Department of Mathematics and Computer Science

University of Heidelberg

Master thesis

in Scientific Computing

submitted by

Veerav Chebrolu

born in Mumbai

2025

Vehicle velocities and Congestion detection on highways from Planetscope Imagery

This master thesis has been carried out by Veerav Chebrolu

at the

Heidelberg University

under the supervision of

Prof Carsten Rother

Prof. Alexander Zipf

(Fahrzeuggeschwindigkeiten und Stauerkennung auf Autobahnen mit Planetscope-Satellitenbildern):

Diese Arbeit untersucht Satellitenbilder von PlanetScope mit einer Bodenauflösung (Ground Sampling Distance, GSD) von 3 Metern zur Schätzung von Fahrzeuggeschwindigkeiten und zur Erkennung von Staus auf Autobahnen. Zwischen den einzelnen Spektralbändern eines Bildes besteht ein zeitlicher Versatz, wodurch sich bewegende Fahrzeuge in den Kanälen verschoben erscheinen. Durch die Bestimmung dieses Versatzes zwischen den Kanälen und die Ableitung der relativen Unterschiede in den Aufnahmezeiten der Spektralbänder lassen sich Fahrzeuggeschwindigkeiten berechnen. Wir formulieren dieses Problem zunächst als Optical-Flow-Problem, wobei wir für die Lokalisierung einen Laplacian-of-Gaussian-Ansatz und für die Messung der Verschiebungen zwischen den Bändern den Lucas-Kanade-Algorithmus verwenden. Um robustere Ergebnisse zu erzielen, formulieren wir es anschließend als Keypoint-Schätzproblem und setzen ein modifiziertes Mask-RCNN im Top-Down-Ansatz ein, um die Verschiebungen mit Schlüsselpunkten über drei Kanäle zu detektieren. Mit HDBSCAN identifizieren wir anschließend Cluster langsam fahrender oder nahezu stehender Fahrzeuge, um die Länge des Staus zu bestimmen. Unsere Ergebnisse validieren wir anhand von Verkehrsdaten von TomTom und beobachten eine gute Übereinstimmung in den Mustern.

(Vehicle velocities and Congestion detection on highways from Planetscope Imagery):

This thesis investigates PlanetScope satellite imagery with a ground sampling distance (GSD) of 3 meters to estimate vehicle velocities and detect congestion on highways.There is a temporal offset between each spectral band in image, and as a result moving vehicles appear shifted across channels. Finding this offset across channels, and deducing relative differences in acquisition times between spectral bands gives us vehicle velocities. We set up this problem, firstly, as an optical flow problem using Laplacian-of-Gaussian for localization and Lucas Kanade for measuring inter band displacements. To obtain more robust results, we then frame it as a keypoint estimation problem, and use a top-down modified Mask-RCNN to detect the shifts with keypoints across three channels. We then use HDBSCAN to identify clusters of slow moving/ near static vehicles to measure congestion tail length. We validate our data with TomTom traffic statistics, and see a good agreement in pattern.

Contents

1 Introduction

1.1 Background

Intelligent transportation systems, infrastructure planning, carbon-emission assessments, and congestion pricing all require accurate, up to date information on traffic volumes and flow patterns. Stop-and-go conditions have been shown to increase fuel consumption and CO_2 emissions compared to steady speeds; both modeling and field measurements link greater delay to higher emissions, with one urban study reporting $\sim 40\%$ higher emissions at ~ 45 km/h under congested conditions Barth and Boriboonsomsin [2008, 2009], Grote et al. [2016]. In 2019, the transport sector accounted for $\sim 25\%$ of total EU CO_2 emissions, of which 71.7% came from road transport—the only sector whose greenhouse gas emissions have risen over the past three decades European Environment Agency [2022].
Traditional traffic monitoring technologies such as inductive loops, radar, and roadside cameras provide high fidelity measurements but are expensive to install, require regular maintenance, and are sparse outside metropolitan areas. Middleton et al. [2002], Yu et al. [2013].

Continuous counting stations provide detailed temporal profiles of traffic flow but are rarely deployed widely enough to accurately represent traffic across an entire network Federal Highway Administration [2022].

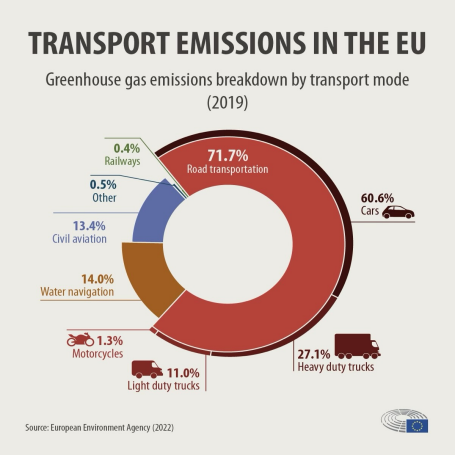
In many developing regions, permanent ground-based sensors are absent altogether, and available probe data from commercial fleets or navigation apps is often proprietary and biased toward certain road classes.

High-resolution satellite imagery offers a cost-effective, scalable alternative for obtaining network-wide traffic information, particularly in locations where ground sensors are sparse or non-existent. It can provide consistent spatial coverage, historical archives for trend analysis, and non-intrusive data collection without the need for physical roadside infrastructure. When fused with sparse permanent sensors and probe data from sources such as TomTom, Google Maps, or INRIX, satellite observations can improve coverage equity while lowering monitoring costs per kilometer.

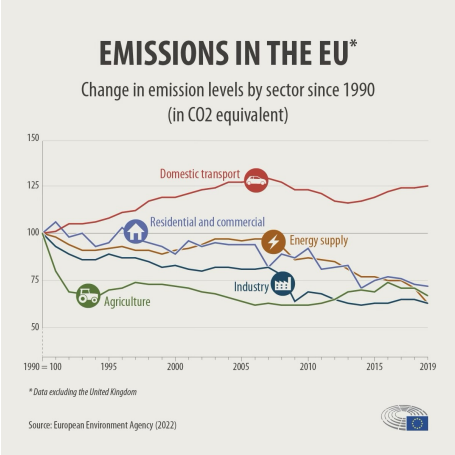
The commercialization of space has greatly expanded the quantity and variety of available Earth observation data, opening new opportunities for large-scale traffic monitoring. PlanetScope imagery, with a ground sampling distance of ~ 3 m and near-daily revisit, is suited for capturing dynamic traffic phenomena such as congestion and incident-related delays. Its pushframe sensor design allows for multi-band captures that can be exploited to estimate vehicle velocities from inter-band timing differences.

In this study, we use PlanetScope imagery to derive vehicle velocities and detect

congestion on highways. Our aim is to demonstrate the feasibility of wide-area traffic monitoring from space, assess the limitations of the approach..



(a) Transport-related CO_2 emissions in the EU



(b) Historical shifts in EU CO_2 emissions by sector

Figure 1.1: Image source European Environment Agency [2022]

1.2 Prior Work and Contributions

1.2.1 Prior Work

This thesis builds on the work of Adamiak et al. [2024], developed at the Heidelberg Institute for Geoinformation Technology (HeiGIT). A preprint and the authors' software implementation were available during this project and served as the baseline. The same work was subsequently published as Adamiak et al. [2025]. Their problem formulation and training pipeline are adopted and extended in this thesis.

1.2.2 Contributions

The specific contributions of this thesis are:

- **Additional semantic label for congestion analysis.** An explicit class for *slow-moving/static vehicles* is added, enabling downstream estimation of congestion tail lengths.
- **Geometry-aware loss.** Additional geometric prior loss terms are added that keeps keypoint trajectories realistic and smoother, especially in dense traffic.
- **Architecture changes for small objects.** Backbone and head are adjusted to keep higher-resolution features; pooling and training settings are tuned, and RGB+NIR input is used to better capture small objects in low-resolution imagery.

- **Validation with TomTom data.** Model outputs are validated against independent TomTom traffic data to assess agreement with real-world traffic.

1.3 Problem Statement

Multispectral push-frame systems such as Planet's SuperDove satellite, capture each spectral band sequentially as the satellite moves along its orbital track. This is due to the physical offset between the sensor arrays that capture different channels, and the placement order of these arrays determines the sequence in which the bands are recorded.
For SuperDove, the order is: Blue, Red, Green1, Green2, Yellow, Red Edge, NIR, and Coastal Blue, as illustrated in Figure 1.2. This sequential acquisition results in a temporal offset between image channels, which offers a unique opportunity to measure motion. Moving objects appear at different positions across channels, and their trajectories can be inferred from the intensity peaks.

This shift is visible across all visible range imagery channels. In our study, we utilize three visible bands — Blue, Red, and Green2. We use Planet's four-channel imagery product (Blue, Red, Green2, and NIR), which is processed and analysis-ready. The subsequent image processing steps are described in Section 2.1.

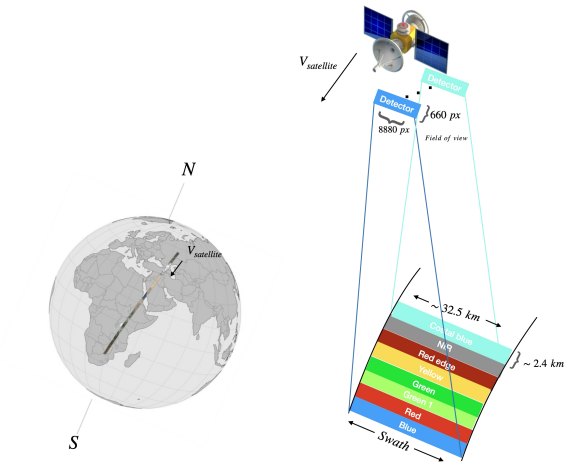


Figure 1.2: Swath illustration of the PlanetScope satellite

The time delay is sufficient to produce a noticeable displacement of fast moving vehicles across channels. The blue sensor array is positioned first with respect to the satellite's direction of motion. Since Green1 lies between red and Green2, the available Green2 channel in our imagery results in a time interval between red and green that is approximately twice the interval between blue and red. This is illustrated in 1.3. Vehicles in a 3m GSD image appear as small bright blobs (few pixels) in each visible range band. We illustrate the shift across channels in 1.4.

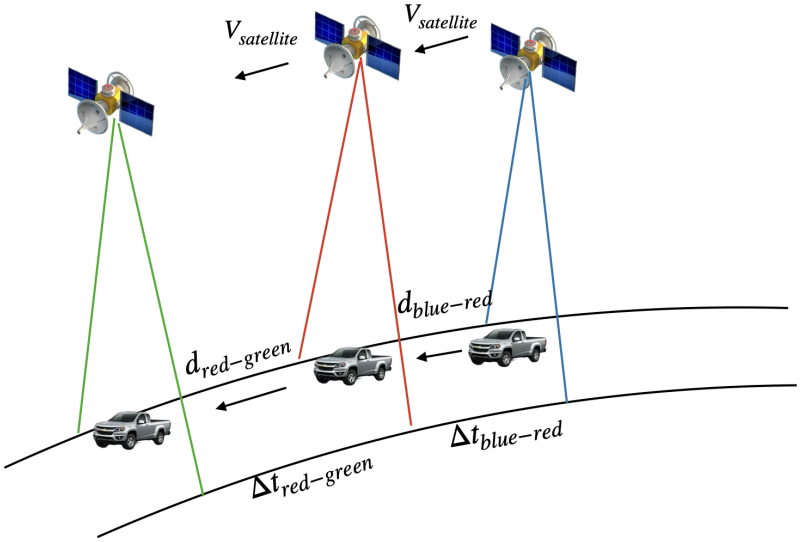
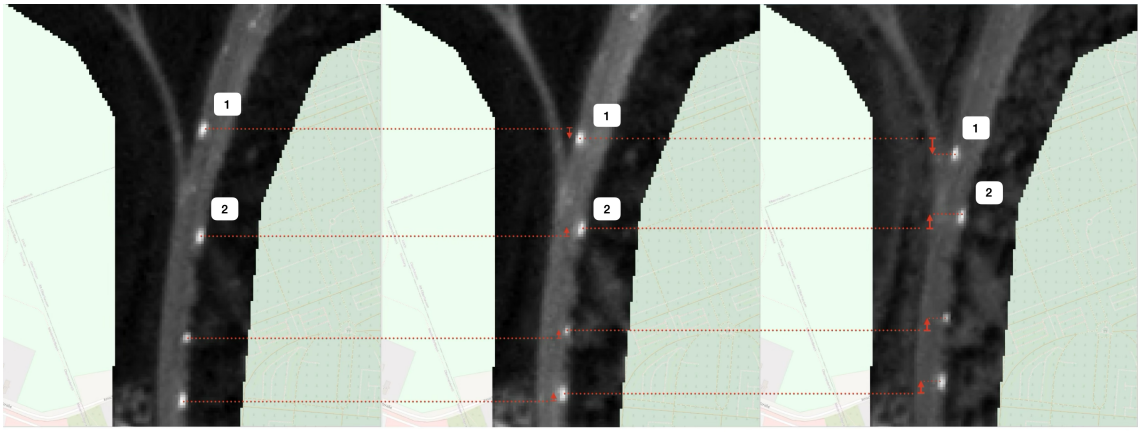
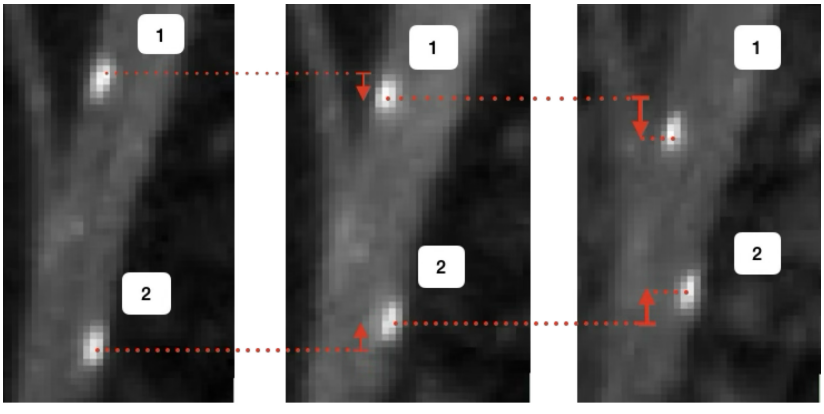


Figure 1.3: Vehicle displacement due to temporal offsets between blue, red, and green band acquisitions in pushframe satellite imaging.



blue band *red band* *green band*

(a) movement of vehicles across bands

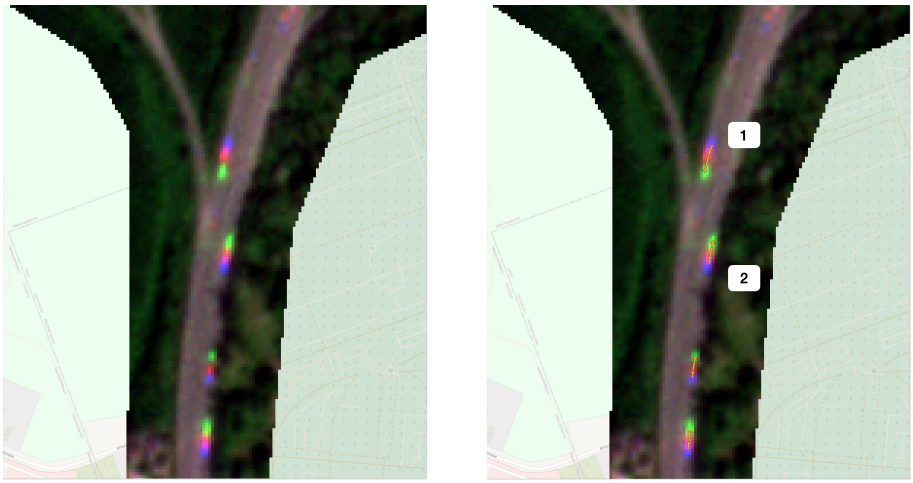


blue band *red band* *green band*

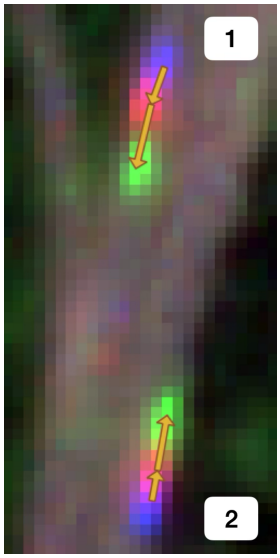
(b) Zoomed-in view of the marked region

Figure 1.4: displacement across imagery channels

Because the blobs are shifted slightly across each channel, they produce “rainbow” effect when seen in RGB image as shown in 1.5



(a) Moving vehicles produce "rainbow" effect. The direction of motion is blue → red →green.



(b) Zoomed-in view of the marked region

Figure 1.5: rainbow effect

Vehicles that are stationary or moving at very low speeds (approximately 30 km/h or less) do not show “rainbow” effect, as there is no measurable shift across channels.

Figure 1.6: Vehicles in lane 0 are moving at low speeds compared to those in the lane 1, resulting in no "rainbow" effect in lane 0.

We can compute the average velocity as :

$$V_{\text{vehicle}}^{\text{avg}} = \frac{d_{\text{blue-red}} + d_{\text{red-green}}}{\Delta t_{\text{blue-red}} + \Delta t_{\text{red-green}}} \tag{1.1}$$

For finding the displacements across the channels, we first formulate it as an optical flow problem and discuss it in 3.1. Then for more robust results, we formulate it as a keypoint estimation model, making use of neural network, and discuss it in 3.2. To deduce the time interval, we make use of sensor array width, satellite velocity, and ground sampling distance. We discuss this in 2.4

2 Data

2.1 About PlanetScope Imagery

The PlanetScope Superdove satellite images are captured daily for the entire planet. The temporal frequency is one day, i.e each location in the planet is captured once daily between 9:30-11:30 local solar time. The data we download has the following nomenclature :

`{acquisition-date}_{acquisition-time}_{acquisition-time-seconds-hundredths}_{satellite-id}_{product-level}_{band-product}.{ext}` An example for the image name would be `20240801_101505_03_241c_3B_AnalyticMS_SR_8b.tif` There are upto 8 bands in an image, and we use 4 band Blue,Red,Green2,NIR image. We use the analysis-ready version, which has undergone processing steps of orthorectification and surface reflectance.

2.1.1 Data Aquisition

The PlanetScope constellation is operated by Planet Labs PBC. It consists of more than 120 nanosatellites known as *Doves*, each weighing only 5.8 kg. These satellites acquire multispectral imagery at a ground sampling distance (GSD) of 3.7-4.2 m (depending on orbital altitude) and produce a 3 m resolution analysis-ready products. They operate in a sun-synchronous orbit (SSO) at an altitude of 475–525 km, with an inclination of approximately 98°. In each orbit, the satellites travel from the northern hemisphere toward the southern pole on the descending pass, imaging the Earth's surface at the same local solar time for a given latitude. This ensures consistent illumination conditions for acquisitions within the same season.

Each Dove satellite carries an eight-band push-frame imager with a butcher-block filter, capturing at wavelengths : Coastal Blue (443 nm), Blue (490 nm), Green 1 (531 nm), Green 2 at (565 nm), Yellow (610 nm), Red (665 nm), Red Edge (705 nm), and NIR (865 nm) . Notably, six of these eight bands : Coastal Blue, Blue, Green 2, Red, Red Edge, and NIR are interoperable with Sentinel-2 bands (Sentinel-2 B1, B2, B3, B4, B5, and B8a, respectively), whereas Green 1 and Yellow do not have direct equivalents.PBC [2024]

The imagery undergoes preprocessing before it is analysis ready. Preprocessing typically includes radiometric and geometric corrections. Radiometric corrections address sensor irregularities and remove unwanted sensor or atmospheric noise, ensuring that the digital numbers accurately represent surface reflectance. Geometric corrections account for distortions arising from sensor-Earth geometry variations.

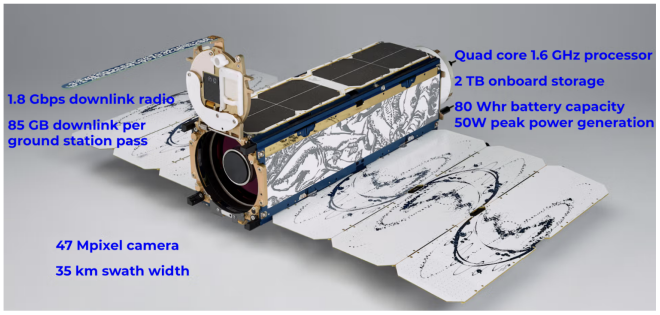


Figure 2.1: Planestscope dove satellite. Image source :Devaraj [2019]

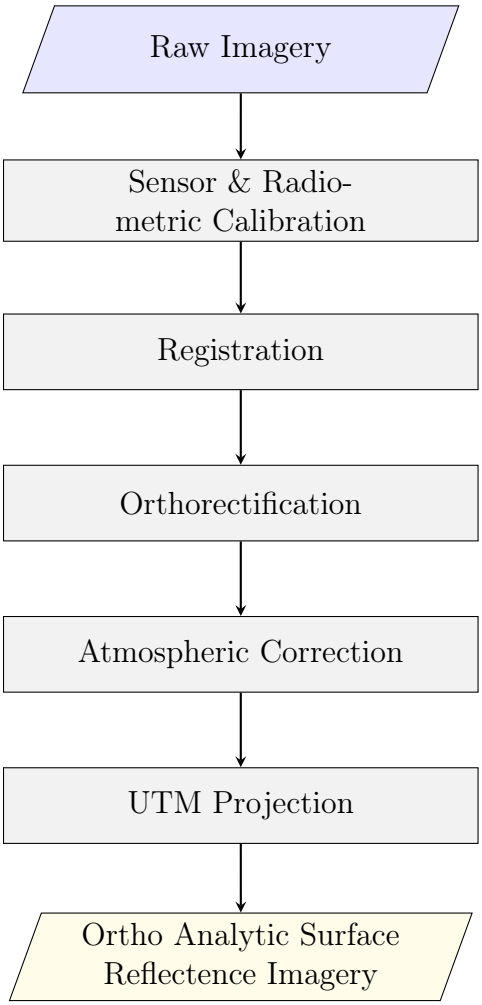


Figure 2.2: Preprocessing pipeline for generating surface reflectance imagery.

