

So you think you can track? Supplementary material

The following appendices are included in this file to supplement the main paper.

1. Dataset Coordinate Systems
2. Homography Re-estimation Method
3. Homography Error Metrics
4. Additional GPS Trajectory Plots
5. Object Detector
6. Experimental Details
7. Treatment of Personally Identifiable Information
8. Known Data Artifacts and Anomalies

1. Dataset Coordinate Systems

The Interstate-24 Video dataset utilizes 3 sets of coordinates utilized throughout I-24 MOTION system [7] datasets:

- **Image Coordinates:** are given in pixels. $(y^{(im)}, x^{(im)})$ denotes the row and column of the specified pixel. By convention the top left pixel is (0,0).
- **State Plane Coordinates:** specify a rectilinear and orthogonal coordinate system. The EPSG 2274 state plane coordinate system for Tennessee is specified in feet relative to a known survey point. $(x^{(st)}, y^{(st)})$ indicates the coordinate (in feet) along the first (roughly east-west) and second (roughly north-south) coordinate axis defined by the state plane coordinate system. (Note that a common conversion from state plane coordinates to latitude/longitude coordinates (e.g. WSG84 or NAD83) can be utilized if desired.) A third orthogonal coordinate axis (z-axis) is defined and corresponds to distance off the roadway, such that $z^{(st)} = 0$ for all points on the roadway plane.
- **Roadway Coordinates:** are defined such that the primary (x) axis lies along the median (or more precisely, midway between the two interior yellow lines for the interstate) at all points within the instrument extents, and the secondary (y) axis is defined locally to perpendicular to the primary axis at all points along the roadway. All coordinates with a distance from the primary axis less than the local radius of curvature (including all points on the roadway) have a unique $(x^{(r)}, y^{(r)})$ coordinate. By left-hand rule convention, we define the positive y-axis to be in the direction of the eastbound roadway lanes at all points along the roadway.

1.1. Notation

Throughout the rest of this appendix to disambiguate the various coordinate systems, the following notation is used:

- $x, y,$ and z refer to coordinate axes. A superscript $(im), (st),$ or (r) specifies all variables corresponding to a specific coordinate system (e.g. $x^{(st)}$).
- Vectors and matrices in a specified coordinate system are denoted in bold (e.g. $\mathbf{O}^{(st)}$).
- Homography matrices are also listed in bold script, without superscript (e.g. \mathbf{H}).
- A subscript indexes a specific point (e.g. $x_{bbi}^{(st)}$), and subscript i indicates an arbitrary element index from a set of elements (e.g. a_i).
- An $x, y,$ or z without a subscript indicates a generic variable along the specified axis within the specified coordinate system.

A list of all variables along with their descriptions is given in Table 1. Transformations between image and state plane coordinates, and transformations between state plane and roadway coordinates are detailed in the next two sections.

1.2. Image \leftrightarrow State Plane Conversion

(parts of this subsection rely on similar definitions and descriptions to the supplement in [6].) A *homography* relates two views of a planar surface. For each camera, we provide homography information such that the 8-corner coordinates of the stored 3D bounding-box annotation can be projected into any camera view for which the vehicle is visible, creating a monocular 3D bounding box within that camera field of view. For each direction of travel in each camera view, for each scene, a homography relating the image pixel coordinates to the state plane coordinate system is defined. (Though the same cameras are used for different scenes, the positions of the cameras changes slightly over time due). A local flat plane assumption is used (the state plane coordinate system is assumed to be piece-wise flat) [8]. A series of correspondence points $a_i = [x^{(im)}, y^{(im)}, x^{(st)}, y^{(st)}, z^{(st)}]$ are used to define this relation, where $(y^{(im)}, x^{(im)})$ is the coordinate of selected correspondence point a in pixel coordinates (row, column) and $(x^{(st)}, y^{(st)}, z^{(st)})$ is the selected correspondence point in state plane coordinates.

All selected points are assumed to lie on the state plane, so $z^{(st)} = 0$ for all selected correspondence points. Visible lane marking lines and other easily recognizable landmarks on the roadway are used as correspondence points in each camera field of view. Each correspondence point is also labeled in *global information system* (GIS) software, giving the precise GPS / state-plane coordinate system coordinates for each labeled corresponding point. The corresponding pixel coordinates are manually selected in each camera field of view, for each direction of travel on the roadway (see Appendix 1.5).

Symbol	Definition
\mathbf{H}	3×3 matrix of homography parameters h_{ij}
\mathbf{P}	3×4 matrix of homography parameters p_{ij}
s	homography scale parameter
$x^{(im)}, y^{(im)}$	image coordinates (y indicates pixel row and x indicates pixel column)
$x^{(st)}, y^{(st)}, z^{(st)}$	state plane coordinates
$x^{(r)}, y^{(r)}$	roadway coordinates
$\mathbf{O}^{(st)}$	state plane coordinates for object, equal to $[\mathbf{o}_{bbl}^{(st)}, \mathbf{o}_{bbr}^{(st)}, \mathbf{o}_{btl}^{(st)}, \mathbf{o}_{btr}^{(st)}, \mathbf{o}_{fbl}^{(st)}, \mathbf{o}_{fbr}^{(st)}, \mathbf{o}_{ftl}^{(st)}, \mathbf{o}_{ftr}^{(st)}]$
$\mathbf{o}_{bbl}^{(st)}$	back bottom left state plane coordinate of object, equal to $[x_{bbl}^{(st)}, y_{bbl}^{(st)}, z_{bbl}^{(st)}]$
$\mathbf{o}_c^{(st)}$	back bottom center state plane coordinate, primary reference coordinate for the object
$\mathbf{o}_{spl}^{(st)}$	state plane coordinates of point on center-line spline ($y^{(r)} = 0$) with the same $x^{(r)}$ coordinate as $\mathbf{o}_c^{(st)}$
$\mathbf{O}^{(r)}$	roadway coordinates for object, $[x^{(r)o}, y^{(r)o}, l, w, h]$
$x^{(r)}$	generic longitudinal roadway coordinate along curvilinear spline axis
$y^{(r)}$	generic lateral roadway coordinate along axis locally perpendicular to longitudinal roadway coordinate axis
$x_o^{(r)}$	object longitudinal roadway coordinate along curvilinear spline axis
$y_o^{(r)}$	object lateral roadway coordinate along axis locally perpendicular to longitudinal roadway coordinate axis
l, w, h	rectangular prism dimensions (length, width and height)
$F(x^{(r)})$	spline defining state plane coordinate roadway center-line spline parameterized by $x^{(r)}$
$\tilde{G}(x^{(st)})$	spline approximating the center-line spline in roadway coordinates $x^{(r)}$ parameterized by $x^{(st)}$

Table 1. Summary of symbols used in this section.

