : After applying post-processing, the final map with vehicle trajectories appears significantly cleaner compared to the raw tracking results. The removal of high-anomaly trajectories ensures that only well-tracked vehicles remain, reducing clutter and improving overall clarity. As a result, individual vehicle trajectories are more distinguishable, making it easier to analyze movement patterns and extract meaningful insights from the final visualization.

4.2.2 Vehicle Tracking Assessment

To evaluate the quality of the trajectory mapping from video to the final panoramic map, we use a metric based on the ratio of vehicles that are successfully shown after post-processing to the total number of vehicles tracked. This metric is defined as follows:

$$\text{Score} = \frac{\text{Vehicles}_{\text{shown}}}{\text{Vehicles}_{\text{total}}} \times 100\%$$

In our case, we have:

$$\text{Score} = \frac{8159}{8550} \times 100\% \approx 95.42\%$$

This result indicates that approximately 95.42% of the tracked vehicles have trajectories that are successfully mapped onto the final panorama. The remaining $100\% - 95.42\% \approx 4.58\%$ correspond to vehicles that were classified as high anomalies (label 2) during post-processing and, consequently, were not displayed in the final map. This metric thus provides a quantitative assessment of the overall tracking and mapping quality.

5 Conclusion

In summary, the developed system successfully integrates drone video, image stitching, and vehicle tracking into a comprehensive mapping solution. The final panoramic map clearly displays individual vehicle trajectories with high accuracy and consistency. This robust approach not only aligns diverse data sources effectively but also provides an invaluable tool for analyzing directly traffic patterns from aerial footage.

References

[1] D. G. Lowe, "Object recognition from local scale-invariant features," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1150–1157, 1999.

[2] M. P. Philipp Lindenberger, Paul-Edouard Sarlin, "Lightglue: Local feature matching at light speed," *International Conference on Computer Vision*, 2023.

[3] S. A. A. H. Nidhal K. EL Abbadi and A. H. Abdulkhaleq, "A review over panoramic image stitching techniques," *Journal of Physics: Conference Series*, 2021.

[4] A. B. Z. G. L. O. F. R. B. Upcroft, "Simple online and realtime tracking," *Proceedings of IEEE International Conference on Image Processing*, 2016.

[5] N. W. A. B. D. Paulus, "Simple online and realtime tracking with a deep association metric," *IEEE International Conference on Image Processing (ICIP)*, 2017.

[6] B.-Z. B. Nir Aharon, Roy Orfaig, "Bot-sort: Robust associations multi-pedestrian tracking," *arXiv preprint arXiv:2206.14651*, 2022.

[7] A. M. A. R. A. A. A. S. N. K. H. G. K. P. V. L. Roman Suvorov, Elizaveta Logacheva, "Resolution-robust large mask inpainting with fourier convolutions," *arXiv preprint arXiv:2109.07161*, 2021.