2.1.2 Sensor and Radiometric Calibration

Calibration is the process of determining and applying corrections to sensor parameters to ensure that the output data accurately represent physical units. This process is essential for maintaining data accuracy and consistency over time. Calibration ensures that spectral measurements accurately capture the radiance received by the sensors from specific ground locations.
Radiometric calibration converts the raw digital numbers recorded by the sensor into physically meaningful units such as radiance or reflectance, which quantify the scattering properties of the observed surface. Factors like sensor artifacts, viewing and illumination angles, or atmospheric conditions affect satellite's digital number (DN) values.

The first step in this process is to convert DN values to at-sensor radiance by applying a radiometric scale factor:

$$\mathrm{RAD}(i) = \mathrm{DN}(i) \times \mathrm{radiometricScaleFactor}(i), \tag{2.1}$$

where $\mathrm{radiometricScaleFactor}(i) \approx 0.01$. The resulting radiance is expressed in watts per steradian per square meter per micron ($\mathrm{W} \cdot \mathrm{m}^{-2} \cdot \mathrm{sr}^{-1} \cdot \mu\mathrm{m}^{-1}$).

Before applying the radiometric scaling, several corrections are performed. First, darkfield or offset correction removes sensor bias and dark current noise. This is done using master offset tables created from on-orbit darkfield measurements, grouped by CCD temperature bins. Next, flat field correction compensates for uneven detector response and illumination across the focal plane. These flat field patterns are measured before launch and updated periodically while in orbit. Finally, camera parameter normalization adjusts for differences in gain, exposure time, time delay integration (TDI) stages, and operating temperature. This ensures a consistent radiometric response across all images.
Because the spectral responses of SuperDove are similar to those of Sentinel 2, it enables for on-orbit absolute calibration using near-simultaneous crossover acquisitions with Sentinel-2 as the reference sensor. No spectral band adjustment factor (SBAF) is required, and crossover scenes can be collected anywhere in the world, not only at well-characterized calibration sites. Planet targets radiometric agreement within 5% of Sentinel-2.
Sentinel-2 itself performs absolute radiometric calibration monthly using an onboard diffuser, with validation carried out over Pseudo-Invariant Calibration (PIC) sites such as bright desert surfaces. By referencing Sentinel-2, Planet ensures long-term stability and interoperability across its SuperDove fleets. Collison et al. [2025], U.S. Geological Survey [2025], Humboldt State University [2020], PBC [2024]

Figure 2.3: Radiometric correction

2.1.3 Registration

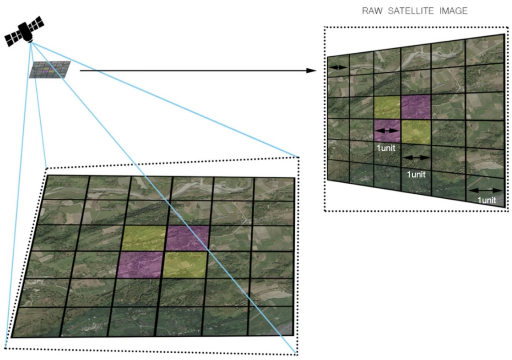
Geometric registration is the process of aligning an image to a known coordinate system or to another image. The goal is to ensure that each pixel in the image corresponds to its correct geographic location. This is necessary because raw satellite images often contain geometric distortions caused by sensor geometry, satellite motion, Earth rotation and curvature, and terrain variations.
The process begins by identifying a set of Ground Control Points (GCPs) in the image. These are distinct, easily recognizable features on the Earth's surface whose geographic coordinates are known with high accuracy, such as road intersections, building corners, or isolated landmarks. The same points are identified in the image.

Once a sufficient number of well-distributed GCP pairs are collected, a mathematical transformation is computed. This includes an affine or polynomial warp, to map the image coordinates to the target coordinate. The parameters of this transformation are typically estimated using least-squares fitting to minimize positional errors at the GCPs. After determining the transformation, the image is resampled to assign pixel values to their corrected locations, using methods such as nearest neighbor, bilinear interpolation, or cubic convolution.

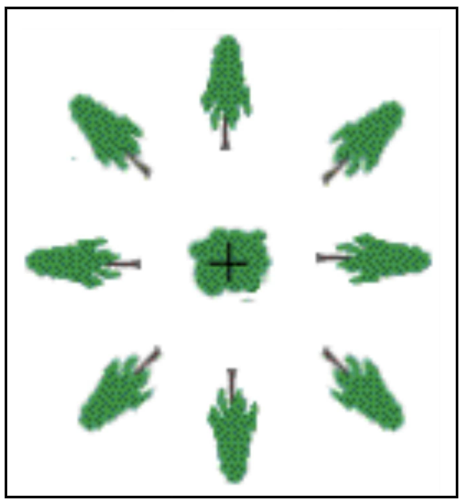
2.1.4 Orthorectification

Orthorectification is the process of removing image distortions caused by sensor tilt and terrain-induced relief displacement. When the sensor views the surface from an off-nadir angle, it causes features farther from the sensor's ground track to appear displaced. Relief displacement results from variations in terrain elevation. Elevated objects, such as buildings or mountains, appear to lean radially outward from the image center relative to their true ground positions.

These distortions can cause significant positional errors if uncorrected, especially in areas with large elevation differences. Orthorectification addresses this by modeling the imaging geometry and reprojecting each pixel to its correct position on the ground in a chosen map projection. This process requires accurate knowledge of the satellite's position, orientation, and viewing geometry at the time of image acquisition. This information is supplied in the form of *Rational Polynomial Coefficients* (RPCs), which provide a mathematical relationship between image coordinates (row, column) and geographic coordinates (latitude, longitude, elevation).



(a) Perspective distortion. Off-Nadir viewing angle. Image source: Setyawan [2019]



(b) Relief displacement. Elevated objects appear to lean radially outward . Image source: Canada Centre for Remote Sensing [2025]

Figure 2.4: Illustration of geometric distortions in satellite imagery.

2.1.5 Surface Reflectance Processing

Surface reflectance accounts for temporally, spatially, and spectrally varying scattering and absorption effects of atmospheric gases, aerosols, and water vapor. Correcting for these effects is essential to reliably characterize the Earth's land surface. It is defined as the fraction of incoming solar radiation that is reflected by the Earth's surface. Correction is applied to top-of-atmosphere (TOA) reflectance obtained after radiometric calibration, to produce surface reflectance product.
Planet uses the 6S radiative transfer model with secondary data from MODIS to account for atmospheric effects on the observed signal at the sensor for the PlanetScope constellation.PBC [2024]

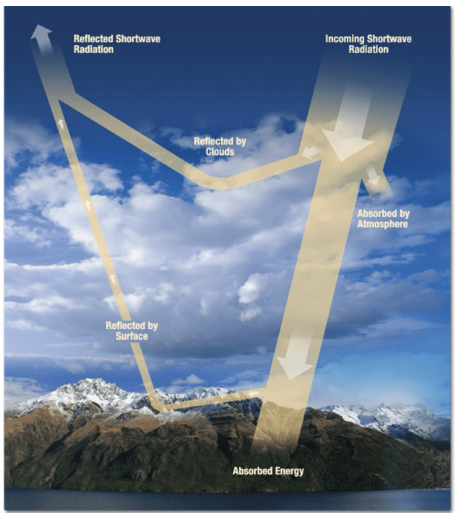


Figure 2.5: Correction is required to remove scattering and absorption effects from atmosphere to obtain surface reflectance properties. Image source : ARSET [2025]

2.2 Downloading road networks at large scale

In order to download data at bulk, we need to smartly divide our area of interest into manageble pieces. Within each piece, we download only the regions surrounding road networks of interest. These regions are buffer polygons generated around the road centerline. For this we require either OpenStreetMap way IDs or relation IDs OpenStreetMap contributors [2025] of the roads of interest.

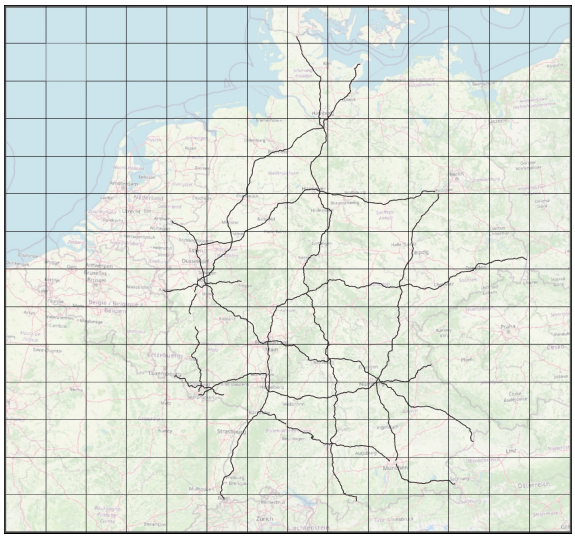


Figure 2.6: Subdivision of the target area to efficiently download highway data at scale. A1-A9 highways of Germany shown here.

The team at HeiGIT have developed a python software module, which helps us automate download bulk data from Planet server. This module takes in wayids or relation ids of the roads within the area of interest, desired date range, and maximum cloud cover parameter (used by Planet to return imagery with a cloud percentage lower than this value).
First, the module creates a road polygon of the requested highway by creating a buffer around the centerline of both lanes. Then a POST query is sent with these details, and we receive a response containing the IDs of available imagery that meets our stated conditions.
Then we place an order for these imagery IDs, and save the order metadata internally for future reference.
The delivery time ranges from a few minutes to several hours, depending on the amount of data and the type of imagery ordered. The delivery call involves regularly polling the Planet server, and once the order is ready, it is downloaded into our storage.

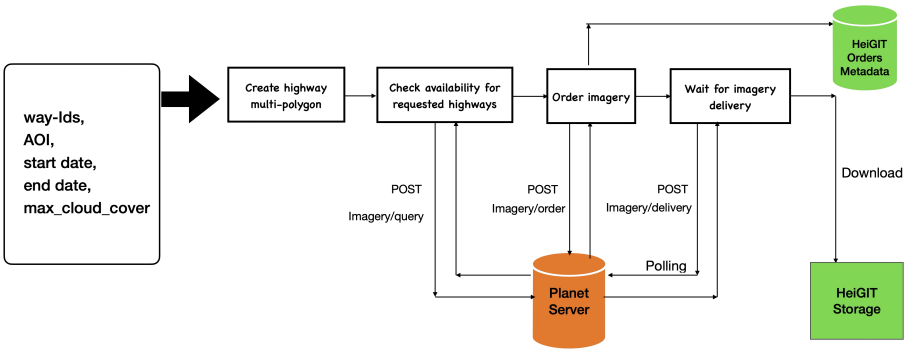


Figure 2.7: architecture for downloading imagery from planet server

In order to compute vehicle velocities with the methods discussed in 3, we break down the road network polygon into small pieces of approximately $350 - 500$ metres in length and $60 - 100$ metres in width for running our model.

Figure 2.8: Preparation of highway segments for analysis

2.2.1 Loading

We use QGIS QGIS Development Team [2025] to load and visualize the satellite imagery, as it supports 16 bit multichannel images, which match the format of our data. To view the entire region at once, the downloaded satellite tiles must first be stitched into a single raster. We apply the following pre-processing steps using GDAL GDAL/OGR contributors [2025] via Python subprocess calls:

1. Organize all GeoTIFF files by acquisition date and UTM zone.
   Implemented in `data_preparation/copy_geotiffs_by_date_and_zone.py`

2. For each date and zone, construct a virtual raster (VRT) using `gdalbuildvrt`, which creates a mosaic without duplicating data.
   Implemented in `data_preparation/create_virtual_raster.py`

3. Generate overview (.ovr) files for each VRT using `gdaladdo` at pyramid levels 2 to 64, improving rendering performance in QGIS.
   Implemented in `data_preparation/create_overview.py`

4. Compute raster statistics—including minimum, maximum, mean, standard deviation, and valid pixel percentage—using `gdalinfo`.
   Implemented in `data_preparation/create_stats.py`

5. Load the resulting VRTs into QGIS. With overviews and statistics precomputed, rendering is efficient and interactive performance is improved.

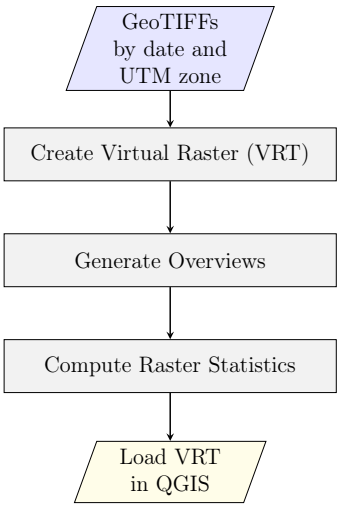


Figure 2.9: Workflow for preparing and loading imagery in QGIS.

2.3 Ground Truth Labelling in QGIS

2.3.1 Method

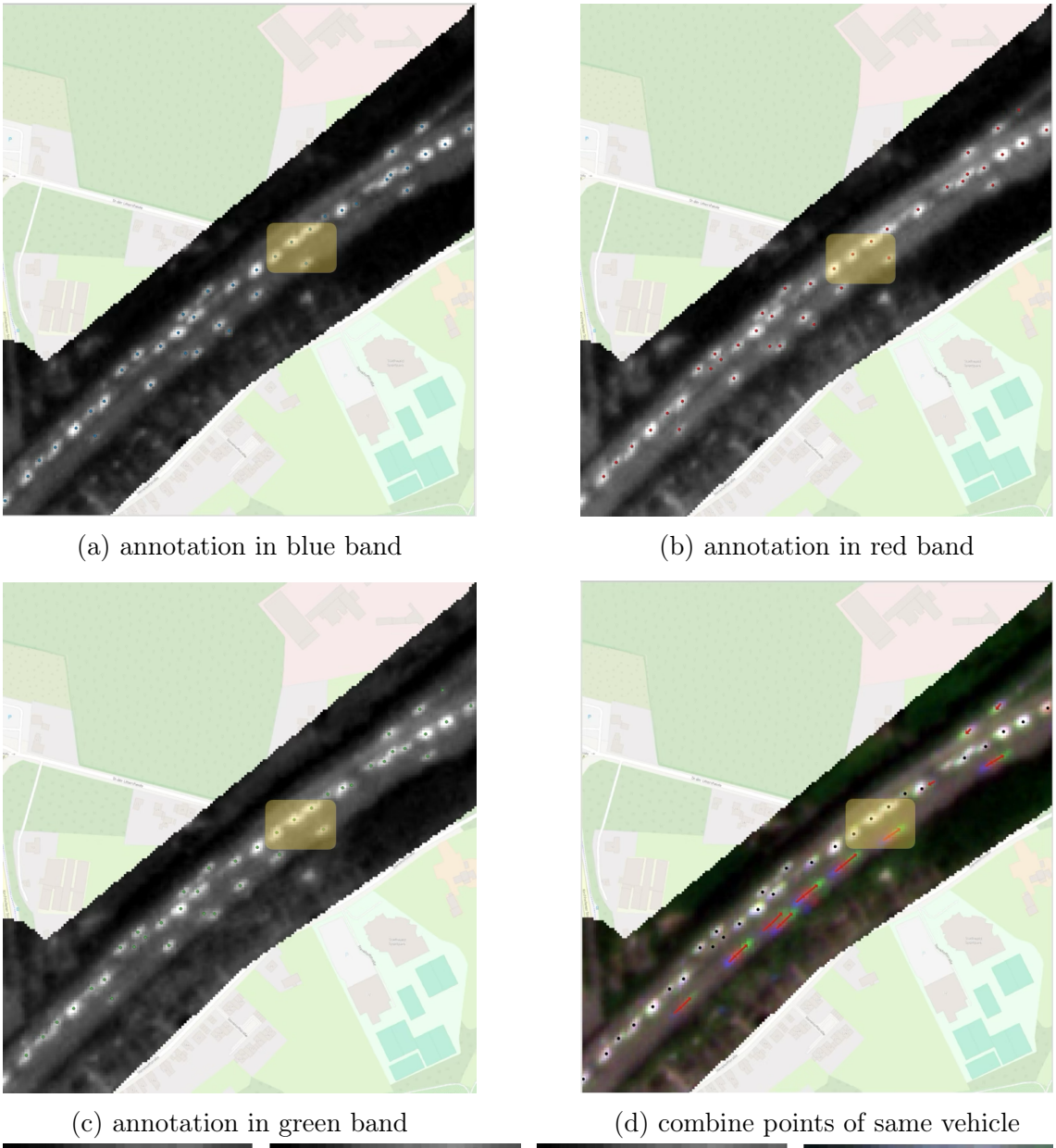
For training our keypoint model described in section 3.2, we need annotated ground truth labels. We make use of QGIS to do this. We first load the virtual raster by following steps discussed in 2.2.1.

Our objective is to open each band of the imagery one by one, observe the tiny blobs on roads, which corresponds to vehicles, and mark a point at the intensity peak of the blob. If the blob has multiple intensity peaks, or no intensity peak, we mark at the center region of the blob. We need to do this for all three channels one by one. To do this, we create a scratch vector layer in GGIS of Points geometry, with three fields: one for each band. We mark points now in this scratch layer and assign a field to each point.

To make labelling more intuitive and easier to follow, we color code each field with band color in QGIS, such that all points corresponding to blue channel will appear blue, and so on for red and green. We join these triplets by a linestring. We save our annotations in a geopackage format, which can be created as a QGIS vector layer for linestring geometry type. The first point in linestring is for blue keypoint, second for red keypoint and 3rd for green keypoint.

Incases where the displacement between blue to red is not significant, especially when vehicle is moving slowly, we save our results with a linestring with 2 points, indicating blue to green displacement.
In cases where no displacement occurs across all channels, we mark with a single keypoint across all channels. We save these points in a separate Point layer of our

annotated geopackage.



(a) annotation in blue band

(b) annotation in red band

(c) annotation in green band

(d) combine points of same vehicle

(e) zoom in view of marked region

Figure 2.10: Ground Truth labelling procedure illustration

2.3.2 Ground Truth Analysis

We plot the speed distributions based on our ground annotations, inferred from blue-red, red-green and the average blue-red-green displacements. The blue-red displacement has a wider spread, which is expected as the time interval is $\sim 300ms$. This can induce significant errors as even a subpixel error (say 0.5 px) of keypoint marked in blue and red bands can generate about $10m/s$ error.

The red-green band speed distributions are much narrower as expected, since the time interval is approximately double ($\sim 600ms$). The spread of the average blue-red-green speeds is slightly narrower, as it takes the average speed. It's speed profile closely matches to the red-green speeds, as our keypoint trajectories smoothened. The $600ms$ interval is short enough that deviations from straight line approximation remain small, even when the road trajectory is curved.

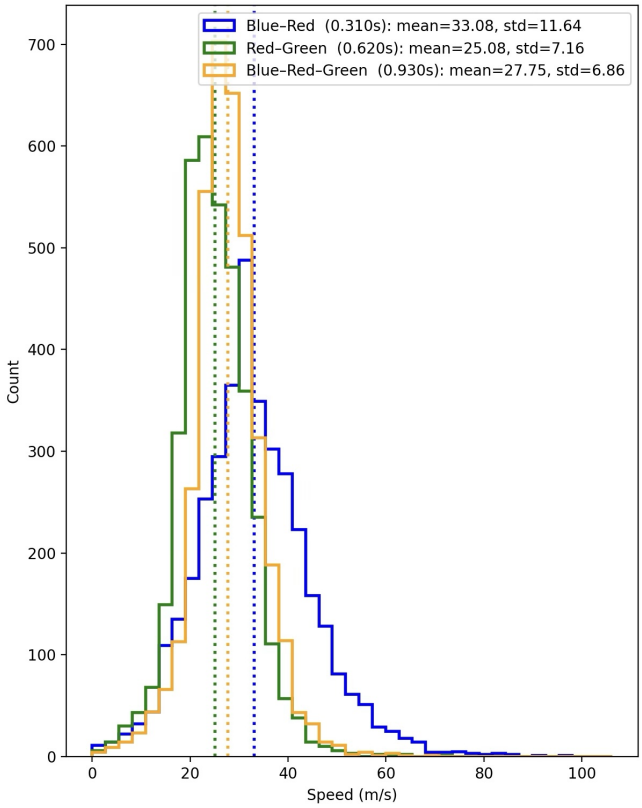


Figure 2.11: speed distribution of b-r, r-g, b-r-g displacements on ground truth

In an ideal case with 0 acceleration between blue-red and red-green time intervals, we would expect the data in the red-green vs blue-red displacement plot to be located around the line $y = 2x$, since the time interval between the red–green and the blue–red band is approximately 2 times. In practice from our ground truth annotations, we notice that the majority of our points lie slightly lower, around $y = 1.5x$.