A *perspective transform* (Equation 2) is fit to these correspondence points. We first define a 2D perspective transform which defines a linear mapping (Equation 1) of points from one plane to another that preserves straight lines. The correspondence points are then used to solve for the best perspective transform \mathbf{H} as defined in equation 2, where s is a scale factor.

$$s \begin{bmatrix} x_i^{(st)} \\ y_i^{(st)} \\ 1 \end{bmatrix} \sim \mathbf{H} \begin{bmatrix} x_i^{(im)} \\ y_i^{(im)} \\ 1 \end{bmatrix} \quad (1)$$

where \mathbf{H} is a 3×3 matrix of parameters:

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \quad (2)$$

For each camera field of view and each direction of travel, the best perspective transform \mathbf{H}^* is determined by minimizing the sum of squared re-projection errors according to equation 3 as implemented in OpenCV's *find_homography* function [4]:

$$\mathbf{H}^* = \arg \min_{\mathbf{H}} \sum_i \left(x_i^{(st)} - \frac{h_{11}x_i^{(im)} + h_{12}y_i^{(im)} + h_{13}}{h_{31}x_i^{(im)} + h_{32}y_i^{(im)} + h_{33}} \right)^2 + \left(y_i^{(st)} - \frac{h_{21}x_i^{(im)} + h_{22}y_i^{(im)} + h_{23}}{h_{31}x_i^{(im)} + h_{32}y_i^{(im)} + h_{33}} \right)^2 \quad (3)$$

The resulting matrix \mathbf{H}^* allows any point lying on the plane within the camera field of view to be converted into state plane coordinates. The corresponding matrix \mathbf{H}_{inv} can easily be obtained to project state plane coordinates on the $z = 0$ plane into image coordinates. However, since each vehicle is represented by a 3D bounding box, the top corner coordinates of the box do not lie on the ground plane. A 3D perspective transform \mathbf{P} is needed to linearly map coordinates from 3D state plane coordinate space to 2D image coordinate space, where \mathbf{P} is a 3×4 matrix of parameters:

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \quad (4)$$

and \mathbf{P} projects a point in 3D space (x', y', z') into the corresponding image point (x, y) according to:

$$\mathbf{P} \begin{bmatrix} x^{(st)} \\ y^{(st)} \\ z^{(st)} \\ 1 \end{bmatrix} \sim s' \begin{bmatrix} x^{(im)} \\ y^{(im)} \\ 1 \end{bmatrix} \quad (5)$$

where s' is a new scaling parameter. By observing the case where $z^{(st)} = 0$, it is evident columns 1,2, and 4 of \mathbf{P} are equivalent to the columns of \mathbf{H}_{inv} and can be fit in the same way. Thus, we need only solve for column 3 of \mathbf{P} . Next, we note as in [8] that $(\frac{p_{11}}{p_{31}}, \frac{p_{21}}{p_{31}})$ is the vanishing point (in image coordinates) of perspective lines drawn in the same direction as the state plane coordinate x-axis. The same is true for the 2nd column and the state plane coordinate y-axis, the 3rd column and the state plane coordinate z-axis, and the 4th column and the state plane coordinate origin.

Thus, to fully determine \mathbf{P} it is sufficient to locate the vanishing point of the z-axis in state plane coordinates and to estimate the scaling parameter p_{33} . The vanishing point is located in image coordinates by finding the intersection point between lines drawn in the z-direction. Such lines are obtained by manually annotating vertical lines in each camera field of view. The scale parameter is estimated by minimizing the sum of squared reprojection errors defined in equation 6 for a sufficiently large set of state plane coordinates and corresponding, manually annotated coordinates in image space.

$$\mathbf{P}^* = \arg \min_{p_{33}} \sum_i \left(x_i^{(im)} - \frac{p_{11}x_i^{(st)} + p_{12}y_i^{(st)} + p_{13}z_i^{(st)} + p_{14}}{p_{31}x_i^{(st)} + p_{32}y_i^{(st)} + p_{33}z_i^{(st)} + p_{34}} \right)^2 + \left(y_i^{(im)} - \frac{p_{21}x_i^{(st)} + p_{22}y_i^{(st)} + p_{23}z_i^{(st)} + h_{24}}{p_{31}x_i^{(st)} + p_{32}y_i^{(st)} + p_{33}z_i^{(st)} + h_{34}} \right)^2 \quad (6)$$

The resulting 3D perspective transform \mathbf{P}^* allows for the lossless conversion of points in roadway coordinates to the corresponding points in image coordinates. Observing that a lossless conversion from image coordinates to state plane coordinates is available provided that the converted point lies on the $z^{(st)} = 0$ plane, it is possible to precisely convert a rectangular prism from image space to state plane coordinates by i.) converting the footprint of the prism near-losslessly into state plane coordinates (the only source of error comes from a set of 4 image coordinates that cannot be perfectly converted into a rectangle in state plane coordinates), ii.) shifting the footprint in state plane coordinates along the z-axis, iii.) re-projecting the resulting points back into the image, iv.) comparing the reprojected “top points” to the original top of the rectangular prism in image coordinates, and v.) adjusting the height iteratively to minimize the re-projection error until convergence.

1.3. State Plane \rightarrow Roadway Coordinate System Conversion

Next, we consider the conversion of points in state plane coordinates to roadway coordinates. In most cases, we care to convert a set of state plane coordinate points roughly in a rectangular prism (i.e. vehicle 3D bounding box) into roadway coordinates; thus, we define this conversion for a rectangular prism. A single point can be converted between state plane coordinates and roadway coordinates by treating it as a rectangular prism with zero length, width and height.

Let $\mathbf{O}^{(st)}$ be a 3D bounding box representation in state plane coordinates, an 8×3 matrix of x,y, and z coordinates for each corner of the box. (Note that these corners need not exactly correspond to an orthogonal rectangular prism, but the roadway coordinate equivalent will be exactly orthogonal so some truncation will occur.) We reference, for example, the back bottom right (from the perspective of the rear of the vehicle) of object $\mathbf{O}^{(st)}$ as $\mathbf{o}_{bbrr}^{(st)} = [x_{bbrr}^{(st)}, y_{bbrr}^{(st)}, z_{bbrr}^{(st)}]$, such that $= \mathbf{O}^{(st)} = [\mathbf{o}_{bbll}^{(st)}, \mathbf{o}_{bbrr}^{(st)}, \mathbf{o}_{btrl}^{(st)}, \mathbf{o}_{btrr}^{(st)}, \mathbf{o}_{fbl}^{(st)}, \mathbf{o}_{fbr}^{(st)}, \mathbf{o}_{ftl}^{(st)}, \mathbf{o}_{ftr}^{(st)}]$. (For the single-point case described above, all 8 corner coordinates are identical).

Next, Let $\mathbf{O}^{(r)} = [x_o^{(r)}, y_o^{(r)}, l, w, h]$ be the corresponding object representation of $\mathbf{O}^{(st)}$ in roadway coordinates. $x^{(r)}$ and $y^{(r)}$ are the roadway coordinate longitudinal and lateral coordinates (in feet), and $l, w,$ and h are the length, width, and height of the object respectively (in feet).

Let $\mathbf{o}^{(st)c}$ denote the back bottom center coordinate of object $\mathbf{O}^{(st)}$. By convention, this point is referenced as the primary position of object $\mathbf{O}^{(st)}$. Let $\mathbf{o}^{(st)spl}$ denote the point on the center-line spline (i.e. $y^{(r)} = 0$) with the same $x^{(r)}$ coordinate as $\mathbf{o}_c^{(st)}$.

Let F be the second-order spline parameterizing the roadway center-line in state plane coordinates. In other words, F defines the longitudinal curvilinear axis $y^{(r)} = 0$ along this spline. F is fit by manually labeling a sufficiently large number of

points along the interior yellow line for both directions of travel (in state plane coordinates). A spline is fit to each yellow line, and a third spline is fit to lie precisely halfway between these two splines. Spline control points are selected at suitably sparse intervals (200 foot minimum spacing) such that the spline is relatively smooth while still capturing the roadway curvature (see Appendix 1.6).

Given $\mathbf{O}^{(st)}$, we first obtain l, w and h by computing the average distance between points on the front and back, left and right, or top and bottom of the vehicle respectively. Next, we obtain $\mathbf{o}_c^{(st)}$ by computing the average $x^{(st)}$ and $y^{(st)}$ state plane coordinates of the 4 back rectangular prism corners.

Next, we solve for $x_o^{(r)}$ by solving the following optimization:

$$x_o^{(r)} = \arg \min_{x^{(r)}} \|(F(x^{(r)}), \mathbf{o}_c^{(st)})\|_2 \quad (7)$$

using L2-norm (Euclidean distance) between the two points in state plane coordinate space. In other words, determine the point on the roadway spline closest to the back center of the rectangular prism $\mathbf{o}_c^{(st)}$. This minimizing point is the corresponding roadway longitudinal coordinate $x_o^{(r)}$, and the distance from the minimum distance point is roadway lateral coordinate $y_o^{(r)}$.

$$y_o^{(r)} = \min_{x^{(r)}} \|(F(x^{(r)}), \mathbf{o}_c^{(st)})\|_2 \quad (8)$$

Noting that the I-24 MOTION roadway segment has monotonically increasing $x^{(st)}$ coordinate, a secondary spline $\tilde{G}(x^{(st)})$ is defined to parameterize $x^{(r)}$ as a function of $x^{(st)}$, which yields a good initial guess for the closest roadway longitudinal coordinate for a given point in state plane coordinates. This optimization can then be solved to arbitrary precision, yielding the complete roadway coordinate for the object $\mathbf{O}^{(r)} = [x_o^{(r)}, y_o^{(r)}, l, w, h]$.

Constant Yellow Line Constraint: It is observed that points along the yellow line for each roadway direction of travel have non-constant y-position due to the varying width of the median. It is desirable that the yellow line (and by extension each set of lane-dividing markings) has constant y position. We finally apply the following shift:

$$y_o^{(r)} += (C - \gamma(x_o^{(r)})) \quad (9)$$

where C is the desired constant yellow line y-coordinate (in this case -12 for westbound-side coordinates and +12 for eastbound-side coordinates) and $\gamma(x_o^{(r)})$ represents the uncorrected yellow-line coordinate at the given x-position per roadway side. Note that this creates a discontinuity in the coordinate system near $y = 0$ on each side, but these portions of the coordinate system are not used for objects on the roadway.

1.4. Roadway \rightarrow State Plane Conversion

Reverse Yellow Line Constraint: First, the inverse yellow-line shift must be applied to the y-coordinate:

$$y_o^{(r)} -= (C - \gamma(x_o^{(r)})) \quad (10)$$

Next, given roadway coordinates for an object $\mathbf{O}^{(r)}$, first find the corresponding point on the roadway center-line spline in state plane coordinates $\mathbf{o}_c^{(st)}$ according to:

$$F(x_o^{(r)}) = \mathbf{o}_{spl}^{(st)} \quad (11)$$

To obtain the back center coordinate $\mathbf{o}_c^{(st)}$, we must offset $\mathbf{o}_{spl}^{(st)}$ by length $y^{(r)}$ in the direction perpendicular to the roadway centerline spline at $\mathbf{o}_c^{(st)}$. Let $\vec{\mathbf{u}}_F$ be the unit vector in the same direction as the derivative spline F' , and let $\vec{\mathbf{u}}_{1/F}$ be the unit vector in the perpendicular direction (along the state plane, i.e. $z^{(st)} = 0$. Note that care should be given to ensure that the positive direction of $\vec{\mathbf{u}}_{1/F}$ points towards the eastbound side of the roadway with positive $y^{(r)}$.) Then, $\mathbf{o}_c^{(st)}$ is given by:

$$\mathbf{o}_c^{(st)} = \mathbf{o}_{spl}^{(st)} + y^{(r)} \cdot \vec{\mathbf{u}}_{1/F} \quad (12)$$

From here, the corner state plane coordinates for the right and left coordinates of the rectangular prism can be obtained by offsetting $\mathbf{o}_c^{(st)}$ by $\pm \frac{1}{2}$ times w in the direction of $\vec{\mathbf{u}}_{1/F}$, and the front coordinates of the rectangular prism can similarly



Figure 1. Correspondence points (blue) and fields of view (shaded polygons) for each roadway direction of travel, labeled in a single camera field of view (P27C01). Around 100 correspondence points are visible for each side of the roadway.

be obtained by offsetting by l in the direction of \vec{u}_F or in the opposite direction for objects on the westbound or negative $y^{(r)}$ side of the roadway. Similarly, the top coordinates can be obtained by offsetting by a factor of h in the $z^{(st)}$ direction. The direction of travel for an object can be obtained as the sign of the $y^{(r)}$ coordinate (negative for WB, positive for EB). For example, for an eastbound object the front top left coordinate can be obtained as:

$$\mathbf{o}_{ftl}^{(st)} = \mathbf{o}_c^{(st)} - \frac{1}{2} \cdot w \cdot \vec{u}_{1/F} + l \cdot \vec{u}_F + h \cdot [0, 0, 1] \quad (13)$$

1.5. Correspondence Point Labeling

A brief description and a few pictures of overhead and camera fields of view.

Correspondence points are obtained for visible lane marking lines and other easily recognizable landmarks on the roadway are used as correspondence points in each camera field of view. Labeling is carried out in a custom OpenCV GUI, with semi-automated point selection to speed this process over the 230+ cameras utilized in this work. An average frame from each camera is used to minimize the presence of occluding vehicles during labeling. Figure 1 shows an example of labeled points and the labeled relevant field of view for a single camera. The corresponding pixel coordinates for each manually selected point in each camera field of view, are stored, for each direction of travel on the roadway, each with a unique identifier. Typically, at least 10 roadway markings or 40 correspondence points are labeled per roadway direction of travel. Additionally, note that this initial correspondence point selection was performed for a day a few weeks prior to the day on which video is recorded in this work.

Each correspondence point is also labeled in *global information system* (GIS) software, giving the precise GPS / state-plane coordinate system coordinates for each labeled corresponding point. The same unique identifier is assigned to each corresponding point previously labeled within a camera image. Additionally, points along each yellow line were labeled periodically (every ~ 50 feet). Figure 3 shows an example of labeled points in the state plane aerial imagery.