This can be expected, as discussed in previous paragraph, that a 0.5 px offset can induce upto 3 metres displacement error (since gsd=3m/px). If we factor in this error, then for example, a point lying in the interval $(9, 10.5)$ in x-axis (blue-red) and $(12 - 15)$ in y-axis (red-green), after adjusting for a 1.5 metres shift on either side would make this interval $(7.5, 8)$ and $(13.5, 16.5)$. This would place majority of points closer to $y = 2x$.

We must also consider potential acceleration between intervals. Overestimation of the blue–red displacement is natural, as the displacement in this interval is small and the signal for marking ground truth can be ambiguous. This, in turn, can lead to an underestimation of the red–green displacement, since the red keypoint is now shifted toward the green keypoint.

A small percentage of points also lie around lines $y = x$ and $y = 3x$. This is plausible in certain cases, as we presume, overlapping between spectral bands during satellite image acquisition could lead to scenarios where time ratio interval between blue-red-green intervals is $1 : 1$ or $1 : 3$. This is discussed further in 2.4.2

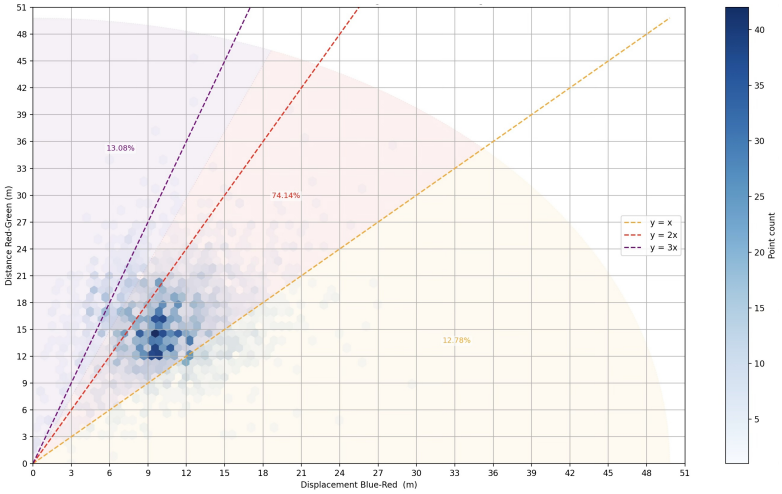
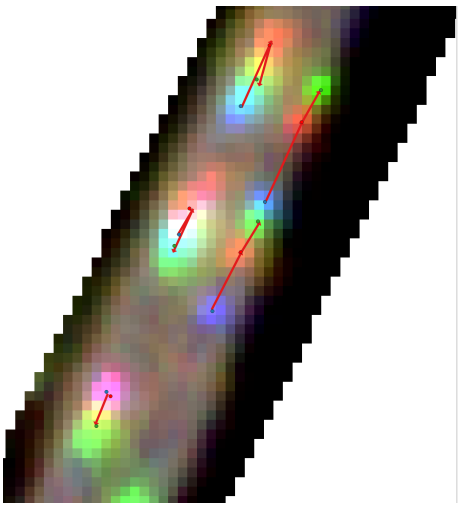
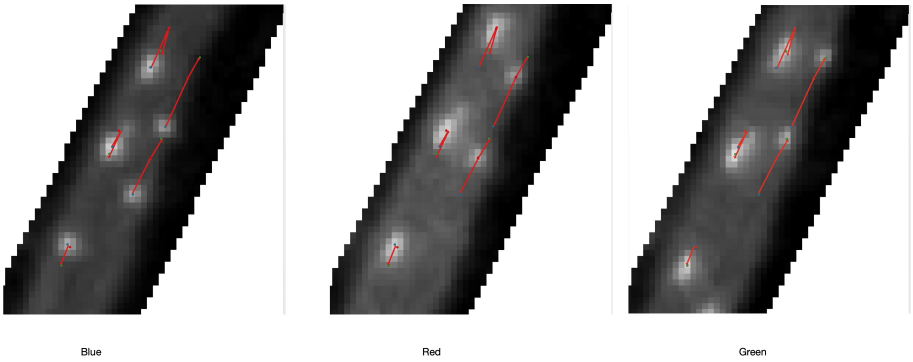


Figure 2.12: displacement red-green vs blue-red

Band Alignment issues While annotating the ground truth, we observed band alignment issues on few occasions. These are obvious in some cases but go unnoticed in many others. In Figure 2.13, on the left lane, the red band is misaligned and shifted upward. This is apparent because the vehicle's trajectory should be from top to bottom, but instead it appears to go from bottom to top and then down again. As a consequence, on the right, the blue–red displacement is increased while the red–green displacement is reduced. Without the adjacent lane for comparison, this misalignment would likely have gone unnoticed.



(a) Alignment error on the red channel. RGB visualization

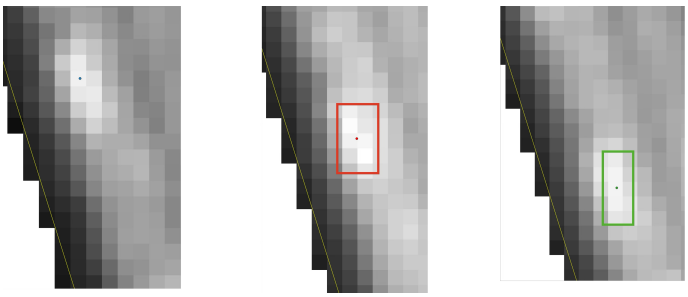


(b) Alignment error on the red channel. Band wise visualization

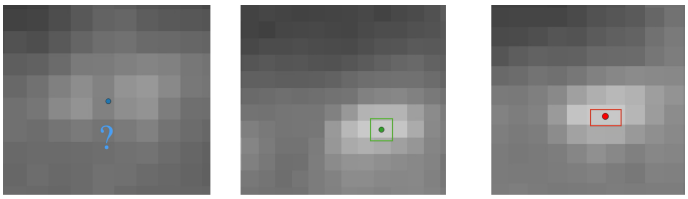
Figure 2.13: Example illustrating band alignment issues in imagery.

Pixel-level ambiguity while labeling Our labeling strategy is to mark the keypoint at the intensity peak of the detected blob. However, intensity peaks occur asymmetrically across bands, making it difficult to determine whether two peaks correspond to the same location. This makes pixel-level labeling challenging.

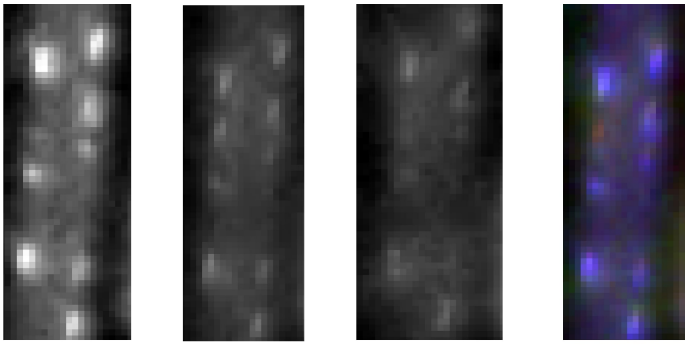
Furthermore, the intensity peak in each channel might not necessarily correspond to the exact same physical location of the vehicle. While this heuristic may work reasonably well for bright, neutral-colored vehicles (e.g., silver), where the reflectance is similar across all bands and the intensity peaks tend to coincide, it is less reliable for strongly colored vehicles. For example, a blue car may produce a strong peak in the blue channel, but the corresponding peaks in the red and green channels will be weaker and may occur at different pixels, leading to greater ambiguity in keypoint placement.
We have illustrated few examples in 2.14



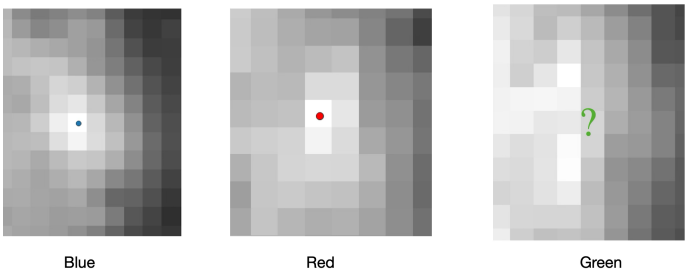
(a) Ambiguity in red and green channels: intensity peaks occur at multiple pixels.



(b) Ambiguity in the blue channel: no peak near the blob center, weak peaks in all channels.



(c) Signal weakens progressively from blue → red → green.



(d) Ambiguity in the green band due to low contrast.

Figure 2.14: Examples of pixel-level ambiguity during keypoint labeling

2.4 Calculating inter band time delay

2.4.1 Ideal

The frame rate of the satellite would be determined by its sensor array width, its orbital speed and the ground sampling distance. As the satellite moves forward, it must capture capture ground continuously without breaks. Therefore the time delay between two successive bands is given by :

$$\Delta t_{\text{b1}-\text{b2}} = \frac{d^{\text{satellite}}_{\text{b1}-\text{b2}}}{V_{\text{satellite}}} \tag{2.2}$$

$$d^{\text{satellite}}_{\text{b1-b2}} = w^{\text{sensor-array}}_{\text{b1}-\text{b2}} \cdot \text{GSD} \tag{2.3}$$

$$\Delta t_{\text{blue-red}} = \frac{w^{\text{sensor-array}}_{\text{blue-red}} \cdot \text{GSD}}{V_{\text{satellite}}}, \quad \Delta t_{\text{red-green}} = \frac{w^{\text{sensor-array}}_{\text{red-green}} \cdot \text{GSD}}{V_{\text{satellite}}} \tag{2.4}$$

Assuming GSD $\sim$ 3.7 m and $V_{sat} = 7600m/s$, sensor width $w = 660$ time delay /per band $= 321ms$

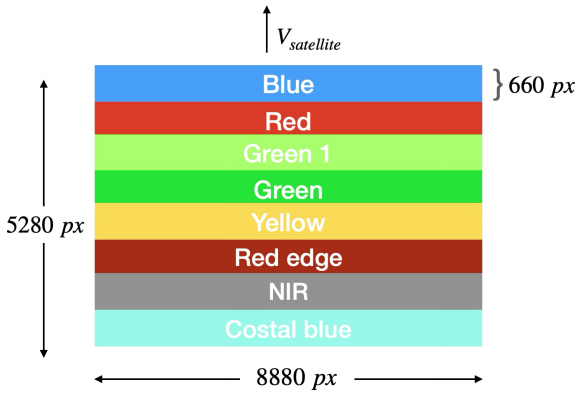


Figure 2.15: spectral band order of pushframe sensor array of superdove

2.4.2 Inferred

Planet does not publish band overlap specifications for the SuperDove sensor, though earlier PlanetScope satellites have been reported to use an 8% spectral overlap Aati et al. [2022]. Assuming the same for SuperDove, the ideal per-band delay from Section 2.4 of 321.32 ms is reduced to $0.92 \times 321.32 \approx 295.61$ ms in the overlap region.

The sensor footprint can be divided into three regions: A–B (8% of footprint) where bands are captured at t_0, t_2, t_4; B–C (16%) with captures at t_0, t_1, t_4; and the remainder (76%) with captures at t_0, t_1, t_3.

Weighting these timings by area gives:

$$\Delta t_{\text{blue}-\text{red}} = 0.92(1 \times 295.61) + 0.08(2 \times 295.61) \approx 319.26 \text{ ms},$$

$$\Delta t_{\text{red}-\text{green}} = 0.76(2 \times 295.61) + 0.16(3 \times 295.61) + 0.08(2 \times 295.61) \approx 638.52 \text{ ms},$$

with a total blue–green interval of ≈ 957.78 ms, matching the ideal mean.

Although the mean is unchanged, the timing distribution differs: in 8% of cases $\Delta t_{\text{blue}-\text{red}}$ and $\Delta t_{\text{red}-\text{green}}$ are roughly equal (1:1), while in 16% of cases they are in a 1:3 ratio. This variability can introduce temporal inconsistencies between bands even when the average delay matches the ideal.

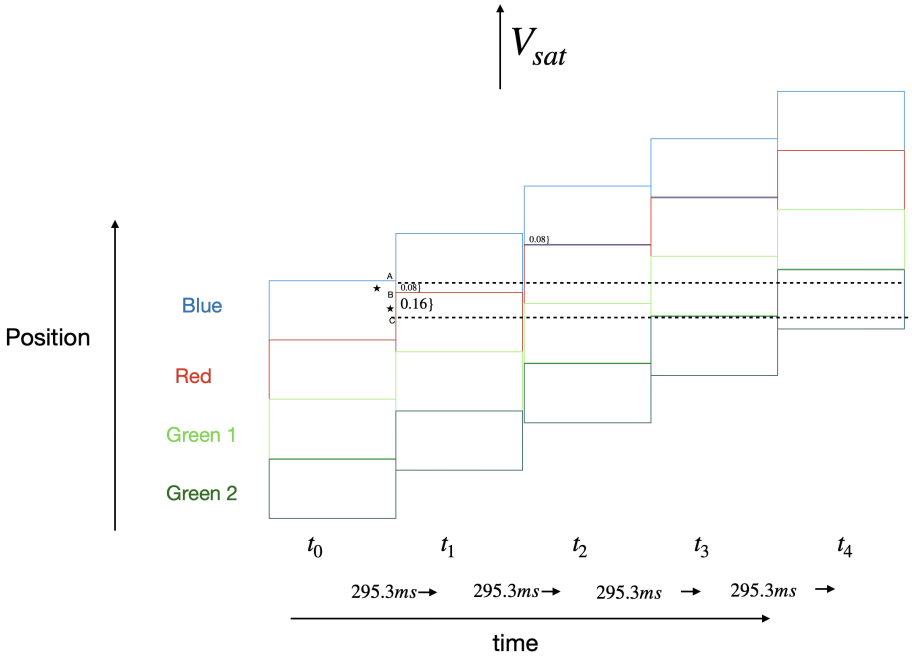


Figure 2.16: visualization: overlap between bands

3 Problem Setup

3.1 Optical Flow

3.1.1 Localization with Laplacian of Gaussian

The Laplacian of Gaussian (LoG) operator Marr and Hildreth [1980] is a classical method for detecting *blobs* in an image that differ in properties such as brightness or texture compared to their surroundings. The LoG operator is defined as:

$$\mathrm{LoG}(x, y; \sigma) = \nabla^2 \left(G(x, y; \sigma) * I(x, y) \right), \tag{3.1}$$

where $I(x, y)$ is the input image, $G(x, y; \sigma)$ is the 2D Gaussian kernel with standard deviation σ, and $*$ denotes convolution.
The Laplacian operator ∇^2 computes the sum of second order derivatives in both spatial directions, producing strong responses in regions where the image intensity changes sharply in all directions. This makes it well suited for blob like structures. The scale parameter σ controls the size of the features to which the filter is most sensitive.
Direct computation of Equation 3.1 can be expensive, so in many computer vision pipelines it is approximated by the *Difference of Gaussians* (DoG). This is equivalent to subtracting two Gaussian blurred versions of the image at different scales.

$$\mathrm{DoG}(x, y; \sigma_1, \sigma_2) = G(x, y; \sigma_1) * I(x, y) - G(x, y; \sigma_2) * I(x, y), \tag{3.2}$$

with $\sigma_2 > \sigma_1$. When $\sigma_2 \approx k\sigma_1$ for a constant k, this becomes a scaled LoG:

$$\mathrm{DoG}(x, y; \sigma, k) \approx c \cdot \sigma^2 \cdot \nabla^2 G(x, y; \sigma), \tag{3.3}$$

where c depends only on k Lowe [2004]. In our application, the number of scales is relatively small, as we are interested in small blobs only. So the computational advantage of DoG is less critical. We therefore compute LoG directly, avoiding any approximation error introduced by DoG.

To capture blobs of varying size, the LoG response is evaluated over multiple σ values, forming a *scale-space representation*. Local extrema are then detected in both spatial and scale dimensions:

$$\hat{\sigma}, \hat{x}, \hat{y} = \arg\max_{\sigma, x, y} \mathrm{LoG}(x, y; \sigma), \tag{3.4}$$

where each maximum corresponds to a blob center, and $\hat{\sigma}$ relates to its characteristic radius ($r \approx \sqrt{2}\,\hat{\sigma}$).

We use Laplacian of Gaussian (LoG) method as implemented in the `scikit-image` Python library Van der Walt et al. [2014]. We choose $\sigma \leq 2.5$ because, beyond this value, neighboring blobs begin to merge and the detected scale becomes too large for our target objects. .

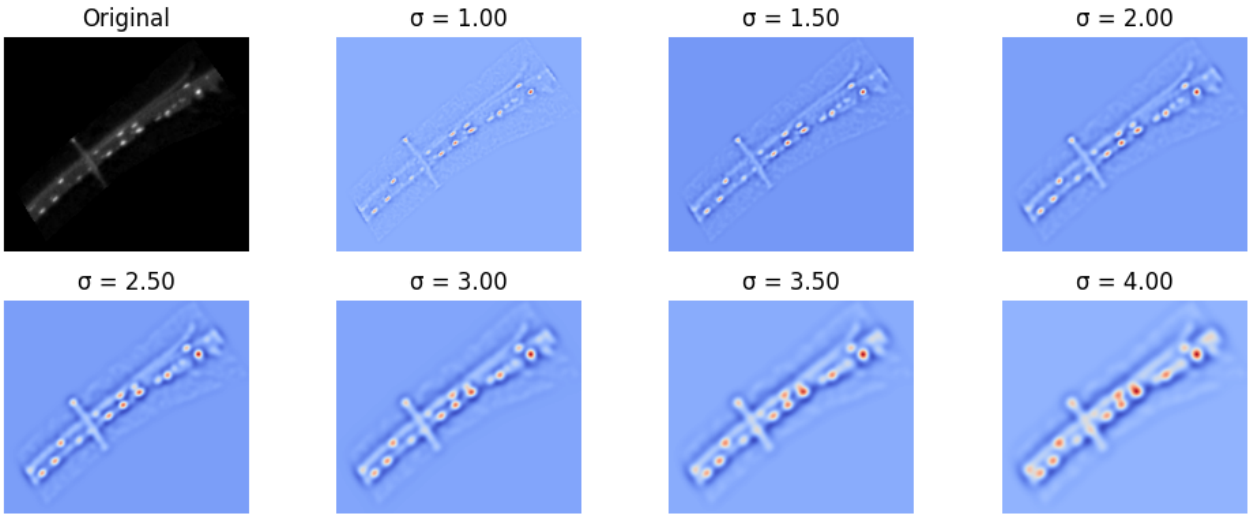
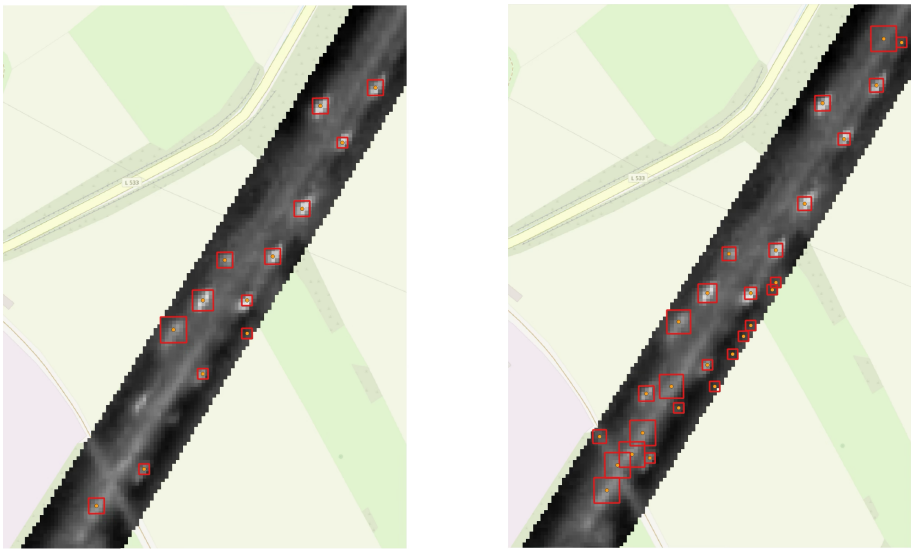


Figure 3.1: loG applied on blue channel of satellite imagery at varying sigmas. We choose scales upto $\sigma = 2.5$

Not all extrema correspond to valid targets. Some are noise or background texture. We therefore apply:

- An intensity threshold to remove weak responses.
- An h-maxima transform Vincent [1994] to suppress maxima that are only marginally higher than their surroundings.

The h-maxima step is especially useful, as it allows us to reject outliers without lowering the global threshold too much.
Parameters for filtering maxima from loG cube must be carefully selected to have a correct balance between keypoint recall and outlier supression. We choose $h = 0.07$ and $threshold = 0.07$. These values are highly image dependent and typically require trial and error tuning; parameters that work well for one scene may fail in others, which highlights a robustness limitation of this approach.



threshold = 0.07
&
h_max = 0.07

threshold = 0.07

Figure 3.2: Filtration method for selecting maxima from loG cube.

3.1.2 Lucas - Kanade method

The Lucas-Kanade method Lucas and Kanade [1981] is a widely used algorithm for estimating optical flow between two consecutive image frames. It assumes that the motion of pixels is small and approximately constant within a local neighborhood. The method is built upon the *brightness constancy* assumption, which states that the intensity of a point in the image remains constant over time:

$$I(x, y, t) = I(x + u, y + v, t + 1), \tag{3.5}$$

where $I(x, y, t)$ is the image intensity at spatial coordinates (x, y) and time t, and (u, v) represents the pixel's displacement (optical flow) between frames.