1.6. Centerline Spline Fitting

The following procedure was used to fit the center-line spline for the roadway coordinate system ($y^{(r)} = 0$):

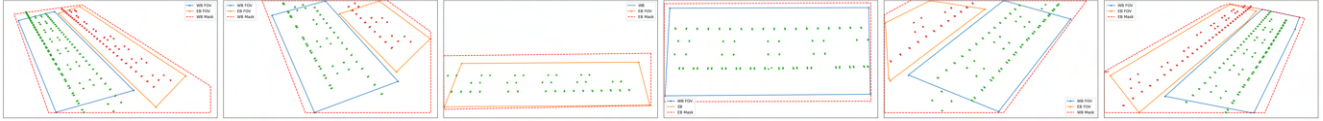


Figure 2. Correspondence points (green for WB red for EB) labeled in camera imagery for all cameras on one pole (P17). Fields of view (blue for WB and orange for EB) and a mask denoting relevant portions of the image for tracking (dotted red polygon) are also shown. Similar plots are included in a separate file for all cameras.

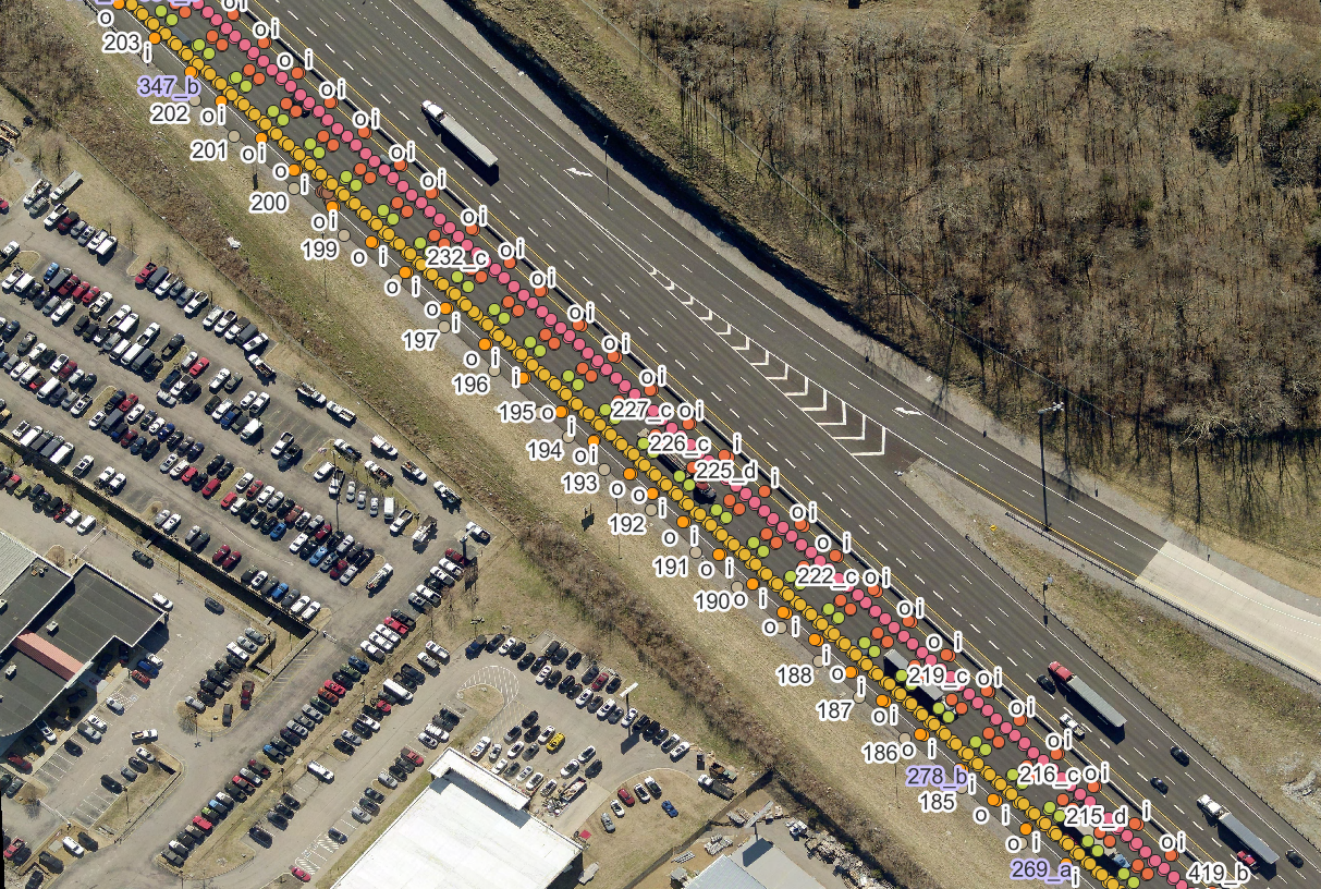


Figure 3. Correspondence points labeled in aerial imagery using the GIS tool for a very small subset (~500 feet) of roadway, for one direction of travel only.

1. For each roadway direction of travel, fit a spline parameterizing the $(x^{(st)}, y^{(st)})$ points as a function of u . Let $F_{(WB)}(u)$ and $F_{(EB)}(u)$ denote these splines. All splines implemented in this work are based on SciPy's spline package [16] and are constrained to have a minimum control point spacing of 200 feet to ensure smoothness.
2. Moving along the EB yellow line spline $F_{(EB)}(u)$, sample points at fine (1 foot) intervals. For each sampled point u , find the closest point on $F_{(WB)}(u)$ u' . Store the midpoint u^* .
3. Fit another spline $F_{med}(u)$ to the set of midpoints from the previous step.
4. Sample $F_{med}(u)$ at fine (0.1 foot intervals). Compute the distance between each consecutive pair, and compute the cumulative distance along the spline to each point via finite difference approximation.
5. Re-parameterize $F_{med}(u)$ such that each point on the midpoint corresponds to the integrated distance along the spline to that point. This yields the final spline $F(x^{(r)})$ used above.

Lastly, $\gamma(x^{(r)})$ must be computed for each roadway direction of travel. This is done by sampling $F(x^{(r)})$ for each direction at regular (5-foot) intervals and recording the distance to the closest point on each of $F_{(WB)}$ and $F_{(EB)}$. Since these offsets change very little (~ 1 foot per mile), the offsets are stored in a lookup table rather than fitting a true offset function. Thus, $\gamma(x^{(r)})$ returns the recorded offset from $F(x^{(r)})$ to $F_{(WB)}$ or $F_{(EB)}$ for the closest sampled value $x^{(r)}$.

Figure 4a shows the labeled points from aerial imagery (yellow and black lines) and the points labeled in each image (blue dots), transformed into roadway coordinates. Figure 4b shows the largest magnitude shift in labeled correspondence points, when converted to the roadway coordinate system. Such “discontinuities” are due to slight misalignments between aerial images taken during different passes of the photographing aircraft, and result in small (less than 2 foot) errors in all cases. While these misalignments could be corrected for by applying a smoothing to the correspondence points labeled within the aerial imagery, this was ultimately decided against because the smoothing would need to be performed with respect to, essentially, the roadway coordinate system spline itself, which is in turn a product of the labeled aerial imagery points. This would produce complex and less well-understood artifacts in the resulting final coordinate system.