Assuming small motion, a first order Taylor expansion yields the *optical flow constraint equation*:

$$I_x u + I_y v + I_t = 0, \tag{3.6}$$

where $I_x = \frac{\partial I}{\partial x}$, $I_y = \frac{\partial I}{\partial y}$, and $I_t = \frac{\partial I}{\partial t}$ are the spatial and temporal image gradients.

Equation (3.6) is underdetermined for a single pixel (one equation, two unknowns). Lucas and Kanade addressed this by assuming constant motion within a local window of n pixels, leading to an overdetermined system:

$$A \begin{bmatrix} u \\ v \end{bmatrix} = -\mathbf{b}, \tag{3.7}$$

where $A \in \mathbb{R}^{n\times 2}$ contains the spatial gradients $[I_x, I_y]$ at each pixel in the window, and $\mathbf{b} \in \mathbb{R}^n$ contains the corresponding temporal gradients $-I_t$. The least-squares solution minimizing the error across all pixels is:

$$\begin{bmatrix} u \\ v \end{bmatrix} = -(A^\top A)^{-1} A^\top \mathbf{b}. \tag{3.8}$$

This method is computationally efficient and forms the basis of many optical flow and tracking systems. Its performance can be enhanced using a multi-scale pyramidal approach or iterative refinement to accommodate larger displacements.Bouguet et al. [2001] . We use a 5×5 pixel local window and pyramid levels 0 and 1 for solving the Lucas–Kanade equations. We use the openCV implementation. Bradski [2000]

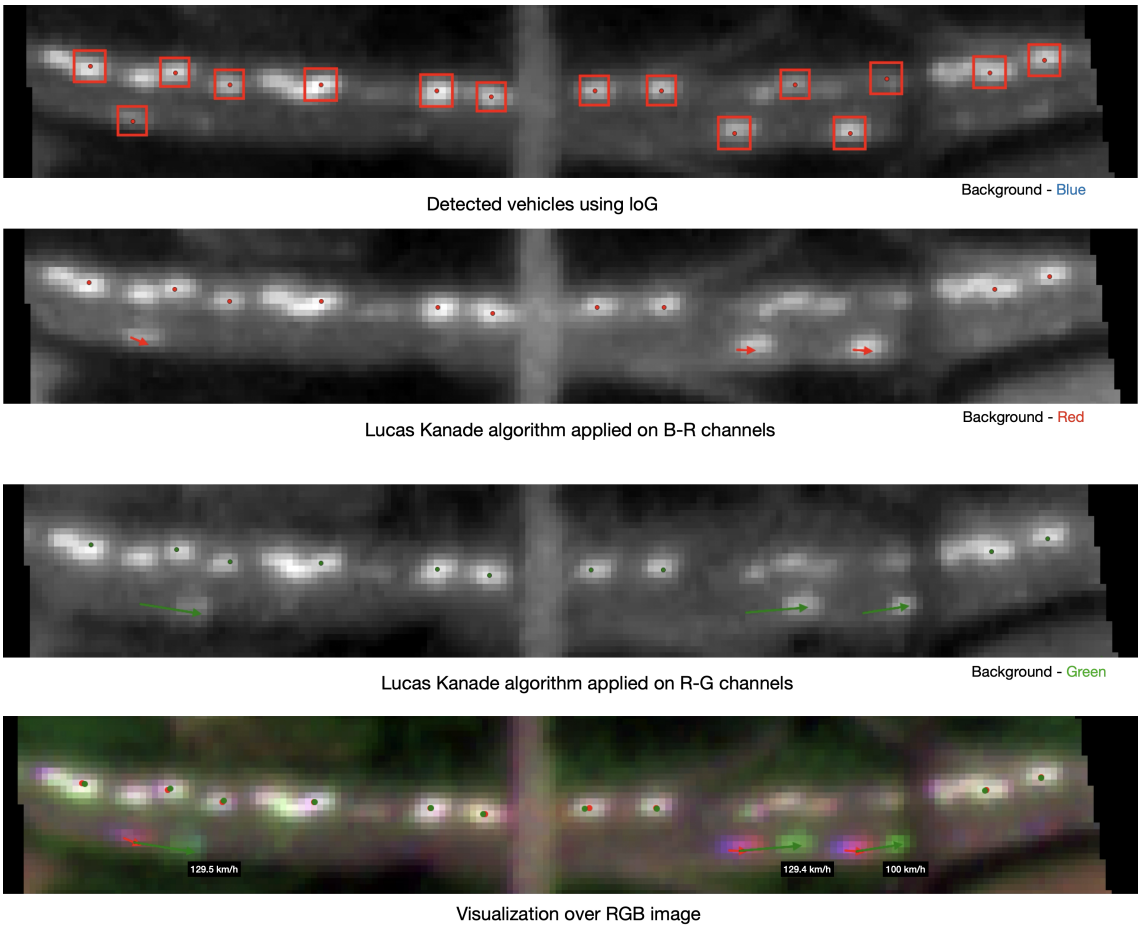


Figure 3.3: Visualization of Lucas-Kanade optical flow algorithm.

Errors can be induced especially when vehicles are very close to each other, and the flow might jump to the vehicle behind or besides on other lane. Thus we do

post-processing, to remove unrealistic trajectories, where the joint angle is extremely sharp.

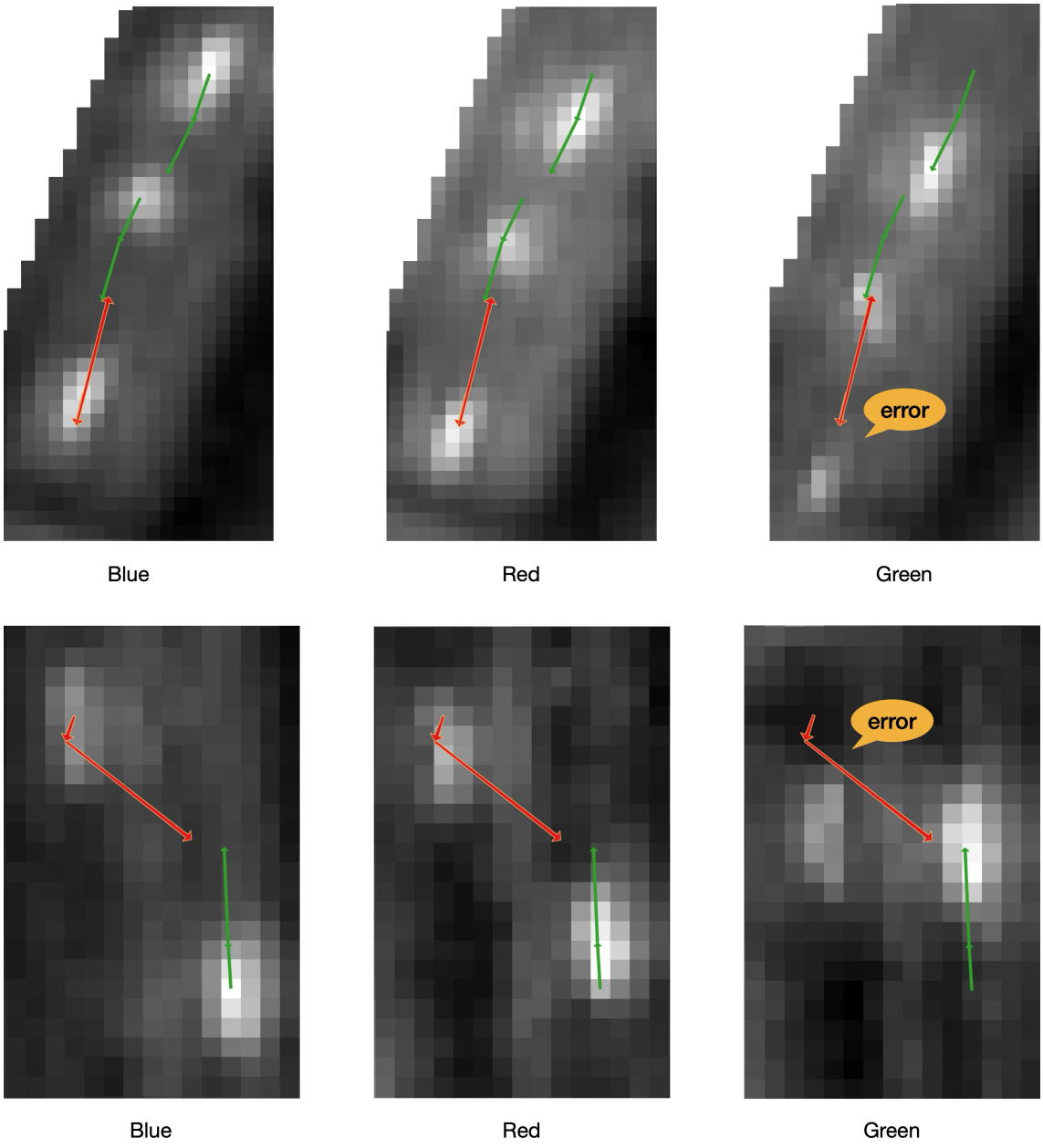


Figure 3.4: nature of errors during optical flow computation.

The eigenvalues (λ_1, λ_2) shown in Figure 3.5 come from the structure tensor $A^\top A$ computed within each 5×5 window. Points where both eigenvalues are large indicate strong texture in two orthogonal directions, which makes them well suited for reliable optical flow estimation. Conversely, low values in one or both directions indicate regions (e.g., edges or flat areas) where motion cannot be estimated accurately, leading to potential tracking errors.

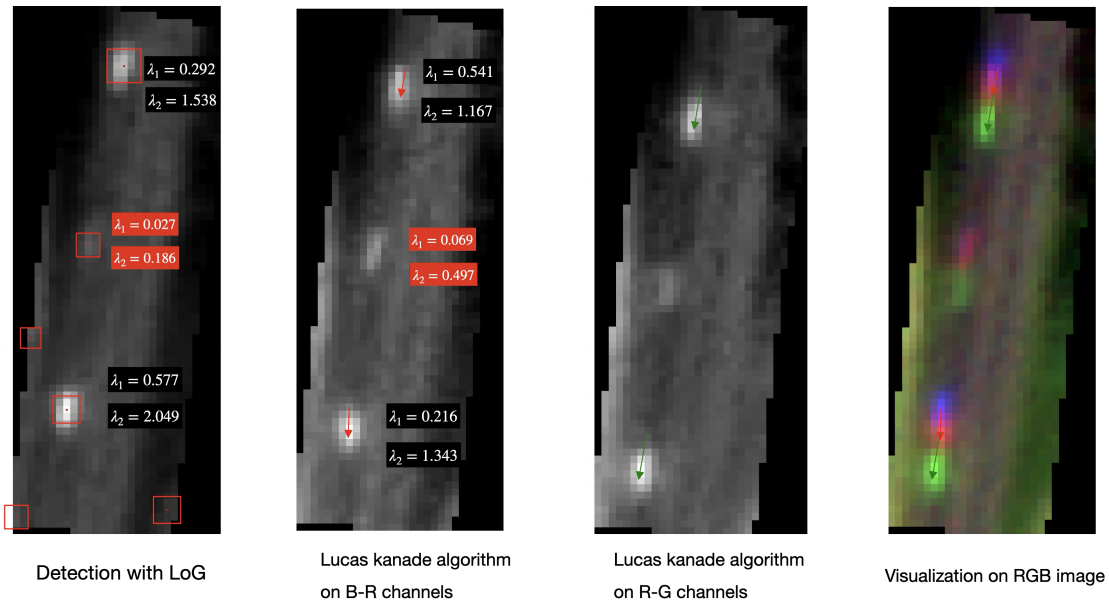


Figure 3.5: Structure tensor eigenvalues (λ_1, λ_2) at keypoints: Points with large values in both directions indicate strong textured regions, which are well suited for optical flow tracking.

3.2 Keypoint Modeling

We frame our problem as a keypoint estimation task, where vehicles in each spectral band are represented as keypoints. The distance between blue–red and red–green keypoints gives the average distance traversed by a vehicle. We define three label types:

- **Label 1:** Vehicles with no detectable motion, represented by a single keypoint shared across all channels.
- **Label 2:** Vehicles where the B–R shift is minimal or ambiguous due to small displacements, represented by a single keypoint for the blue and red channels and a second keypoint for the green channel.
- **Label 3:** Fast-moving vehicles with clear displacement across all channels.

The modeling details are discussed in following subsection.

3.2.1 Architecture

We adopt a modified Mask R-CNN architecture He et al. [2017], which is a top-down architecture, designed to jointly perform object detection and keypoint localization within a single network. The backbone extracts shared feature maps, which are

then used by both the detection and keypoint branches, enabling efficient end-to-end training.
Object detection is performed using a two-stage approach: a Region Proposal Network (RPN) first generates candidate proposals, which are then refined by the box head. For each detected object, keypoint head predicts the spatial positions of predefined keypoints within the bounding box.

The overall architecture is tailored for our application. The backbone is modified to process four-channel RGB+NIR imagery and optimized for small-object detection. The downstream subnetworks—RPN, box head, and keypoint head—are trained jointly with a composite loss that combines detection, keypoint, and shape prior terms. Specifically, we replace the binary cross-entropy loss with focal loss in the classification branches of the RPN, box head, and keypoint head. In the keypoint head, we further incorporate soft-label ground truths for the heatmap focal loss, and introduce an additional shape prior loss terms to improve keypoint consistency.

An overview is shown in Figure 3.6, and detailed descriptions are provided in the following subsections He et al. [2017].

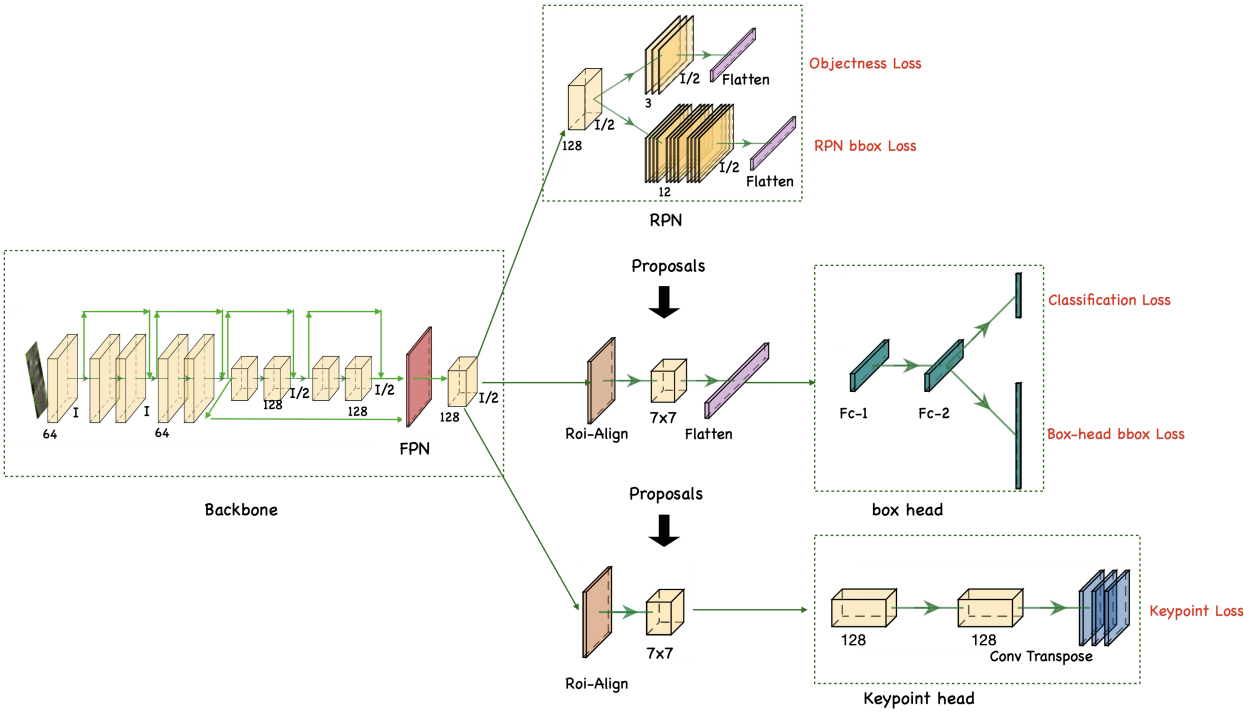


Figure 3.6: Architecture.

3.2.1.1 Backbone

We use a modified ResNet-18 backbone, pretrained with Imagenet weights. We use `conv0`, `layer1`, and `layer2` blocks. This is because, vehicles in our dataset are very small, occupying an area of roughly 15×15 pixels. Since there are'nt any high level

features, we remove further layers. We ensure that sufficient receptive field is intact, which is 27×27 pixels, comfortably exceeding the object size.

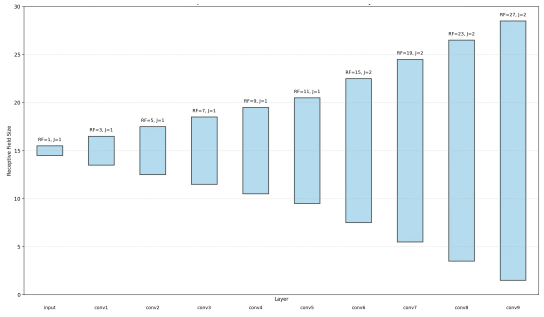
The input is four channel with red, green, blue, and near-infrared (NIR) bands. Since the pretrained ResNet-18 expects a three-channel RGB input, we replace the initial convolution weights with He initialization to accommodate the fourth NIR channel. The inclusion of NIR band is motivated by its ability to enhance road–background separation. In NIR imagery, vegetated areas surrounding highways appear bright relative to the darker asphalt. This helps suppress false positives caused by bright background objects outside the roadway.
For image normalization,we use our training dataset statistics. These are:

$$\text{image\_mean} = \{0.288, 0.316, 0.244, 0.427\},$$
$$\text{image\_std} = \{0.219, 0.199, 0.191, 0.229\}.$$

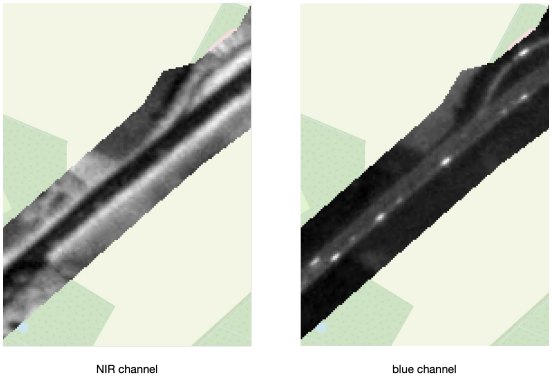
Since these values differ significantly from the ImageNet statistics, we use this instead.

We replace the default 7×7 convolution in `conv0` with a 3×3 kernel, again using He initialization. Additionally, we remove both the stride and the max-pooling operation from `conv0`, preserving the spatial resolution of image at the output of this stage. `Layer1` by default uses stride = 1 and no pooling, so the resolution is maintained. `Layer2` begins with stride = 2, halving the resolution. We retain this reduction to increase the receptive field.

The feature maps from `layer1` and `layer2` are then fused via a Feature Pyramid Network (FPN) Lin et al. [2017a]. Each feature map is first passed through a 1×1 convolution to ensure consistent channel dimensions, followed by a 3×3 convolution for smoothing. Both stages produce 128-channel feature maps, which are then merged. The resulting fused feature map at 1/2 of the input resolution is used by the downstream sub-networks: Region Proposal Network (RPN), bounding box head, and keypoint head.



(a) Receptive field for the backbone.



(b) NIR band is passed alongside RGB to help reduce outliers. It helps to delienate roads

Figure 3.7

3.2.1.2 Region Proposal Network

RPN is responsible for generating candidate proposals that likely contain objects, regardless of their specific class. For each location in the feature map, the RPN predicts two things: an *objectness score*—the probability that the region contains an object of any class, and a set of bounding box regression offsets to refine the anchor position.

A key property of the RPN is *translation invariance*. If an object is shifted in the image, the corresponding proposal also shifts with it. This is achieved through the use of anchors and convolutional operations, both of which are inherently translation-invariant Ren et al. [2015].

In our setup, each feature map location has three anchor boxes, with a single scale of 10 pixels and aspect ratios $[0.5, 1.0, 2.0]$. An anchor is considered a positive sample if it has an Intersection-over-Union (IoU) of at least 0.5 with a ground truth bounding box. This threshold is chosen to suit our small objects, ensuring enough positive matches for training.

The RPN head consists of a single 3×3 convolutional layer for feature extraction, followed by two sibling 1×1 convolutions for classification and regression, respectively. We replace the standard binary cross-entropy loss in the classification branch with *focal loss* to better handle the high foreground–background imbalance, while retaining the Smooth L1 loss for bounding box regression. During training, 512 anchors are sampled per image in each iteration. We use the default sampling strategy which tries to maintain a balanced ratio of foreground and background to stabilize learning.

3.2.1.3 RoIAlign

Region of Interest (RoI) operations are used to extract fixed size feature maps from variable sized proposals so that they can be processed by box head and keypoint head. In earlier methods such as RoIPool, the continuous coordinates of the RoI in the feature map were quantized to discrete grid boundaries before pooling. This introduced misalignment between the extracted features and the original image, which is harmful for keypoint localization, which requires pixel level accuracy.