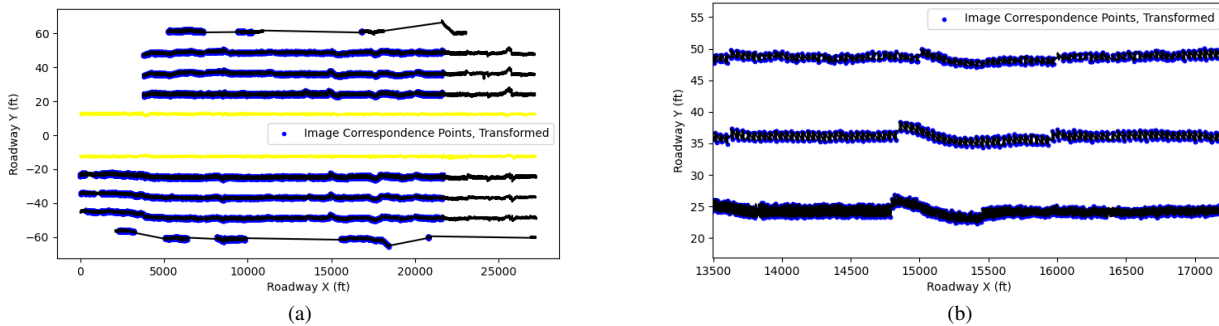


Figure 4. (a) correspondence points labeled in aerial imagery (yellow and black) and in individual camera fields of view (blue) are projected into the roadway coordinate system. (b) A close-up detailing a type of artifact visible in the coordinate system as a result of misalignment in the aerial imagery.

2. Homography Re-estimation Methods

In ideal circumstances homographies can be estimated once (as discussed in the previous section), and used continuously until some drastic change in the system, (e.g. the roadway section is rebuilt or the lane markings are repainted). Aside from these rare events there are several other factors which significantly degrade accuracy; long term events can be the replacement of the camera, inaccuracies in the pan-tilt mechanism during homing, settlement of the foundation or the seasonal temperature change. Although these can be dealt with occasional re-calibration, the *sunflower effect* requires constant re-estimation of the homographies. This effect, the tilting of the metal infrastructure poles due to the differential heating of the sun and shade-facing sides, can cause significant homography errors (sometimes greater than 10 feet, see Figure 5 for a typical camera example), both in timescales of hours for the daily warm up and cool down cycle, and also in the timescales of minutes, caused by the varying cloud coverage (see Figure 6, showing how the positional drift in homographies varies over just 12.5 minutes). A video is included in supplementary material showing the magnitude of drift over the course of a day. The camera movement can easily be seen by viewing the relation between the lane markers and the ROI rectangle (which maintains constant pixel coordinates throughout the video). For reference, a typical dash line is 10 feet long.

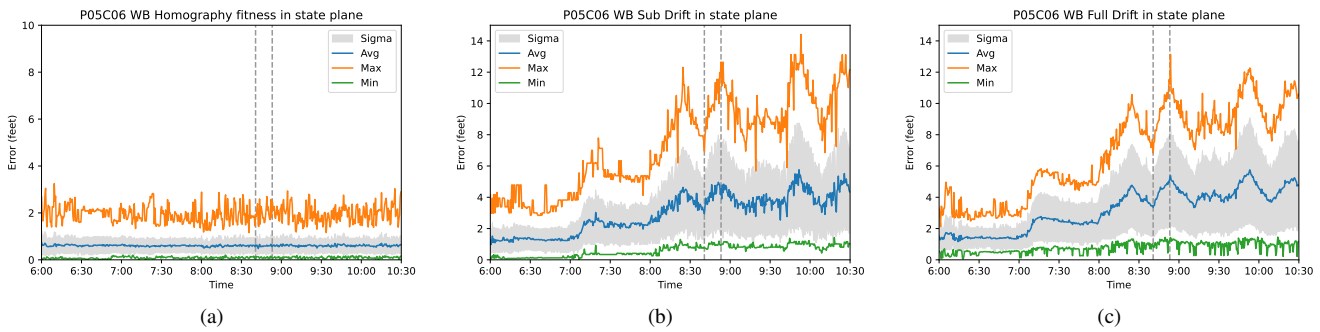


Figure 5. Homography goodness of fit and uncorrected drift between reference homography and true scene homography according to two metrics (Sub Drift and Full Drift) for a single typical camera. Horizontal grey dashes show the two time instances for Figure 6.

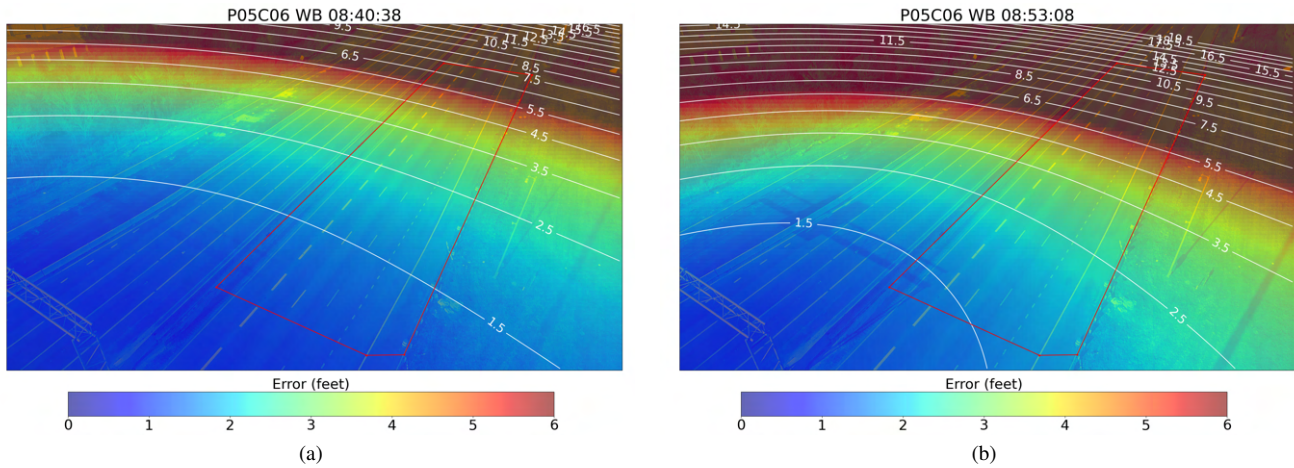


Figure 6. Full drift for the same camera at two time instances. For a given error value X , this indicates that the original reference homography and the true homography for the scene map the same image coordinate to points in the state plane X feet apart. The westbound field of view (i.e. the most important portion of the image) is shown as a red polygon.

Note, that tilting not only degrades single-camera accuracy, but due to the multiple pole architecture and the camera FOVs, homography errors cause significant misalignment in state plane coordinates between poles, resulting in vehicle tracking fragmentations. (Cameras on the same pole are less susceptible because they are rigidly mounted and roughly "moving" together.) Thus, the re-estimation of the homography is necessary on relatively short timescales, or at a minimum on a daily basis.

Since manual re-annotation is not possible for continuous operation, a feature point re-identification method is developed, along with re-estimation methods for a static daily and for a dynamic, time-dependent homography.

All methods and filtering steps are based on the assumption that the image is only slightly distorted from the manually labeled ground truth. This assumption is necessary, because the homography and the re-discovery relies on the lane markings which are non-unique and repetitive, thus a significant shift can cause a mis-identification and misalignment. All presented methods are *offline* (they utilize both past and future information relative to the specified time instance (i.e. use a non-casual filter)).

Automated homography generation is divided into two distinct steps. First the feature points are re-discovered and a new homography is calculated for a single time instance. Second, a refinement step operates on the homographies themselves of which removes outliers, filters, and aggregates over time. This approach provides robustness for the final estimates based on the assumption that the re-estimation provides good results *in general* but are prone to outliers and errors. To comply to this idea the implemented refinement processes are draconian, and tuned to minimize false positives at the expense of occasionally removing some true positives. The "good in general" assumption is empirically tested, by rigorously inspecting the results and fine tuning the filter parameters of each steps. In case of faulty estimates manual inspection always shows problems which could cause confusion even for human observers or significant noise and drop in image quality (e.g.: at dawn).

2.1. Notation

The following notation convention is used throughout the remainder of this Section. Also see Table 2 for a list of utilized symbols and their meanings.

- Mathcal script (e.g. \mathcal{I}) is used for sets of points. Bold Text (e.g. \mathbf{H}) is used for vectors and matrices.
- The subscript t denotes a set of points or a homography for a specific time instance (e.g. \mathcal{I}_t)
- No subscript denotes the initial reference points or homography (e.g. \mathcal{I} , \mathcal{A} , and \mathbf{H})
- The prime symbol denotes rediscovered points (e.g. \mathcal{I}'_t , which corresponds to subset \mathcal{A}_t), or homographies based on rediscovered points (e.g.: \mathbf{H}'_t)
- a superscript s denotes an image-to-image homography based on SIFT-FLANN matching (e.g. \mathbf{H}_t^s)
- A superscript asterisk symbol $*$ denotes a homography *estimate* calculated with methods presented in the previous appendix (e.g.: \mathbf{H}^*).
- An arrow $\mathcal{I} \xrightarrow{\mathbf{H}}$ indicates a linear transformation on the point set \mathcal{I} according to homography \mathbf{H} . (e.g. $\mathcal{I}'_t \xrightarrow{\mathbf{H}'_t} \hat{\mathcal{A}}_t$ denotes that the set \mathcal{I}'_t of rediscovered points at time t , projected from image coordinates to state plane coordinates by \mathbf{H}'_t , estimates the set of corresponding points labeled in state plane coordinates \mathcal{A}_t).

2.2. Homography Generation for a Single Time Instance

This section briefly describes the steps necessary for the homography generation for a single time instance t .

1. Generate a background extracted image to remove the vehicles from the scene. This is implemented as a 1 minute long averaging of frames, with 50% overlap in time. Note that in case of heavy traffic (essentially for stopped vehicles) this time period might not be sufficient for complete removal.
2. Compute an initial alignment estimate between the frame and the reference frame, on which the original annotations were made. This step utilize the SIFT algorithm to find feature points and a FLANN based matcher (both implemented in OpenCV). Note that the detected features can be anywhere on the image, not necessarily corresponding to semantically meaningful points (e.g. grass, trees, parking cars outside the road), and most importantly generally not lie on the plane of the roadway.
3. Based on the SIFT-FLANN generated corresponding points generate an image-to-image homography (\mathbf{H}_t^s).
4. Examine \mathbf{H}_t^s : transformation (e.g.: translation, rotation, scale) should be minimal, based on the slight distortion assumption. Otherwise discard the current time instant, because the calculated alignment is likely from erroneously matched points. No further steps are possible.
5. Binarize the averaged frame via OpenCV's *threshold* function [4] and run a contour detection algorithm. This provides good results because lane markings are high contrast, distinct features.

Symbol	Definition
\mathcal{I}	labeled image points (in image coordinates, unit: pixels)
\mathcal{I}_t	subset of the labeled image points at time t , corresponding to the successfully rediscovered points ($\mathcal{I}_t \subset \mathcal{I}$)
\mathcal{I}'_t	rediscovered image points ($ \mathcal{I}'_t \leq \mathcal{I} $)
\mathcal{I}_t^s	labeled image points after perspective transformation, utilizing the SIFT-FLANN matcher, such that $\mathcal{I} \xrightarrow{\mathbf{H}_t^s} \mathcal{I}_t^s$
\mathcal{I}_t^*	”labeled” image points generated such as $A \xrightarrow{\mathbf{H}_t^{*\top}} \mathcal{I}_t^*$
\mathcal{A}	aerial points (in state plane coordinates, unit: feet)
\mathcal{A}_t	subset of the aerial points at time t , corresponding to the successfully rediscovered points ($\mathcal{A}_t \subset \mathcal{A}$)
\mathbf{H}	homography such that $\mathcal{I} \xrightarrow{\mathbf{H}} \mathcal{A}$
\mathbf{H}^\top	inverse homography such that $\mathcal{A} \xrightarrow{\mathbf{H}^\top} \mathcal{I}$
\mathbf{H}'_t	homography such that $\mathcal{I}'_t \xrightarrow{\mathbf{H}'_t} \mathcal{A}_t$
\mathbf{H}^*	new <i>static homography estimate</i> by method from Appendix 2.3
\mathbf{H}_t^*	new <i>dynamic homography estimate</i> for time t by methods from Sections 2.4-2.5
\mathbf{H}_t^s	projection (image to image) generated with the SIFT-FLANN matcher results

Table 2. Notations for various point sets and homographies utilized in this work.

6. Transform the originally labeled feature points \mathcal{I} , i.e. the four corner points of the lane markers, with the calculated \mathbf{H}_t^s (producing \mathcal{I}_t^s) and calculate the geometric center for each marker. This provides a seed point for rediscovery.
7. Select contour areas which include a seed point, thus creating a set of candidate for lane marks.
8. For each marker, select the 4 corner points on the corresponding contour that produce the largest area, thus creating a quadrilateral.
9. Filter the candidates by contour area, contour area to quadrilateral area ratio, ratio of area compared to the labeled reference area. This step is important to remove candidates which are merged, partially occluded, or otherwise not rediscovered correctly.
10. Select the proper labels and store the 4 corner points as feature points of each remaining markers. With this step the rediscovery steps are complete, yielding \mathcal{I}'_t .
11. Run a homography matrix estimator (which utilizes RANSAC) between the rediscovered \mathcal{I}'_t and the corresponding aerial \mathcal{A}_t points. Remove the feature points which are considered outliers by the algorithm. In the current implementation these are points of which are further than $\tilde{2}$ feet on the state plane. The resulting homography for time instance t is written as \mathbf{H}'_t .

The result of the listed process is a set of rediscovered feature points (\mathcal{I}'_t) and a new homography (\mathbf{H}'_t) for a given time instant t . Note that although the steps involve heavy filtering and outlier detection the resulting, a standalone homography should not be used without further processing for the following reasons:

- The number and positions of the rediscovered lane markings are not guaranteed, thus it is possible to only rediscover lane markings which lie on the same line, thus providing poor estimates for the perpendicular axis.
- Lane markings ”far” from the camera are hard to properly detect due to their small area and slight camera movements can cause misidentification.