RoIAlign He et al. [2017] eliminates the misalignment caused by rounding RoI coordinates to discrete pixels. Instead of snapping the region boundaries to the nearest grid location, it samples the feature map at precise, fractional coordinates. For each bin, four points are chosen at regular positions, and their values are obtained using bilinear interpolation from the surrounding pixels. This preserves exact spatial alignment between the RoI and the extracted features, which improves accuracy for per pixel level classification tasks like keypoint estimation. We extract 7×7 feature map per RoI instead of the default 14×14) to reduce overfitting .

3.2.1.4 Box Head

The box head takes the 7×7 feature maps from RoIAlign and classifies the object and refine its bounding box. It consists of two fully connected (MLP) layers, each with a feature size of 512 (reduced from the default 1024 to reduce overfitting) From these features, two parallel branches are used: a classification branch that predicts the object class, and a regression branch that outputs four bounding box offsets for each proposal from RPN.

3.2.1.5 Keypoint Head

The keypoint head is responsible for predicting a spatial heatmap for each keypoint within a detected object. It takes feature map from ROI align as input. Again, similar to box head, e use 7×7 map. The head consists of two convolutional layers followed by a transposed convolution layer that upsamples the feature map to 14×14. Each output channel corresponds to a single keypoint, with the predicted values trained using the heatmap-based focal loss and shape priors described in Section 3.2.2.3.

3.2.2 Loss Function

Each of the modules discussed above has its own loss function. The total loss is computed as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{RPN}} + \mathcal{L}_{\text{bbox}} + \mathcal{L}_{\text{keypoint}}.$$

Here, $\mathcal{L}_{\text{RPN}}$ includes the *objectness loss*, which classifies whether a given region proposal contains an object, and the bounding box regression loss to refine anchor coordinates. $\mathcal{L}_{\text{bbox}}$ contains both the classification loss, predicting object classes,

and regression loss further refining the proposals into final detections. $\mathcal{L}_{\text{keypoint}}$ estimates the locations of keypoints within detected objects, incorporating both heatmap supervision and the geometric prior terms. We discuss about these losses in detail in the following subsection.

3.2.2.1 Classification : Focal Loss

In object detection tasks, class imbalance between foreground and background (or between object classes) is a significant challenge. Focal Loss Lin et al. [2017b] addresses this issue by dynamically scaling the standard cross-entropy loss. They focus training on hard, misclassified examples and down-weighting the contribution of well-classified ones.

For binary classification, the standard cross-entropy loss is defined as:

$$\mathcal{L}_{\text{CE}}(p, y) = -\left[y \log(p) + (1 - y) \log(1 - p)\right], \tag{3.9}$$

where $y \in \{0, 1\}$ is the ground truth label and $p \in [0, 1]$ is the predicted probability for the positive class.

Focal loss introduces a modulating factor $(1 - p_t)^\gamma$ to this loss:

$$\mathcal{L}_{\text{Focal}}(p, y) = -\alpha_t (1 - p_t)^\gamma \log(p_t), \tag{3.10}$$

where $p_t = p$ if $y = 1$, and $p_t = 1 - p$ if $y = 0$. The parameter $\gamma > 0$ controls the focus on hard examples (typically $\gamma = 2$), and α_t is a class-balancing weight to compensate for label imbalance (e.g., $\alpha = 0.25$ for background class).

For multi-class classification, such as in the box head of Mask R-CNN, focal loss generalizes over the softmax probabilities p_c for class c as:

$$\mathcal{L}_{\text{Focal}} = -\sum_{c=1}^{C} \alpha_c (1 - p_c)^\gamma y_c \log(p_c), \tag{3.11}$$

where $y_c \in \{0, 1\}$ is the one-hot encoded ground truth label for class c, and C is the total number of classes.

The key intuition is that correctly classified examples with high confidence (i.e., $p_t \to 1$) receive small loss values due to the $(1 - p_t)^\gamma$ term. This prevents the model from being overwhelmed by the large number of easy negatives and instead focuses gradient updates on hard, informative examples.

In our implementation, focal loss is used for binary classification in the RPN and for multi-class classification in the box head to mitigate class imbalance and improve convergence.

3.2.2.2 Bounding Box Regression: Smooth L_1 Loss

Bounding box regression aims to predict the precise coordinates of object bounding boxes. A robust loss function is critical for stable and accurate training. The

Smooth L_1 loss, also known as the Huber loss, is commonly employed due to its balance between sensitivity to small errors and robustness to outliers.

Formally, for a predicted coordinate x and ground truth coordinate y, the Smooth L_1 loss is defined as:

$$\mathcal{L}_{\text{Smooth } L_1}(x, y) = \begin{cases} \frac{1}{2\beta}(x-y)^2, & \text{if } |x-y| < \beta \\ |x-y| - \frac{\beta}{2}, & \text{otherwise} \end{cases} \tag{3.12}$$

This piecewise function behaves like the mean squared error (L2 loss) when the prediction error is small (less than 1), encouraging precise localization by penalizing small deviations quadratically. For larger errors, it transitions to a linear penalty similar to the mean absolute error (L1 loss), reducing the influence of outliers and stabilizing training.

In many deep learning libraries, including the PyTorch Paszke et al. [2019] implementation, the default is $\beta = 1$, yielding the commonly used form:

$$\mathcal{L}_{\text{Smooth } L_1}(x, y) = \begin{cases} 0.5(x-y)^2, & \text{if } |x-y| < 1 \\ |x-y| - 0.5, & \text{otherwise} \end{cases}$$

Using Smooth L_1 loss over pure L2 loss mitigates the issue of exploding gradients caused by large errors early in training, while still maintaining high sensitivity for fine adjustments as predictions approach ground truth.

In this work, Smooth L_1 loss is applied to both the Region Proposal Network (RPN) and the final bounding box regression head to optimize the predicted box coordinates.

3.2.2.3 Keypoint Estimation: Heatmap with Geometric Priors

keypoint estimation can be framed as multi class classification problem, where all the pixels in heatmap corresspond to classes, and the keypoint containing pixel is the positive class.

A challenge with cross-entropy loss is that it enforces overly confident predictions, even in ambiguous regions. This is especially problematic in our setting: vehicles appear as small blobs without high-level features, and in many cases, intensity peaks are weak or absent. Ground-truth labelling is also not perfectly pixel level accurate. Neighboring pixels can be equally valid as the keypoint. To address this, we use *soft labelling*. Müller et al. [2019], Lukasik et al. [2020].

The ground-truth pixel is assigned a weight of 0.5, and each of its eight neighbors (N_8) is assigned 0.0625. This provides a positive signal for both the center and its immediate neighborhood, improving robustness when the exact pixel is uncertain. This strategy is similar to approaches used in pose estimation models such as HR-Net Wang et al. [2020] and Stacked Hourglass Newell et al. [2016], where Gaussian heatmaps are trained with MSE loss, or in some recent works with wing loss Feng et al. [2018] and adaptive wing lossWang et al. [2019]. In our case, the same concept

is adapted to classification.
We replace cross-entropy with *focal loss*, using $\alpha = 1.0$ and $\gamma = 2$ for the keypoint and its neighbors, and $\alpha = 0.25$ for background pixels. This biases the model to focus more on the sparse positive pixels than the many negatives in the 14×14 heatmap.
In addition, we introduce geometric prior terms (discussed in the next section) to further guide keypoint estimation. Computing these priors requires decoding the keypoint location from the predicted heatmap. Since the argmax operation is not differentiable, we use *soft-argmax* Sun et al. [2018], Nibali et al. [2018] instead:

$$\hat{\mathbf{x}} = \sum_i \mathbf{p}_i \cdot \frac{\exp\left(\frac{\mathbf{h}_i}{\tau}\right)}{\sum_j \exp\left(\frac{\mathbf{h}_j}{\tau}\right)}, \tag{3.13}$$

where $\mathbf{p}_i$ is the pixel coordinate, $\mathbf{h}_i$ is the predicted heatmap value, and $\tau = 0.25$, is the temperature parameter controlling peak sharpness. Intuitively, this computes a weighted average of all pixel coordinates, giving pixel keypoint locations while remaining differentiable.

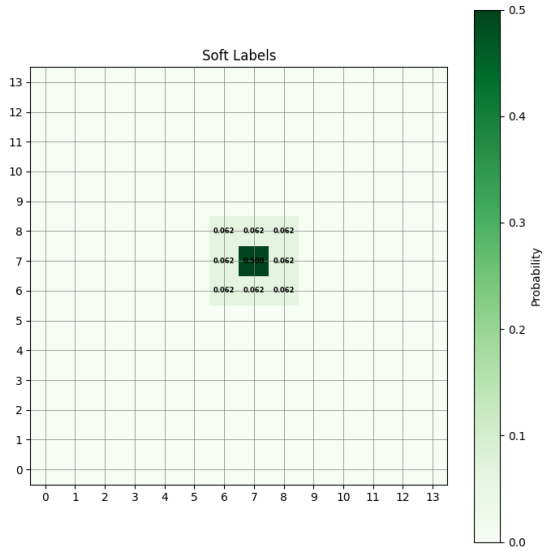


Figure 3.8: Ground Truth soft labelling.

3.2.2.4 Distance Loss

This loss penalizes per keypoint deviation between prediction and ground truth. Let $\mathcal{V}$ be the set of visible keypoints (over batch), and let $H_\beta(\cdot)$ denote the Smooth L1 applied elementwise and averaged over the 2D coordinates.

$$\mathcal{L}_{\text{dist}} = \frac{1}{|\mathcal{V}|} \sum_{(n,k)\in\mathcal{V}} H_\beta\big(\mathbf{p}_{nk} - \mathbf{g}_{nk}\big). \tag{3.14}$$

3.2.2.5 Pairwise segment Loss

This loss enforces consistency of the pairwise displacement vectors between keypoints, ensuring that the predicted segment lengths and directions match the ground truth. Here $(0, 1)$ corresponds to the Blue–Red displacement, $(1, 2)$ to Red–Green, and $(0, 2)$ to Blue–Green. The loss is computed only for pairs where both keypoints are visible.

Let $\mathcal{S} = \{(0, 1), (1, 2), (0, 2)\}$ be the set of keypoint pairs and $\mathcal{S}_{\text{vis}}$ the subset where both endpoints are visible. The segment loss is given by:

$$\mathcal{L}_{\text{seg}} = \frac{1}{|\mathcal{S}_{\text{vis}}|} \sum_{(i,j)\in\mathcal{S}_{\text{vis}}} H_\beta\Big((\mathbf{p}_j - \mathbf{p}_i) - (\mathbf{g}_j - \mathbf{g}_i) \Big), \tag{3.15}$$

where $H_\beta(\cdot)$ denotes the Smooth L1 (Huber) loss with transition point β.

3.2.2.6 Joint angle Loss

While the distance and segment losses ensure that keypoints are close and segment lengths are preserved, they do not guarantee that the relative orientation of the points is correct. The angle loss addresses this by enforcing that the predicted triplet of points forms the same angle as the ground truth.

Consider a triplet $(i, j, k) = (0, 1, 2)$ where j is the central point. We first compute the normalized direction vectors from j to i and j to k:

$$\mathbf{u}_1^{\text{pred}} = \frac{\mathbf{p}_i - \mathbf{p}_j}{\|\mathbf{p}_i - \mathbf{p}_j\| + \varepsilon}, \quad \mathbf{u}_2^{\text{pred}} = \frac{\mathbf{p}_k - \mathbf{p}_j}{\|\mathbf{p}_k - \mathbf{p}_j\| + \varepsilon}, \tag{3.16}$$

and similarly for $\mathbf{u}_1^{\text{gt}}$, $\mathbf{u}_2^{\text{gt}}$ from ground truth.

The cosine of the angles at j are then:

$$\cos\theta_{\text{pred}} = \langle \mathbf{u}_1^{\text{pred}}, \mathbf{u}_2^{\text{pred}} \rangle, \quad \cos\theta_{\text{gt}} = \langle \mathbf{u}_1^{\text{gt}}, \mathbf{u}_2^{\text{gt}} \rangle. \tag{3.17}$$

We also compute the sines via:

$$\sin\theta_{\text{pred}} = \sqrt{1 - \cos^2\theta_{\text{pred}} + \varepsilon}, \quad \sin\theta_{\text{gt}} = \sqrt{1 - \cos^2\theta_{\text{gt}} + \varepsilon}. \tag{3.18}$$

Finally, the cosine of the difference between the predicted and ground truth angles is:

$$\cos(\theta_{\text{pred}} - \theta_{\text{gt}}) = \cos\theta_{\text{pred}} \cdot \cos\theta_{\text{gt}} + \sin\theta_{\text{pred}} \cdot \sin\theta_{\text{gt}}. \tag{3.19}$$

The angle loss penalizes deviations from perfect alignment:

$$\mathcal{L}_{\text{ang}} = 1 - \cos(\theta_{\text{pred}} - \theta_{\text{gt}}). \tag{3.20}$$

3.2.2.7 Smoothness Loss

The term penalizes the *second-order difference*, which captures abrupt changes in motion direction Eilers and Marx [1996]. Let $\hat{\mathbf{p}}_0, \hat{\mathbf{p}}_1, \hat{\mathbf{p}}_2$ denote three consecutive predicted keypoints along the trajectory. Mathematically, this term:

$$\hat{\mathbf{p}}_2 - 2\hat{\mathbf{p}}_1 + \hat{\mathbf{p}}_0, \tag{3.21}$$

is the discrete analog of the second derivative (acceleration) and reflects the *curvature* or "jerk" in the point sequence. If the three points lie on a straight line or move with uniform velocity, this term evaluates to zero. Larger values indicate sudden directional changes, which we aim to suppress.

3.2.3 Pre & Post Processing

We apply the same pre & post processing steps to our ground truth annotations and results from the model. This is discussed in detail in the following sub-section.

3.2.3.1 Shift towards h-maxima

Local maxima are defined as connected sets of pixels whose intensity is strictly greater than that of all pixels in their direct neighborhood (N_8). A local maximum M of height h is a local maximum for which the minimal intensity drop required to reach a higher maximum is at least h (i.e., the highest saddle point connecting M to a higher maximum has an intensity of $f(M) - h$).

The h-maxima transform suppresses all local maxima whose height is less than h. It is computed using morphological reconstruction by dilation, starting from the image $f - h$ and iteratively dilating it under the constraint of the original image f until stability is reached. Formally,it is expressed as:

$$\text{HMAX}_h(f) = R_f^{\delta}(f - h)$$

where R_f^{δ} denotes the geodesic reconstruction by dilation of $(f-h)$ under the mask f.

Tuning $h = 0.1$ provides a good balance between detecting true points of interest and suppressing outliers. We use the scikit-image implementation Van der Walt et al. [2014].

To refine our ground truth annotations, we shift each annotation towards the closest h-maxima if it lies within a distance threshold δ, set to 3.0 pixels by default. The shift is performed in the average direction of the N_8 neighboring pixels whose intensities are within 10% of the local maximum. This is illustrated in 3.10

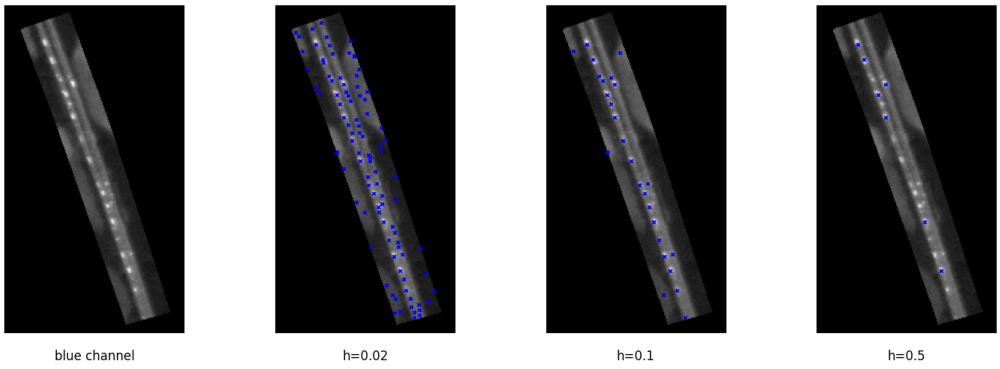


Figure 3.9: Peak detection at different h values; h=0.1 balances detection and outlier reduction.

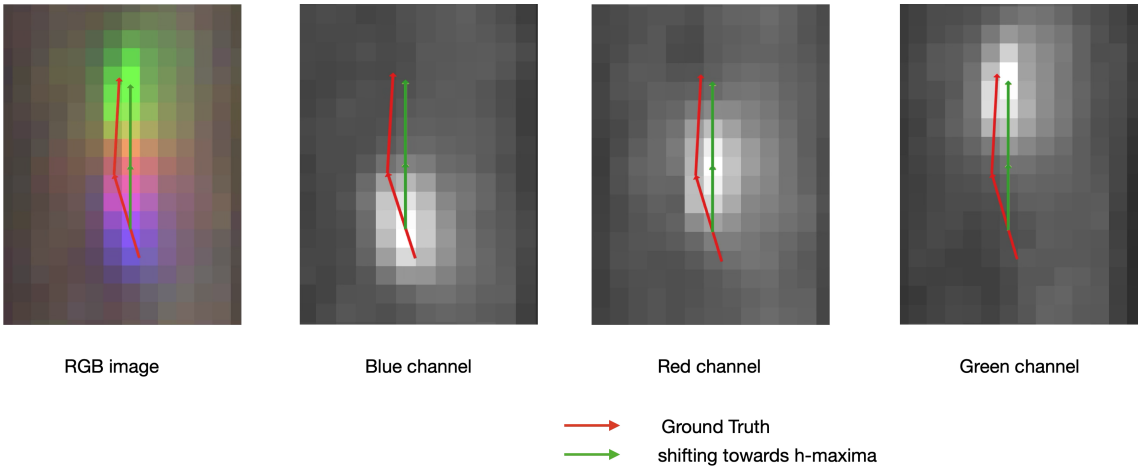


Figure 3.10: Applying shift towards h-maxima to input ground truth annotations.

3.2.3.2 Smoothness

We apply the smoothness loss used in 3.2.2.7 so that the three consecutive points form a gentle curve after the h_maxima correction. Sometimes, after applying the correction, there will be a sharp joint. This can be reduced by minimizing a smoothness term weighted by $\lambda_{\text{smooth}} = 0.1$.
We add two additional terms to ensure that the corrected points do not drift too far off from their original position, and maintain relative lengths between keypoints. This is similar to our discussion in 3.2.2.3 : a distance loss ($\lambda_{\text{dist}} = 0.1$) and a segment length loss ($\lambda_{\text{len}} = 1.0$). For a sequence of three points $P_0, P_1, P_2 \in \mathbb{R}^2$, we define smoothed points $\hat{P}_i$ and minimize the following objective:

Formally, the optimization is

$$\min_{\hat{P}} \lambda_{\text{smooth}} L_{\text{smooth}} + \lambda_{\text{dist}} L_{\text{dist}} + \lambda_{\text{len}} L_{\text{len}}, \tag{3.22}$$

with

$$L_{\text{smooth}} = \|\hat{P}_2 - 2\hat{P}_1 + \hat{P}_0\|^2, \tag{3.23}$$

$$L_{\text{dist}} = \sum_{i=0}^{2} \|\hat{P}_i - P_i\|^2, \tag{3.24}$$

$$L_{\text{len}} = \left(\|\hat{P}_1 - \hat{P}_0\| - \|P_1 - P_0\|\right)^2 + \left(\|\hat{P}_2 - \hat{P}_1\| - \|P_2 - P_1\|\right)^2, \tag{3.25}$$

subject to

$$\hat{P}_{i,j} \in [P_{i,j} - \delta,\ P_{i,j} + \delta], \quad i \in \{0, 1, 2\},\ j \in \{1, 2\}. \tag{3.26}$$

This objective is a small quadratic problem and is efficiently minimized using SciPy's Virtanen et al. [2020] `minimize` function with the L-BFGS-B optimizer. Since each object has only three keypoints per frame, the optimization is lightweight and applied independently per instance. The regularization weight λ controls the trade-off between fidelity and smoothness.

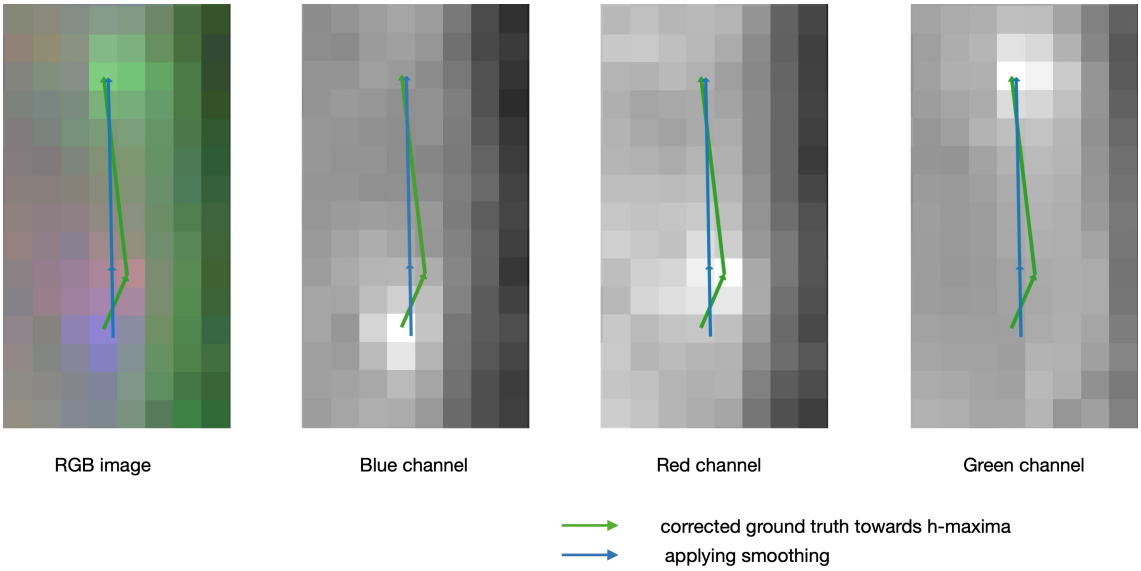


Figure 3.11: Applying smoothing to the corrected ground truth annotations.

3.3 Clustering

3.3.1 Heirarchical DBSCAN

Density-based clustering algorithms identify groups of closely packed points (clusters) while marking points in low-density regions as noise. Two prominent methods in this category are DBSCAN and its hierarchical extension, HDBSCAN Campello et al. [2013].

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) Ester et al. [1996] defines clusters as areas of high point density separated by regions of

low density. It requires two parameters: the neighborhood radius ε and the minimum number of points $minPts$ to define a dense region.

A point x_i is considered a *core point* if at least $minPts$ points fall within its ε-neighborhood:

$$N_\varepsilon(x_i) = \{x_j \mid d(x_i, x_j) \leq \varepsilon\}, \quad \text{with } |N_\varepsilon(x_i)| \geq minPts, \tag{3.27}$$

where $d(\cdot, \cdot)$ denotes the distance metric, typically Euclidean. Clusters are formed by connecting core points and their density-reachable neighbors. DBSCAN is robust to noise and can capture arbitrarily shaped clusters. However, it struggles with datasets containing clusters of varying densities due to its reliance on a fixed ε.

HDBSCAN Campello et al. [2013] improves on DBSCAN by removing the need to specify a fixed ε and instead constructs a hierarchy of clusters based on variable density levels. The key concept is the *mutual reachability distance* between points x_i and x_j:

$$d_{\text{mreach}}(x_i, x_j) = \max\left\{d_{\text{core}}(x_i),\ d_{\text{core}}(x_j),\ d(x_i, x_j)\right\}, \tag{3.28}$$

where $d_{\text{core}}(x_i)$ is the distance from x_i to its k-th nearest neighbor, representing local density.

Using these mutual reachability distances, a minimum spanning tree (MST) is computed to capture the connectivity of the dataset at varying density thresholds. From this MST, a hierarchical cluster dendrogram is constructed. Clusters that persist over a significant range of density levels are considered stable. The cluster stability score is defined as

$$\text{Stability}(C) = \int_{\lambda_{\min}}^{\lambda_{\max}} |C(\lambda)|\, d\lambda, \quad \text{where} \quad \lambda = \frac{1}{\text{distance}}, \tag{3.29}$$

with $|C(\lambda)|$ representing the size of cluster C at density level λ.

HDBSCAN therefore adapts to clusters of varying density, avoids manual parameter tuning, and is robust to noise, making it well-suited for complex real-world data.

We utilize the widely adopted Python implementation of HDBSCAN provided by the `hdbscan` library McInnes et al. [2017].

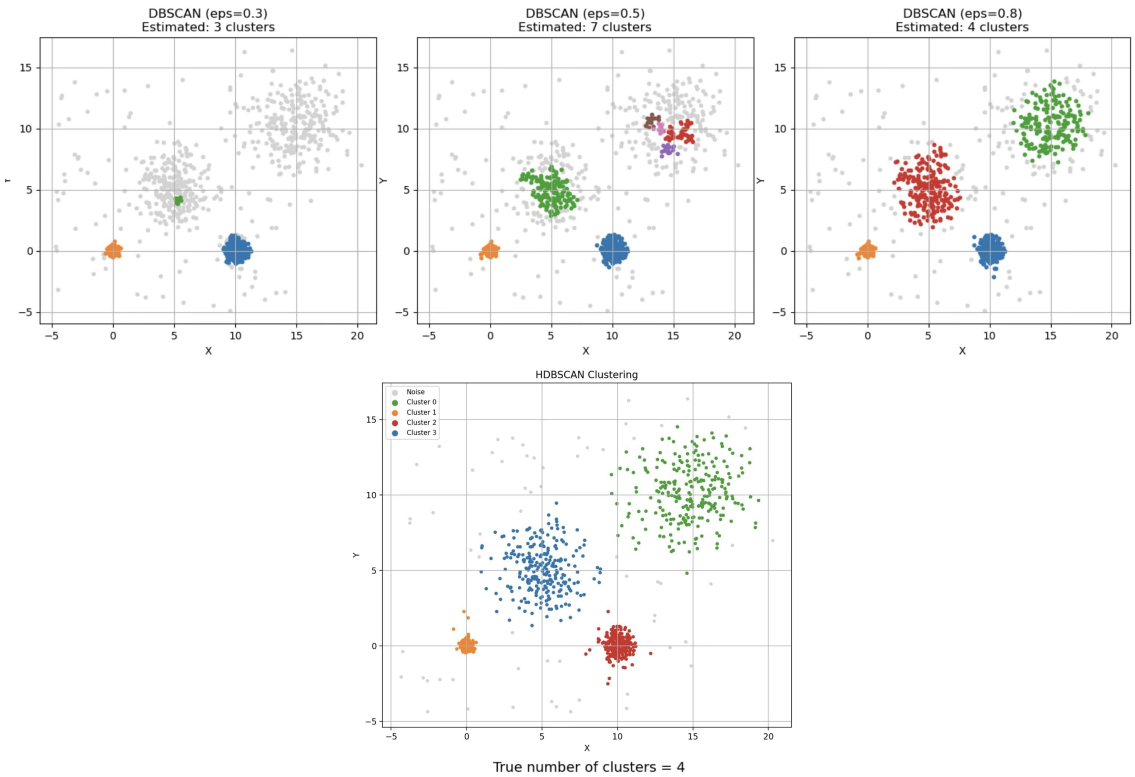


Figure 3.12: DBSCAN vs. HDBSCAN clustering on data with 4 clusters of varying spread (std: 0.2, 1.5, 0.5, 2). HDBSCAN better adapts to density differences by default, while DBSCAN requires careful tuning of ϵ

3.3.2 Principal Component Analysis (PCA) for Path Ordering and Length Estimation

To estimate the length along a curved roadway from scattered points, Principal Component Analysis (PCA) is used to identify the dominant direction along which the data varies most. This dominant axis captures the main orientation of the road shape, regardless of its curvature or rotation. Projecting the points onto this axis provides a linear ordering that approximates the sequence along the path, enabling length estimation by connecting points in this order.

Given a set of 2D points $\mathbf{x}_i = (x_i, y_i)$ for $i = 1, \ldots, n$, we first compute the mean point

$$\mu = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i, \tag{3.30}$$

and center the data by subtracting the mean,

$$\mathbf{z}_i = \mathbf{x}_i - \mu. \tag{3.31}$$

The covariance matrix

$$C = \frac{1}{n}\sum_{i=1}^{n} \mathbf{z}_i \mathbf{z}_i^\top = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{yy} \end{bmatrix} \tag{3.32}$$

captures how the points vary in the 2D space. We then find the eigenvalues $\lambda_1 \geq \lambda_2$ and corresponding eigenvectors $\mathbf{v}_1, \mathbf{v}_2$ by solving

$$C\mathbf{v} = \lambda \mathbf{v}. \tag{3.33}$$

The eigenvector $\mathbf{v}_1$ associated with the largest eigenvalue λ_1 defines the first principal component (PC1), representing the direction of greatest variance in the data.

By projecting each point onto $\mathbf{v}_1$,

$$p_i = (\mathbf{x}_i - \mu)^\top \mathbf{v}_1, \tag{3.34}$$

we obtain scalar values p_i that provide a linear ordering of the points along the dominant axis of the road. Connecting points sequentially according to these ordered projections approximates the path along the curved roadway, simplifying length measurement and analysis. Jolliffe [2011]

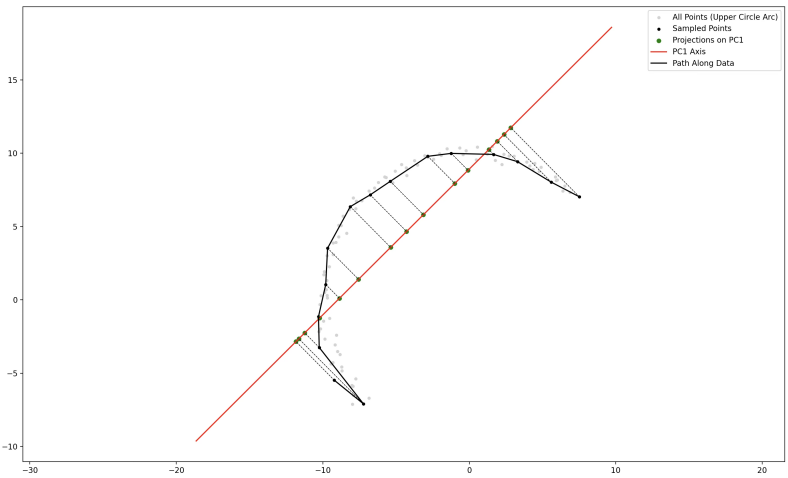


Figure 3.13: Projection of sampled points onto the first principal component axis for ordering. This enables approximate calculation of congestion length along the curved road segment.

Figure 3.14: congestion clusters of varying lengths identified by HDBSCAN.

4 Results

The keypoint model takes small segments of highways as input and predict trajectories. It takes about 2 minutes to compute trajectories for approximately 500 kms ($\sim$ 1500 segments) of road on a macbook M4 pro. Speeds are computed from the trajectories based on the time interval as discussed in 2.4 ($\sim$ 900ms between blue-green bands).

To assign the vehicles into one of the two travel directions, we make use of the centerline trajectory (computed during preparation of input images) of the highway under analysis. For moving vehicles, if their estimated heading is within 90° of the centerline azimuth, they are assigned to one direction; otherwise, they are assigned to the opposite direction. For static vehicles (represented by a single point), the assignment is based on which side of the centerline the vehicle lies.

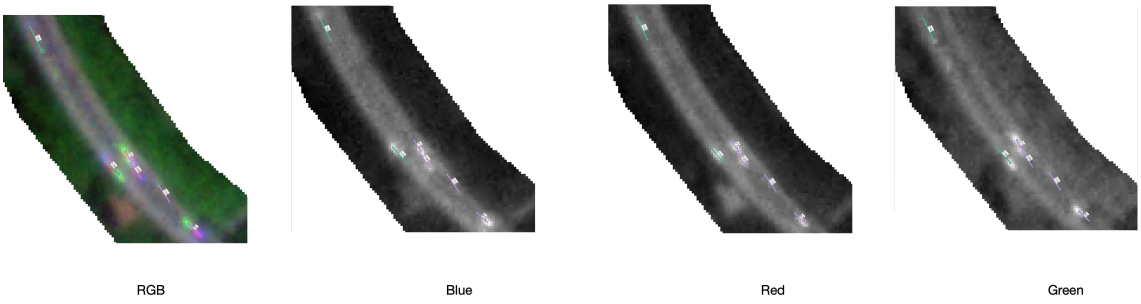


Figure 4.1: Keypoint Model Inference on A3 highway. We show speeds and direction of travel (green vs magenta).

4.1 Model Evaluation

We evaluate the object detection stage using the Average Precision (AP) metric, which summarizes the precision–recall curve into a single value by integrating precision over all recall levels:

$$AP = \int_0^1 p(r)\, dr \tag{4.1}$$

where $p(r)$ is the precision at recall r. AP is computed at an Intersection-over-Union (IoU) threshold of 0.5, meaning a detection is considered correct if:

$$IoU = \frac{|B_p \cap B_{gt}|}{|B_p \cup B_{gt}|} > 0.5 \tag{4.2}$$

where B_p and B_{gt} denote the predicted and ground-truth bounding boxes, respectively.

We report our evaluation metrics below. Label 1 represents static vehicles. We achieve a reasonable $AP_{0.5}$ for this class. Most errors are due to background noise in the imagery, where small bright spots are occasionally mistaken for static vehicles. Label 2 corresponds to vehicles with no shift in the blue-red channel, but a small shift in the red-green channel, representing slow moving vehicles. This class was under represented in the training data, resulting in a lower $AP_{0.5}$. However, our application is tolerant to detecting either. Label 2 or Label 3—(as both are vehicles) as long as the average trajectory length is similar.so this has minimal practical impact.
Label 3 corresponds to "fast" moving vehicles with three keypoints. This is the most common class in the training set, and achieves a very high $AP_{0.5}$. Since all classes correspond to small objects (Label 3: $\approx 15 \times 15$ px; Label 1: $\approx 3 \times 3$ px), $AP_{0.75}$ is of limited relevance in our application, but is included for completeness.

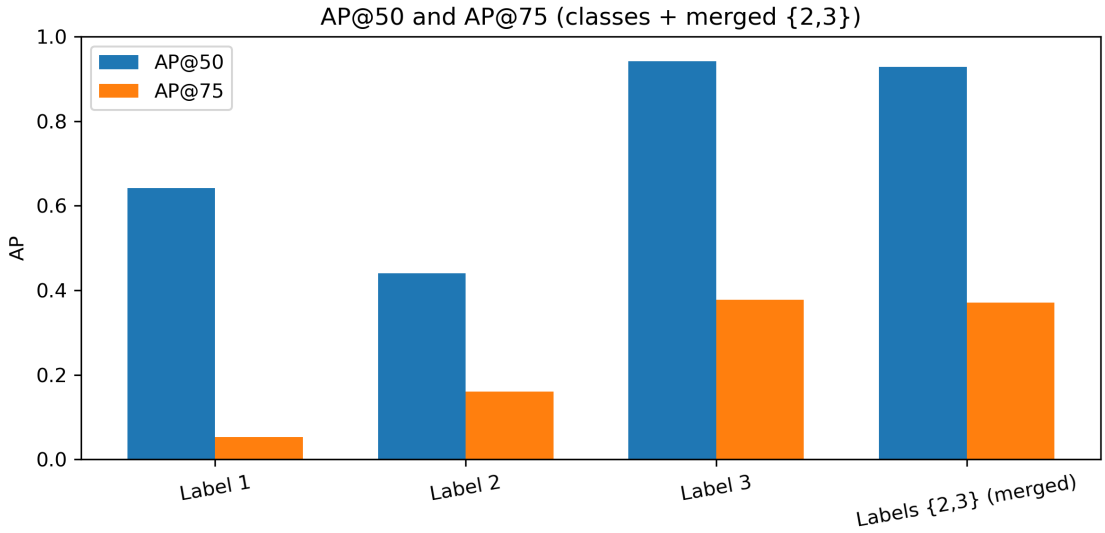
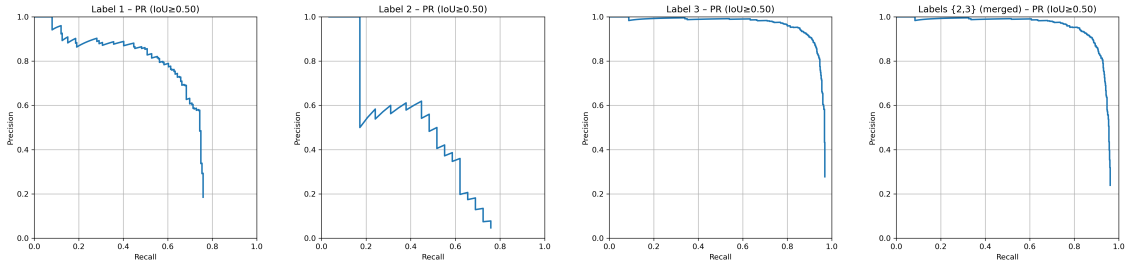


Figure 4.2: $AP_{0.5}$ and $AP_{0.75}$

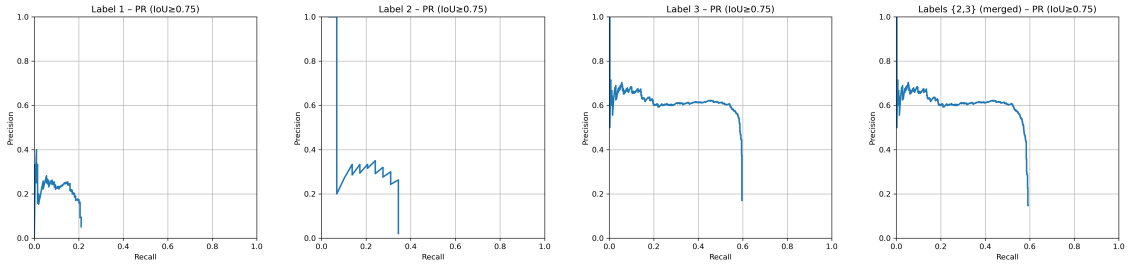
Group	$AP_{0.50}$	$AP_{0.75}$
Label 1	0.64	0.05
Label 2	0.44	0.16
Label 3	0.94	0.38
Labels {2,3} (merged)	0.93	0.37
mAP (macro over 1/2/3)	0.67	0.20

Table 4.1: Average Precision at IoU thresholds.



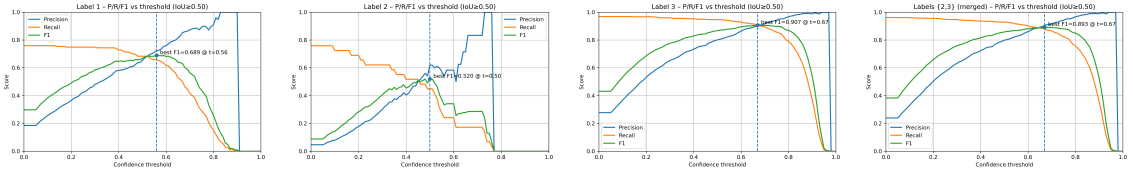
(a) Label 1 (b) Label 2 (c) Label 3 (d) Labels 2–3 (merged)

Figure 4.3: Precision–recall curves showing $AP_{0.50}$ for each category.



(a) Label 1 (b) Label 2 (c) Label 3 (d) Labels 2–3 (merged)

Figure 4.4: Precision–recall curves showing $AP_{0.75}$ for each category.



(a) Label 1 (b) Label 2 (c) Label 3 (d) Labels 2–3 (merged)

Figure 4.5: Threshold sweep results at IoU = 0.50 for each category.

For detections meeting this IoU criterion, we additionally assess keypoint localization accuracy using the Root Mean Squared Error (RMSE) between predicted and ground-truth keypoint coordinates:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{k}_i^{(p)} - \mathbf{k}_i^{(gt)}\right\|_2^2} \tag{4.3}$$

where N is the number of keypoints in the object, $\mathbf{k}_i^{(p)}$ is the predicted coordinate, and $\mathbf{k}_i^{(gt)}$ is the ground-truth coordinate for the i^{th} keypoint. This jointly measures

the correctness of object localization and the precision of keypoint placement within correctly detected objects. The table below shows RMSE on test dataset

Group	RMSE (m)	RMSE (px)	n_pairs
Label 1	3.669102	1.223034	108
Label 2	6.296832	2.098944	44
Label 3	4.785135	1.595045	2142
Overall	4.772278	1.590759	2294

Table 4.2: RMSE per label inside TP boxes (IoU ≥ 0.5).

4.1.1 Ablation

Without Shape Priors We observe that when the keypoint model is trained only with heatmap loss without the shape priors described in Section 3.2.2.3, it struggles in scenarios where vehicles have overlapping proposals, often due to their close proximity. Such overlaps are unavoidable because the proposals are generated perpendicular to the image frame, whereas the road network is typically oriented at an angle. An example is shown in 4.6, where two vehicles appear within the same overlapped region in the green band. Without shape priors, the model lacks awareness of the expected trajectory geometry, and thus assigns high likelihood to both vehicles. As shown in Figure 4.7, this results in a slightly higher likelihood being assigned to the wrong vehicle. 4.7

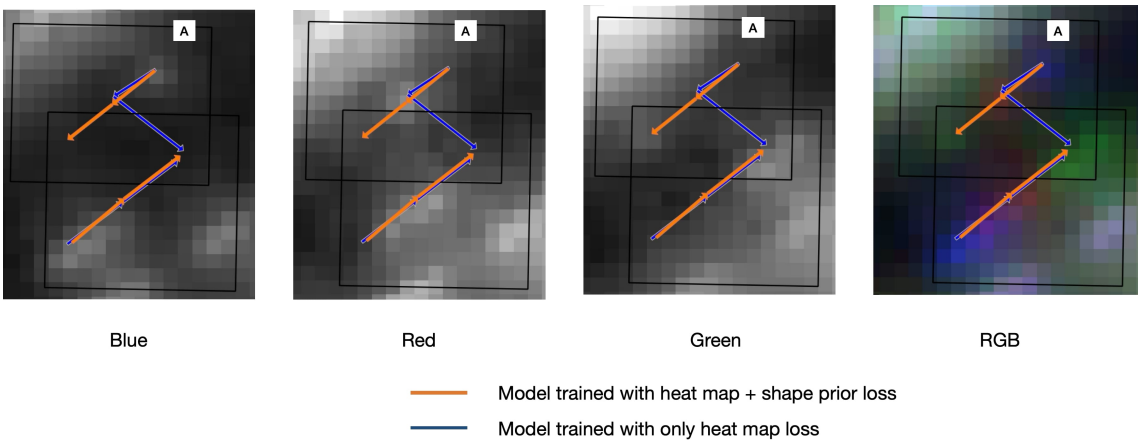
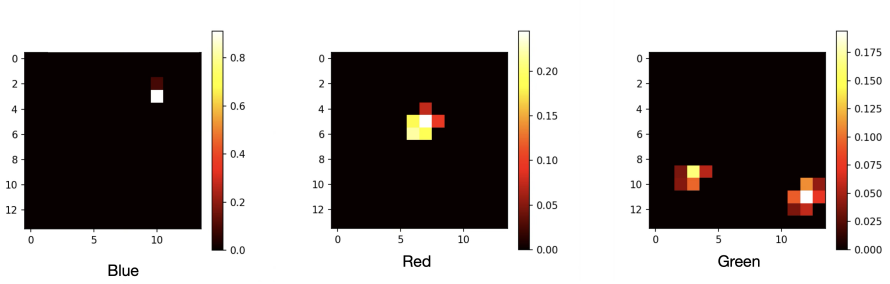
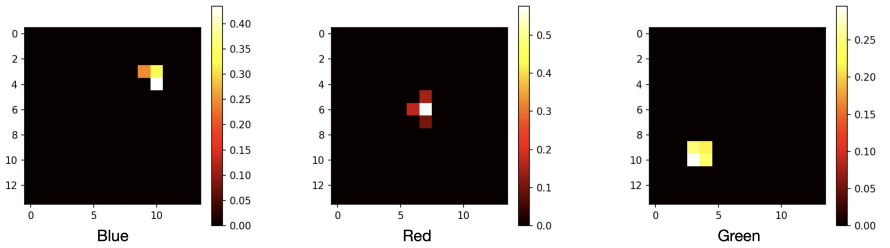


Figure 4.6: model trained without shape priors struggle when proposals overlap. It can get confused with neighbouring vehicle.