Figure 7. An example of SIFT-FLANN based seed points from step 6 (red dots), detected contours (purple outlines) and re-detected dash corner points (yellow dots) for a 1-minute average frame. Because traffic at this time instance was partially stopped, some vehicle artifacts are visible in the averaged frame. A few dashes are not successfully rediscovered.

- The RANSAC based homography estimator does not guarantee that the selected points are proper. In extreme cases it can select an erroneous subset and fit a good homography to it. As discussed previously, the lane markings are non-unique and repetitive: usually there are multiple subsets of points with nearly identical relative arrangement.

2.3. Static Homography Generation

To counter for the long term homography errors and somewhat compensate for the sunflower effect a single homography can be generated for a given time interval, e.g.: the length of the recording or for the desired time range. This method has the advantage of time-invariance (simplicity) and can utilize all the instantaneous estimates over the specified window, resulting in ample redundancy for outlier removal. The method includes two steps:

1. In the preparation phase the outliers are removed, i.e. the homography estimates of which significantly differ from the others. The current implementation considers a homography an outlier if any of its component (in the 3x3 matrix) deviates more than 30% from the arithmetic mean of all estimates. Note that this process is iterative.
2. The main step is the averaging, i.e. calculating the arithmetic mean for each matrix component to produce \mathbf{H}^* .

2.4. Dynamic Homography Generation

To counter for both long and short term homography problems a dynamic, time dependent method is proposed. This solution is capable of modeling short-term camera motion (e.g. from the sunflower effect), but is less trivial implement and use compared to the static version because it requires modifying the homography as a function of time.

The underlying idea of this method is the "smoothing" of the original estimates, based on the temporally nearby values. Note that this is not straightforward because estimates are not necessarily available at all time instances, e.g.: in case of heavy traffic the occlusion prevents homography estimation, similarly at the beginning and at the end of the recording there is no preceding and subsequent information available. (This is especially problematic at dawn and dusk where the even the

recorded images are noisy and blurred thus making feature rediscovery hard or even impossible.) Thus a method is necessary of which is capable to adapt to these situations. An important and necessary property of a good dynamic homography is that it provides "smooth" variability over time, preventing discontinuities in positional data (which could in turn cause fragmentations during object tracking.)

The proposed solution includes three steps:

1. In the preparation phase the outliers are removed. This step is exactly the same as in the static case.
2. A "window size" parameter is computed, based on the available number of homography estimates in a specified temporal neighbourhood around the given time instance. This step ensures that in case of missing estimates the window size is larger, thus accommodating more data points.
3. The final step is the "smoothing" of each homography matrix component, where a Gaussian kernel function is used over time. The shape of the function, or more precisely the variance is determined by the previous step.

Note that this method can provide a homography estimate \mathbf{H}_t^* at any time instance, not just when instantaneous estimates are available. In our implementation a new estimate is generated at 10 second intervals.

2.5. SIFT-FLANN-based homography generation (existing)

No lane marking rediscovery is performed. Instead, the set of shifted points (\mathcal{I}_t^s) generated from the original labels (\mathcal{I}) is directly used from Appendix 2.2 step 6 and above to fit a homography estimate \mathbf{H}_t^* for time t i.e. $\mathcal{I} \xrightarrow{\mathbf{H}_t^s} \mathcal{I}_t^s \xrightarrow{\mathbf{H}_t^*} \mathcal{A}$.

3. Homography Error Metrics

Next, we turn to defining metrics for assessing the effectiveness and accuracy of the proposed point rediscovery and homography re-estimation metrics. This appendix provides an overview of the possible metrics and their properties.

3.1. Notation

In addition to the notation defined in the previous section, the comparison operator E is defined as a function of two sets of points ($\mathcal{B}, \mathcal{C} \subset \mathbb{R}^2$). The operator calculates the point-wise L2-norm between the common subset of points shared by \mathcal{B} and \mathcal{C} , (i.e. the distances between two points which correspond to the same feature), and produces a set of values. The resulting set can be used to calculate aggregate metrics, such as mean, maximum, and standard deviation of error.

$$E(\mathcal{B}, \mathcal{C}) = \{\|b_i, c_i\|_2, \dots, \|b_n, c_n\|_2\} \forall i \text{ s.t. } b_i \in \mathcal{B}, c_i \in \mathcal{C}, b_i \hat{=} c_i \quad (14)$$

3.2. Reference Metrics

Metrics listed in this appendix are computed by comparing the original reference labels (\mathcal{I} and \mathcal{I}') to rediscovered points (\mathcal{I}'_t). In some cases the same set of points is transformed with the original reference homography (\mathbf{H} or \mathbf{H}^\top) and the instantaneous homography estimate \mathbf{H}'_t . They measure i.) the quality of the rediscovery and instantaneous homography re-estimation process, and ii.) the movement of the cameras.

Most metrics presented here can be calculated in both image and state plane coordinate systems; thus we present them in pairs. The resulting error values have units of pixels or distance (feet), respectively. In practical evaluations we utilize the state plane distance (units of feet) as it is invariant to the image resolution and FOV, thus comparable across cameras. As in Appendix 1, we use a superscript (*im*) or (*st*) to denote the coordinate system for each metric.

- **Sub Drift I:** This metric compares a subset of the originally labeled *image* points to the rediscovered points. The resulting distances can be interpreted as the *drift* caused by the camera movement. Note that in case of perfect alignment the maximum error is zero.

$$\text{SubDriftI}^{(im)} = E(\mathcal{I}_t, \mathcal{I}'_t) \quad (15)$$

$$\boxed{\text{SubDriftI}^{(st)} = E(\mathcal{I}_t \xrightarrow{\mathbf{H}}, \mathcal{I}'_t \xrightarrow{\mathbf{H}'_t})} \quad (16)$$

- **Sub Drift II (not used):** This metric compares a subset of the originally labeled *aerial* points to the rediscovered points. The resulting distances are a combined error of the drift caused by the camera movement *and* the homography fitness error. Thus even in case of no movement the error is non-zero, as the homography is not a perfectly fit mapping of \mathcal{I} to \mathcal{A} .

$$\text{SubDriftII}^{(im)} = E(\mathcal{A}'_t \xrightarrow{\mathbf{H}^\top}, \mathcal{I}'_t) \quad (17)$$

$$\text{SubDriftII}^{(st)} = E(\mathcal{I}'_t \xrightarrow{\mathbf{H}}, \mathcal{A}_t) \quad (18)$$

- **Full Drift:** This metric compares the full set of the originally labeled image or aerial points transformed by the reference and the re-estimated homography. The resulting distances can be interpreted as *drift* caused by the camera movement. In the case of no camera movement, the error is zero (except for possible stochasticity in the optimal homography H'_t selected by the RANSAC algorithm utilized.)

$$\text{FullDrift}^{(im)} = E(\mathcal{A} \xrightarrow{\mathbf{H}^\top}, \mathcal{A} \xrightarrow{\mathbf{H}'_t}) \quad (19)$$

$$\boxed{\text{FullDrift}^{(st)} = E(\mathcal{I} \xrightarrow{\mathbf{H}}, \mathcal{I} \xrightarrow{\mathbf{H}'_t})} \quad (20)$$