(a) model trained with just heatmap loss



(b) model trained with heatmap and shape prior loss

Figure 4.7: Comparision of heatmaps for proposal A from above.

Without Heamap loss If we omit heatmap loss, and use only the geometric prior losses (distance loss,pairwise segment loss smootheness loss, joint angle loss), heatmap for middle keypoint (red band) is not learnt properly. This occurs because, regardless of the trajectory orientation, the middle keypoint almost always lies near the geometric center of the proposal. Without explicit heatmap supervision, the model learns to distribute its likelihood uniformly across directions so that the effective average position lies at the center. As a result, it cannot develop a strong spatial preference for the true middle keypoint location.

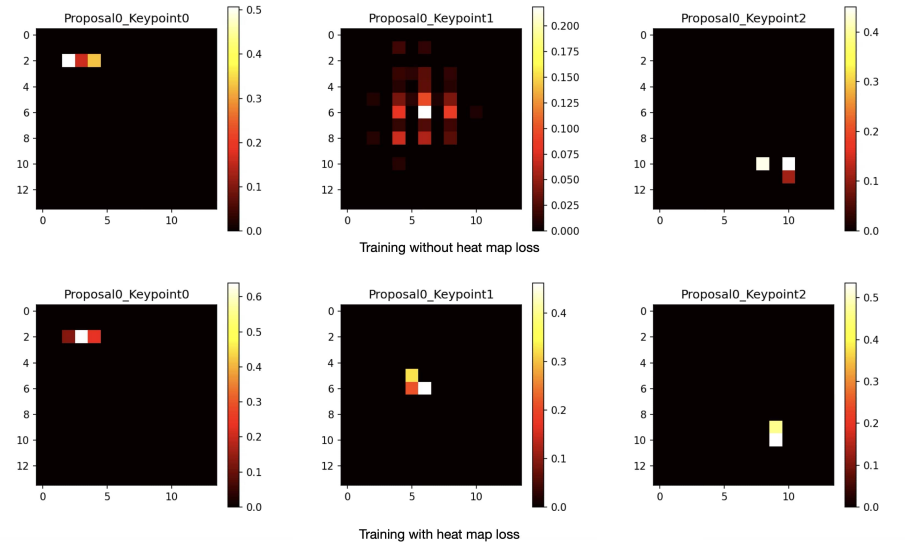
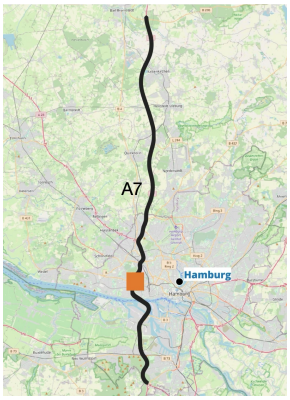


Figure 4.8: Heatmap for middle keypoint is not learnt if heatmap loss is not explicitly used during taining.

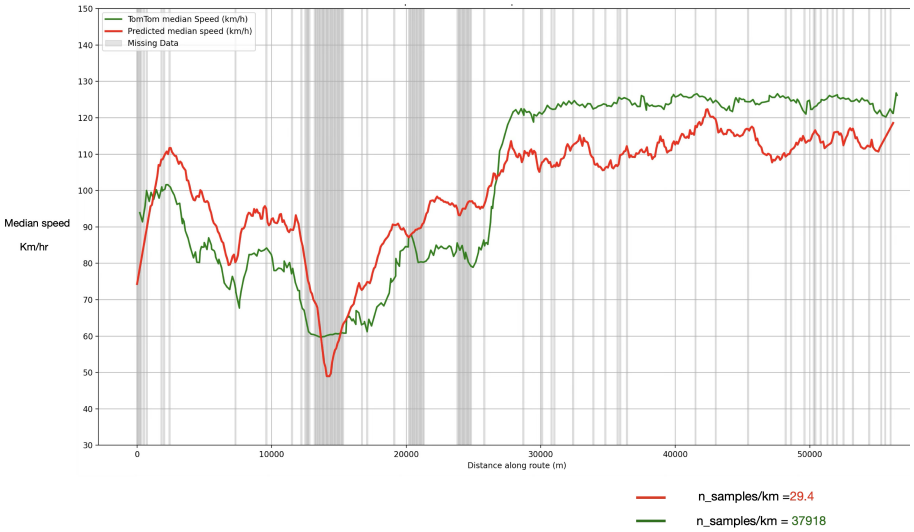
4.2 Validation

We validate our results by downloading traffic speeds data using TomTom Move trial account. TomTom [2025]. We have downloaded planetscope satellite imagery for all major roadways in Germany for the time period May-Aug 2024. Through TomTom 's trial account, we have the access for August 2024 data.
We run our analysis on 3 segments on national highways A7, A2, A3 of lengths 56km, 80km and 17km respectively for the month of August 2024. We run inference with our keypoint model in these highways segments for all the available data during the month of August 2024. Subsequently we download the the traffic statistics report for these routes from TomTom as well.
We run our inference by dividing the highway into 300 metres length segment. We take a buffer of 30m on both sides from the centerline of highway way lines obtained from Openstreet to create an area of interest, which is our input image to the model.
Once our inference is completed, we obtain a resulting geopackage with vehicle trajectories, speeds, and direction of travel(lane 0/1).
We now segmentize the highway into 100m length and compute the median velocities of all the vehicles detected in that segment during the entire month. We compare our median velocities with TomTom data's median velocity, which is available for different small segments(50-200 m). We pick the median velocities during the same time interval as that of our satellite imagery. This time interval is between (9:30 to 11:00 am) . The plots are shown below in 4.9, 4.10, 4.11, which compares median speed along the entire route.

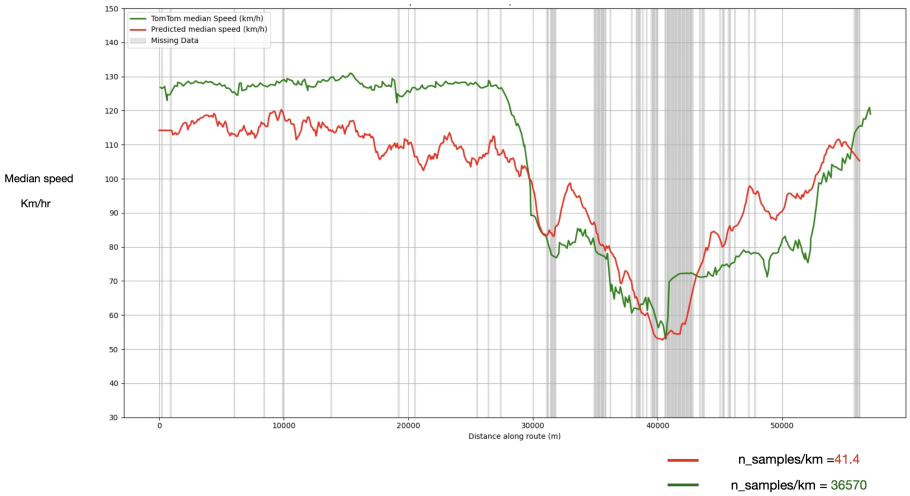
We observe that our speed patterns are in agreement with TomTom's data. In each plot, we see the speeds declining at the same bottleneck areas. At other locations, the speeds are usually within 10-15kmph. On an average, the number of vehicle trajectories that we obtain per km for the entire month in these highway segments are between 25-100. This in comparison to TomTom's data which uses about 35000-40000 per km to estimate its median speeds. Despite such a stark difference in sampling density, the agreement in the speed profile shows the potential of our method to estimate velocities at large scale.



(a) A7 near Hamburg. Total distance 56 km. Direction of travel South→North in 4.9b and North→South in 4.9c.



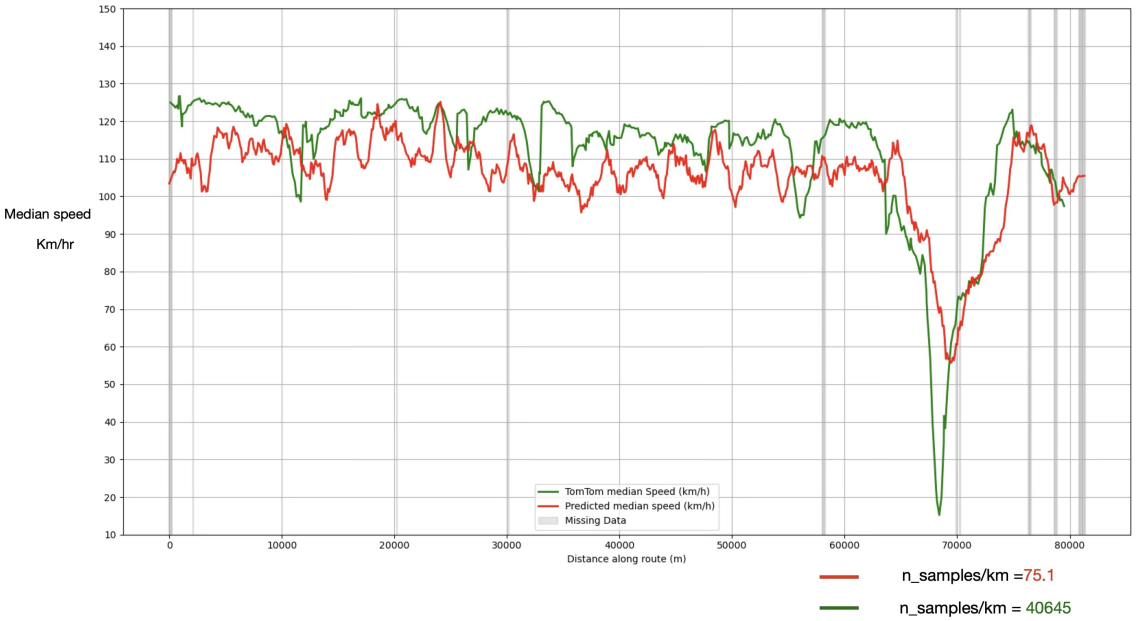
(b) Comparison of monthly median speeds with TomTom data in Aug 24 between 9:30 and 11:00 am over shown route. Direction South→North We smooth predicted results with Savitzky–Golay filter (window=21, p=1).



(c) Comparison of monthly median speeds with TomTom data in Aug 24 between 9:30 and 11:00 am over shown route. Direction : North→South We smooth predicted results with Savitzky–Golay filter (window=21, p=1).

Figure 4.9

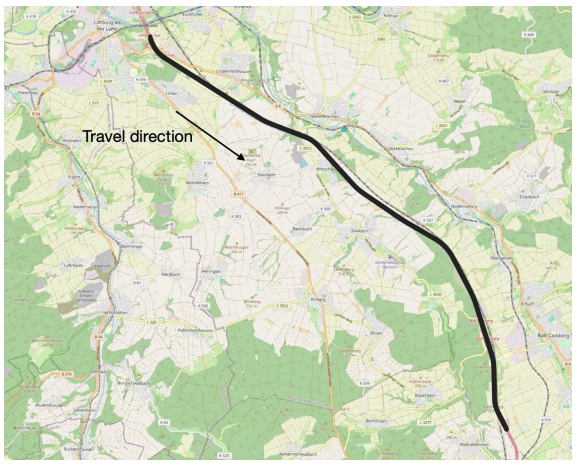
(a) A2 highway near Dortmund. Total distance 80 km. Direction of travel East→West in 4.10b and West→East in 4.10c .



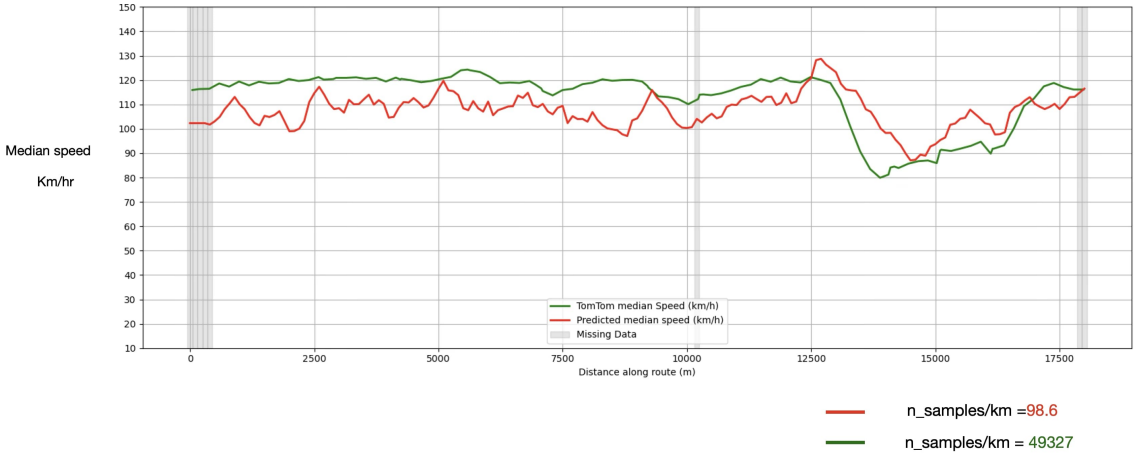
(b) Comparison of monthly median speeds with TomTom data in Aug 24 between 9:30 and 11:00 am over shown route.Direction: East → West. We smooth predicted results with Savitzky–Golay filter (window=9, p=1).

(c) Comparison of monthly median speeds with TomTom data in Aug 24 between 9:30 and 11:00 am over shown route.Direction: West → East We smooth predicted results with Savitzky–Golay filter (window=9, p=1).

Figure 4.10



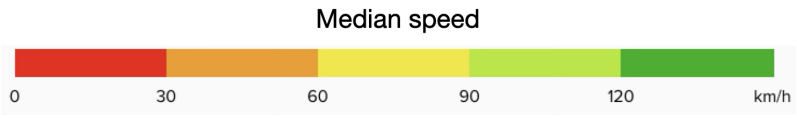
(a) A3 highway near Mainz. Total distance 17 km. Direction of travel North→South.



(b) Comparison of monthly median speeds with TomTom data in Aug 24 between 9:30 and 11:00 am over shown route. We smooth predicted results with Savitzky–Golay filter (window=5, p=1).

Figure 4.11

Now, we again download traffic speed data using TomTom Move account for the above routes (4.9a,4.10a,4.11a)for the same time period i.e August 2024 monthly median data. This time we download the speeds data for different time intervals throughout the day . We notice that bottleneck at all these routes are at the same location irrespective of the time of day. This shows that even though we have satellite data available for only one time snapshot during the day, we can capture congestion patterns irrespectively. We can also estimate the speed profile of the segment, as they are consistently either higher or lower compared with other segments irrespective of the time of day as well.



(a) Legend

(b) Tom Tom monthly median speeds for the route shown in fig 4.9c throughout the day during August 2024

(c) Tom Tom median speeds for route shown in 4.10b throughout the day during August 2024

(d) Tom Tom median speeds for route shown in 4.11 throughout the day during August 24

Figure 4.12: Bottleneck generally occurs at same location irrespective of the time of day

4.2.1 Fundamental diagram of traffic flow

The fundamental traffic flow diagram describes macroscopic relationships between three traffic variables: flow q (veh/h), density k (veh/km), and speed v (km/h),

They are connected by the identity

$$q = k \cdot v. \tag{4.4}$$

A classical model for the speed–density relation was proposed by Greenshields

et al. [1935] and is commonly used as an elementary fit to observed data:

$$v(k) = v_f \left(1 - \frac{k}{k_j}\right), \tag{4.5}$$

where v_f is the free-flow speed (speed at near zero density) and k_j is the jam density (maximum possible vehicle density).

Combining $v(k)$ with $q = kv$ gives the flow–density curve

$$q(k) = v_f\, k \left(1 - \frac{k}{k_j}\right), \tag{4.6}$$

which is a concave parabola attaining a maximum (capacity) at the *critical density*

$$k_c = \frac{k_j}{2}, \qquad q_{\max} = \frac{v_f\, k_j}{4}. \tag{4.7}$$

The fundamental diagram effectively captures essential traffic behavior: at low densities, flow increases with density as vehicles travel close to free-flow speed; beyond the critical density, increasing vehicle density causes speed to drop significantly, reducing overall flow until a jam condition is reached at $k = k_j$, where the flow is zero.

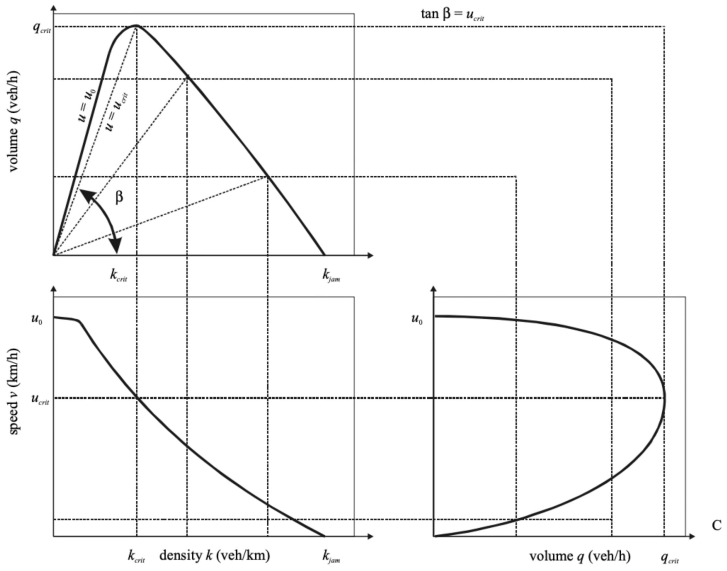
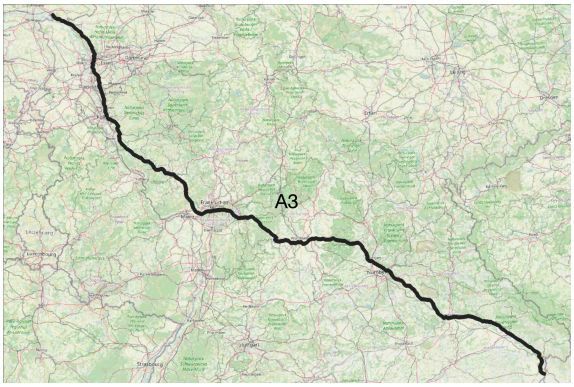


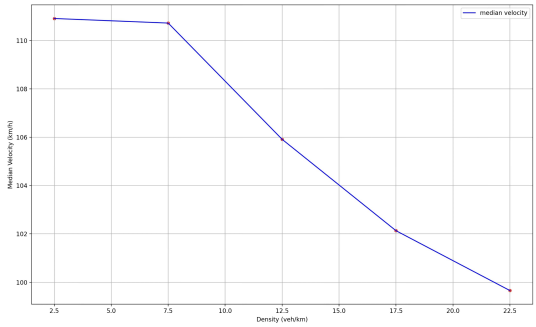
Figure 4.13: Fundamentral traffic diagram. Image source Hoogendoorn [2010]

Application in this study. We compute velocities on the A3 highway for the May 2, 2024 dataset. Median speeds are calculated over 300 m segments, with each median taken as the representative speed for its respective interval. These segments are then binned by density, and plotting the median velocity per bin yields an empirical speed–density curve. Additionally, box plots for each bin display the distribution (Q1, median, Q3, whiskers, and outliers), explicitly illustrating the variability around the statistical trend.

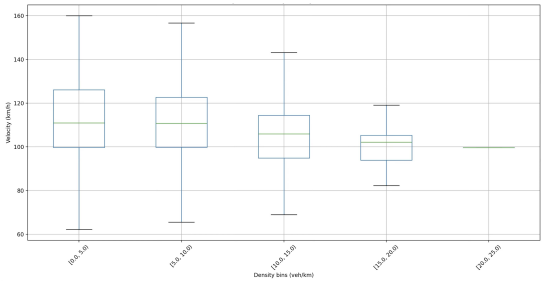
We observe the expected inverse relationship between speed and density. This provides a useful sanity check: when more vehicles are detected, it is likely that more vehicles are indeed present, and, as expected, higher densities lead to lower speeds. Conversely, when fewer vehicles are detected, there are likely fewer vehicles present, resulting in generally higher speeds.



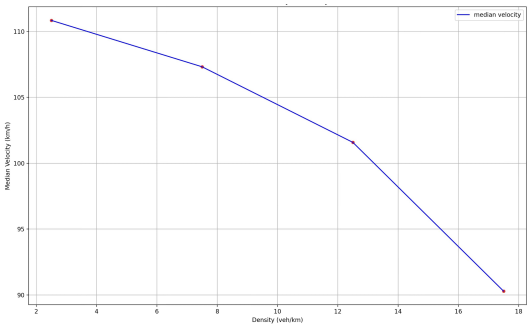
(a) A3 highway



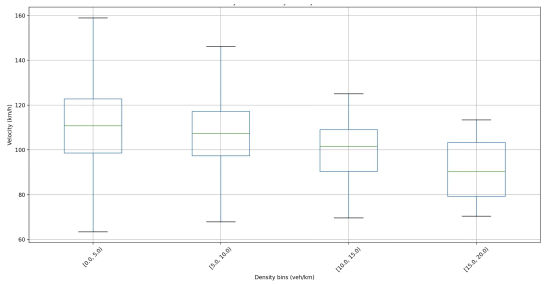
(b) Lane 0: median speed vs. density (Direction: NW → SE)



(c) Lane 0: box plot of speed distribution



(d) Lane 1: median speed vs. density (Direction: SE → NW)



(e) Lane 1: box plot of speed distribution

Figure 4.14: Speed vs density plot using Planetscope data. Analysis on A3 highway, 02 May 2024

5 Conclusion

In this work, we demonstrate that medium resolution satellite imagery can be used to estimate vehicle velocities and detect highway bottlenecks, including the identification of long-tail congestion clusters. By exploiting the temporal offsets between spectral bands in PlanetScope imagery, we transform what is typically considered a sensor artifact into a source of motion information.

We implemented a modified Mask R-CNN for keypoint detection, making changes to the backbone architecture such as removing stride and max-pooling, in the initial stage, downsizing feature maps and deeper layers, using smaller kernels to better preserve small-object details in low-resolution imagery. The loss function was also modified with the inclusion of shape prior loss terms that encourages geometrically plausible keypoint trajectories. This proved especially beneficial in dense traffic scenes, where vehicles are close together and trajectory mismatches are common, leading to visibly better qualitative results. HDBSCAN clustering was applied to group static and slow-moving vehicles, enabling the computation of congestion tail lengths.

Quantitative evaluation showed a mean $AP_{0.50}$ of 0.67 across all vehicle categories. All three labels represent vehicles, which appear as tiny blobs in the imagery but differ in their temporal displacement patterns. Label 1 corresponds to static vehicles whose intensity peaks occur at the same location across channels. Label 2 and Label 3 both correspond to moving vehicles, with the difference being that in Label 2 no displacement is visible in the first channel, whereas in Label 3 the displacement is apparent from the first channel onwards. Since Label 2 and Label 3 both describe motion, we are interested in trajectory length differences rather than class. Their merged $AP_{0.50}$ is 0.93, indicating very high detection accuracy for moving vehicles. The overall RMSE between predicted and ground-truth keypoints was 4.77 m ($\approx$ 1.59 px at 3 m GSD), demonstrating strong localisation accuracy.

For velocity validation, we considered three highway segments of length 56, 80 and 17km. We computed median velocity per 100 meter segment over a month of acquisitions. These were compared to TomTom median velocities for equivalent 50–200 m segments in the same time window (09:30–11:00). Across multiple case studies, our speed profiles aligned closely with TomTom's, with matching bottleneck locations and typical differences within 10–15 km/h elsewhere. Notably, this agreement was achieved despite a much lower sampling density (25–100 vehicle trajectories per km per month) compared to TomTom's $\sim$ 35,000–40,000, underscoring the potential of

our approach for scalable, low-cost traffic monitoring from space.

Future work may include using 8-band PlanetScope imagery, which provides additional visible channels and as a result longer time intervals between far off bands, which may potentially produce much more stable velocity profile. Higher resolution Planet imagery could be used for more accurate ground truth labelling, and if made available at scale and low cost, it could become the primary data source for congestion analysis. Further, fusing PlanetScope data with other traffic and remote sensing sources could enhance calibration and extend the method to regions where reference datasets such as TomTom are unavailable.

Part I

Appendix

5.1 Training plots

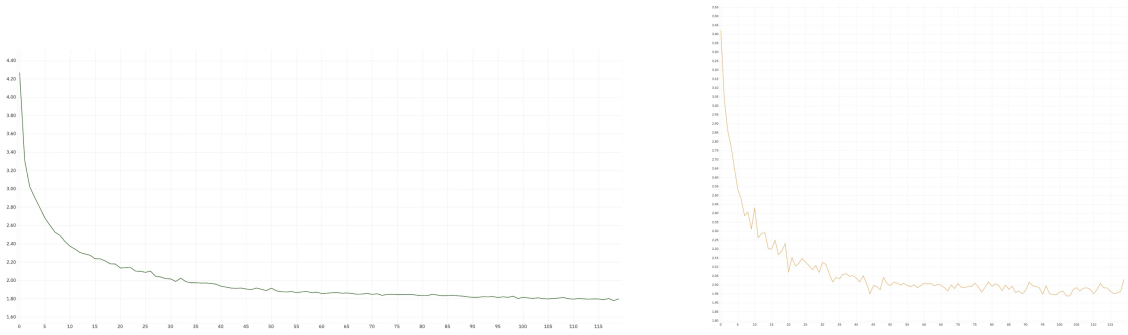


Figure 5.1: epoch loss curve: Train and Validation

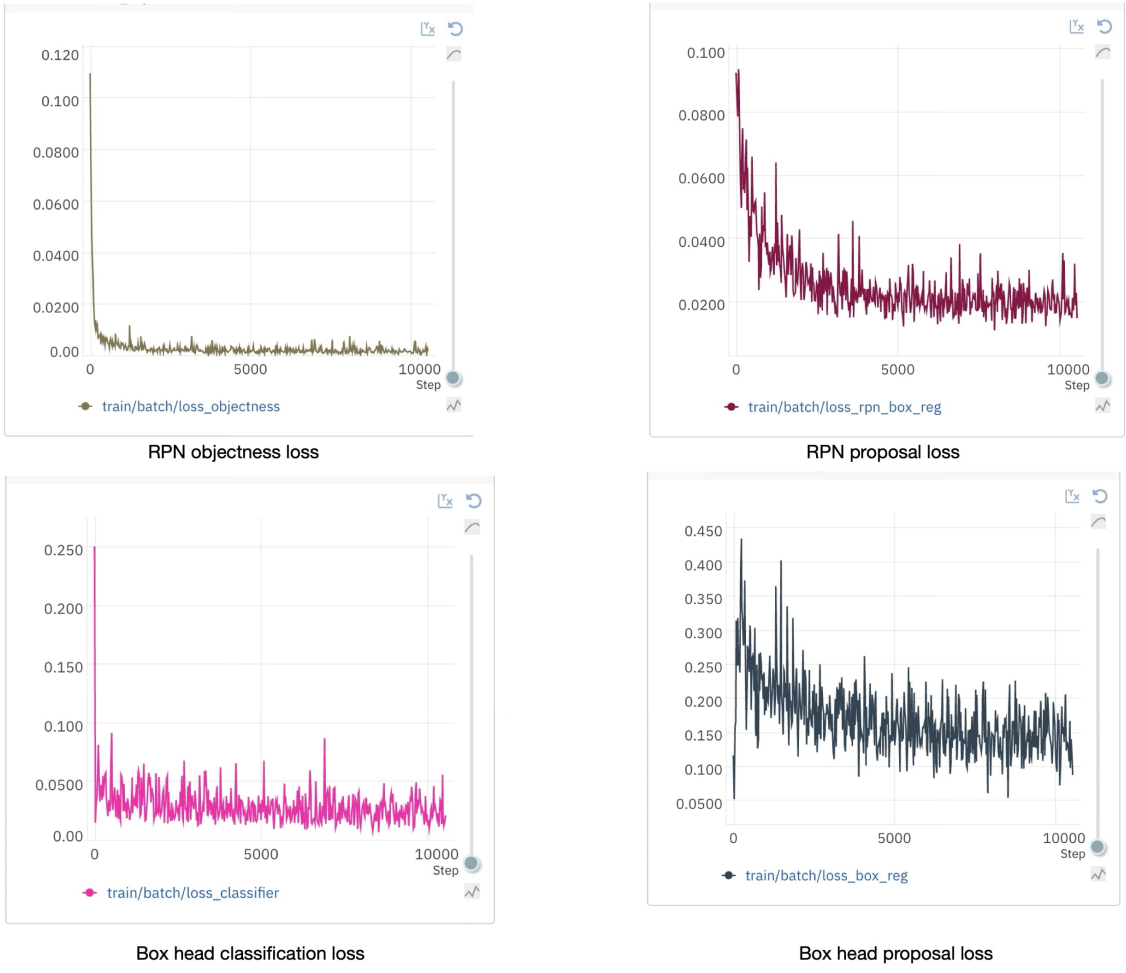
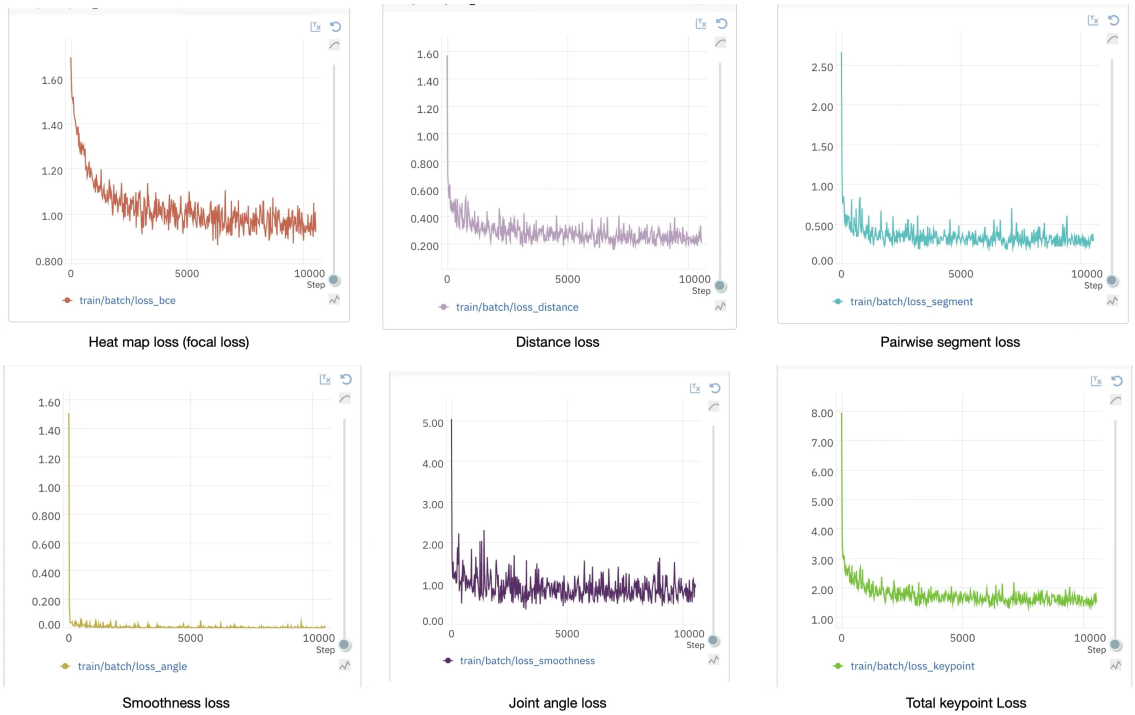


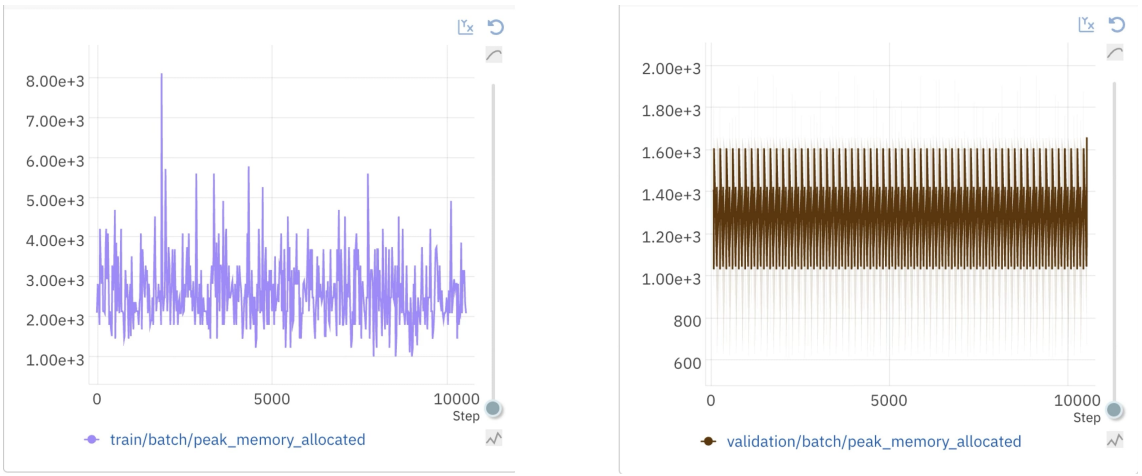
Figure 5.2: Train batch loss curve: Region Proposal head and Box head



Heat map loss (focal loss) | Distance loss | Pairwise segment loss

Smoothness loss | Joint angle loss | Total keypoint Loss

Figure 5.3: Train batch loss curve: Keypoint head



Peak memory allocated per batch during validation

Peak memory allocated per batch during training

Figure 5.4: Peak memory allocation during training and validation

A Lists

A.1 List of Figures

A.2 List of Tables

B Bibliography

Saif Aati, Jean-Philippe Avouac, Ewelina Rupnik, and Marc-Pierrot Deseilligny. Potential and limitation of planetscope images for 2-d and 3-d earth surface monitoring with example of applications to glaciers and earthquakes. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–19, 2022. doi: 10.1109/TGRS.2022.3215821.

Maciej Adamiak, Yulia Grinblat, Julian Psotta, Nir Fulman, Himshikhar Mazumdar, Shiyu Tang, and Alexander Zipf. Deep learning enhanced road traffic analysis: Scalable vehicle detection and velocity estimation using planetscope imagery. *arXiv preprint arXiv:2410.14698*, 2024.

Maciej Adamiak, Yulia Grinblat, Julian Psotta, Nir Fulman, Himshikhar Mazumdar, Shiyu Tang, and Alexander Zipf. Deep learning enhanced road traffic analysis: Scalable vehicle detection and velocity estimation using planetscope imagery. *International Journal of Applied Earth Observation and Geoinformation*, 142:104707, 2025. ISSN 1569-8432. doi: https://doi.org/10.1016/j.jag.2025.104707. URL https://www.sciencedirect.com/science/article/pii/S1569843225003541.

ARSET. *Fundamentals of Remote Sensing*, 2025. URL https://appliedsciences.nasa.gov/get-involved/training/english/arset-fundamentals-remote-sensing. Online self-paced training.

Matthew Barth and Kanok Boriboonsomsin. Real-world carbon dioxide impacts of traffic congestion. *Transportation research record*, 2058(1):163–171, 2008.

Matthew Barth and Kanok Boriboonsomsin. Traffic congestion and greenhouse gases. *Access Magazine*, 1(35):2–9, 2009.

Jean-Yves Bouguet et al. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel corporation*, 5(1-10):4, 2001.

G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013.

Canada Centre for Remote Sensing. *Fundamentals of Remote Sensing*, 2025. URL https://natural-resources.canada.ca/sites/nrcan/files/earthsciences/pdf/resource/tutor/fundam/pdf/fundamentals_e.pdf.

Alan Collison, Arin Jumpasut, Hannah Bourne, and Planet.com. Radiometric calibration white paper. White paper, Planet Labs, 2025. URL https://assets.planet.com/docs/radiometric_calibration_white_paper.pdf. Accessed: 2025-07-21.

Kiruthika Devaraj. *B14: The Cubesat with One of the World's Fastest Satellite Radios*, August 2019. Planet Pulse, 3min read.

Paul HC Eilers and Brian D Marx. Flexible smoothing with b-splines and penalties. *Statistical science*, 11(2):89–121, 1996.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

European Environment Agency. Greenhouse gas emissions from transport in europe. https://www.eea.europa.eu/ims/greenhouse-gas-emissions-from-transport, 2022.

Federal Highway Administration. Traffic monitoring guide. https://www.fhwa.dot.gov/policyinformation/tmguide/2022_TMG_Final_Report.pdf, 2022.

Zhen-Hua Feng, Josef Kittler, Muhammad Awais, Patrik Huber, and Xiao-Jun Wu. Wing loss for robust facial landmark localisation with convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2235–2245, 2018.

GDAL/OGR contributors. *GDAL/OGR Geospatial Data Abstraction software Library*. Open Source Geospatial Foundation, 2025. URL https://gdal.org.

Bruce D Greenshields, J Rowland Bibbins, WS Channing, and Harvey H Miller. A study of traffic capacity. In *Highway research board proceedings*, volume 14, pages 448–477. Washington, DC, 1935.

Matt Grote, Ian Williams, John Preston, and Simon Kemp. Including congestion effects in urban road traffic co2 emissions modelling: Do local government authorities have the right options? *Transportation Research Part D: Transport and Environment*, 43:95–106, 2016.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

S.P. Hoogendoorn. Chapter 4: Fundamental diagrams. https://ocw.tudelft.nl/wp-content/uploads/Chapter-4.-Fundamental-diagrams.pdf, 2010. URL https://ocw.tudelft.nl/wp-content/uploads/Chapter-4.-Fundamental-diagrams.pdf.

GeoSpatial Project Humboldt State University. Radiometric calibration and corrections. https://gsp.humboldt.edu/olm/Courses/GSP_216/online/lesson7/radiometric.html, 2020.

Ian Jolliffe. Principal component analysis. In *International encyclopedia of statistical science*, pages 1094–1096. Springer, 2011.

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017a.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017b. doi: 10.1109/ICCV.2017.324.

David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI'81: 7th international joint conference on Artificial intelligence*, volume 2, pages 674–679, 1981.

Michal Lukasik, Srinadh Bhojanapalli, Aditya Menon, and Sanjiv Kumar. Does label smoothing mitigate label noise? In *International Conference on Machine Learning*, pages 6448–6458. PMLR, 2020.

David Marr and Ellen Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167):187–217, 1980.

Leland McInnes, John Healy, Steve Astels, et al. hdbscan: Hierarchical density based clustering. *J. Open Source Softw.*, 2(11):205, 2017.

Dan Middleton, Robert Parker, and Robert Longmire. Evaluation of non-intrusive technologies for traffic detection. Technical Report Report 2119-1, Texas Transportation Institute, College Station, TX, 2002. URL https://library.ctr.utexas.edu/hostedpdfs/tti/2119-1.pdf.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? *Advances in neural information processing systems*, 32, 2019.

Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016.

Aiden Nibali, Zhen He, Stuart Morgan, and Luke Prendergast. Numerical coordinate regression with convolutional neural networks. *arXiv preprint arXiv:1801.07372*, 2018.

OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org. https://www.openstreetmap.org, 2025.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Planet Labs PBC. Planet documentation, 2024. URL https://docs.planet.com/. Accessed: 2025-06-03.

QGIS Development Team. *QGIS Geographic Information System*, 2025. URL https://qgis.org. Open Source Geospatial Foundation Project.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

Eric Setyawan. *Orthorectification in a Nutshell*, 2019. Intermap Technologies blog post.

Xiao Sun, Bin Xiao, Fangyin Wei, Shuang Liang, and Yichen Wei. Integral human pose regression. In *Proceedings of the European conference on computer vision (ECCV)*, pages 529–545, 2018.

TomTom. Tomtom traffic stats – historical speed data, august 2024. https://move.tomtom.com/traffic-stats/, 2025. Accessed via MOVE trial account, August 2025.

U.S. Geological Survey. Landsat calibration & validation. https://www.usgs.gov/landsat-missions/landsat-calibration-validation, 2025.

Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: Image processing in python. *PeerJ*, 2:e453, 2014. doi: 10.7717/peerj.453.

Luc Vincent. Morphological area openings and closings for grey-scale images. In *Shape in picture: mathematical description of shape in grey-level images*, pages 197–208. Springer, 1994.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3349–3364, 2020.

Xinyao Wang, Liefeng Bo, and Li Fuxin. Adaptive wing loss for robust face alignment via heatmap regression. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6971–6981, 2019.

Runze Yu, Miao Yu, and Jian Guo. A non-intrusive method for traffic flow parameters estimation using video detectors. *Journal of Transportation Technologies*, 3 (1):68–79, 2013. doi: 10.4236/jtts.2013.31007. URL https://file.scirp.org/Html/1-2630015_32663.htm.

Erklärung:

Ich versichere, dass ich diese Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, den August 13, 2025 .....

Declaration:

I hereby confirm that I wrote this work independently and did not use any sources other than those indicated.

Heidelberg, August 13, 2025 .....