- **Sub Fitness:** This metric shows the fitness of the estimated homography, utilizing the rediscovered points and a matching subset of aerial points. This error is caused by a combination of e.g.: inaccurate feature point rediscovery, lens distortion, non flat roadway.

$$\text{Fitness}^{(im)} = E(\mathcal{A}_t \xrightarrow{\mathbf{H}'_t}, \mathcal{I}'_t) \quad (21)$$

$$\boxed{\text{Fitness}^{(st)} = E(\mathcal{I}'_t \xrightarrow{\mathbf{H}'_t}, \mathcal{A}_t)} \quad (22)$$

3.3. Homography Re-estimation Metrics

Metrics listed here are derived from the metrics introduced at the previous section. Substituting the reference labels and homographies with the re-estimated homographies and points sets derived from them. The interpretation and purpose of these metrics are to qualify how good are the new estimations; i.e. a perfect homography provides a perfect alignment to the detections.

- **Sub Drift:** This metric compares a subset of points generated with the re-estimated homography (from the originally labeled points) to the rediscovered points. The resulting distances can be interpreted as a combined error of the re-estimation and the homography fitness error. Therefore even in case of a perfectly aligned homography the error is non-zero. Note that, because the new subset of points are generated from the reference labeled points this metric essentially merge Sub Drift I and II presented in the previous section.

$$\text{SubDrift}^{(im)} = E(\mathcal{I}_t^*, \mathcal{I}'_t) \implies E(\mathcal{A}_t \xrightarrow{\mathbf{H}_t^{*\neg}}, \mathcal{I}'_t) \quad (23)$$

$$\text{SubDrift}^{(st)} = E(\mathcal{I}_t^* \xrightarrow{\mathbf{H}_t^*}, \mathcal{I}'_t \xrightarrow{\mathbf{H}_t^*}) \implies E(\mathcal{A} \xrightarrow{\mathbf{H}_t^{*\neg}}, \mathcal{I}'_t \xrightarrow{\mathbf{H}_t^*}) \implies E(\mathcal{A}_t, \mathcal{I}'_t \xrightarrow{\mathbf{H}_t^*}) \quad (24)$$

- **Full Drift:** This metric compares the full set of the originally labeled image or aerial points transformed by the re-estimated and by the instantaneous homography. In effect, these original labeled points are chosen as “proxy points” for the real locations of the correspondence points in the image as they are a full set known to lie near the true locations. The resulting distances can be interpreted as the error of the homography re-estimation process. Note that in case of a perfect alignment the error is zero.

$$\text{FullDrift}^{(im)} = E(\mathcal{A} \xrightarrow{\mathbf{H}_t^{*\neg}}, \mathcal{A} \xrightarrow{\mathbf{H}_t^{\neg}}) \quad (25)$$

$$\boxed{\text{FullDrift}^{(st)} = E(\mathcal{I} \xrightarrow{\mathbf{H}_t^*}, \mathcal{I} \xrightarrow{\mathbf{H}_t^*})} \quad (26)$$

- **Fitness:** Since the fitness metrics does not depend on the re-estimated homography, they are equivalent to the ones discussed in the previous section.

The homography re-estimation performance can be measured by the Full and Sub Drift metrics, but both have caveats to consider: Sub Drift is a better measure of the *real magnitude of the error*, because it incorporates inaccuracies of both the homography and the homography fitness, e.g. errors caused by intrinsic camera problems and the non-flat roadway. On the other hand the Sub Drift metric only includes feature points which are re-detected, thus hard to detect points further from the pole are often missing from the set. This is crucial because the perspective those ”far” points account for larger state plane errors than the more easily detectable closer ones. The combined effect of these that the Sub Drift is better in the assessment of the expected minimal error, this is not the case for the maximum. Also worth noting is that Sub Drift does not explicitly rely on H'_t being well-fit, so produces a reliable estimate of error even when the fitting of H'_t fails (though in practice this usually indicates poor point rediscovery).

The Full Drift metric is useful to assess the performance of the *homography re-estimation* itself, because in case of perfect alignment the error is reduced to zero. In opposite to the Sub Drift this, metric includes all labeled points, some of which are actually outside the processed ROI, thus resulting in an apparently larger maximum error than during vehicle detection.

Based on these considerations, we select Full Drift (Equation 20) as our primary metric and use it to report the *error* for each method utilized as equation 2 the main text. We also utilize Sub Fitness (Equation 22) as a baseline measure for the error floor of each fit homography (*fitness* in equation (1) of the main text).

3.4. Additional Homography Metric Figures

Figure 8 shows the different state-plane error metrics for a camera with a long field of view. P05C06 is chosen as it clearly illustrates both long and short-term drift in the homography, as well as the relations between the different metrics: On Figure 8a the homography Fitness is shown, representing the minimum achievable accuracy for the vehicle detection. On Figure 8b the SubDriftI error is slightly higher than FullDrift shown on Figure 8c, because it contains both the fitness and homography error. In this case, the maximum SubDriftI values are higher than the maximum FullDrift values because the rediscovery rate for the feature points is $\sim 80\%$ for this particular case (relatively high), thus distant points are also likely included. If the drift

were more extreme, the most distant correspondence points might not be successfully rediscovered, and the overall reported SubDriftI error may be driven down as a result. This dependence on the number of rediscovered points is one reason why FullDrift is preferred to SubDrift in this work for comparing homography re-estimation performance.

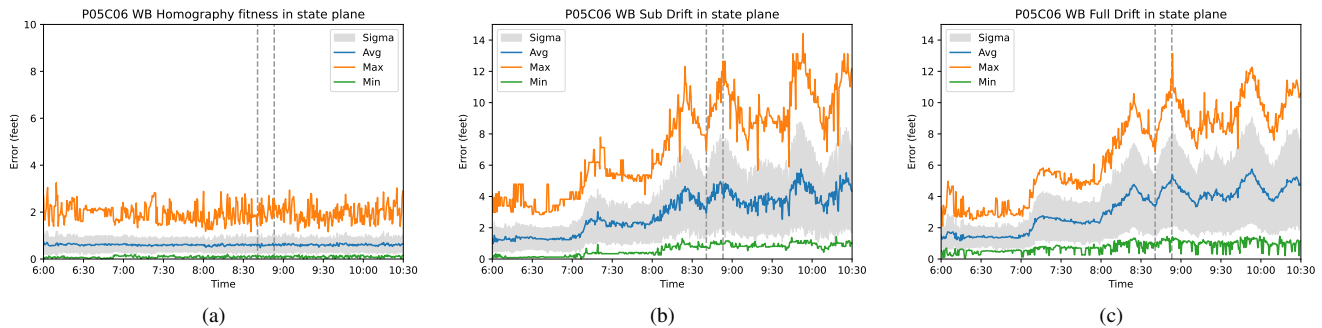


Figure 8. (Repeated from Section 2.) Goodness of fit and uncorrected drift between reference homography and true scene homography according to two metrics (Sub Drift and Full Drift) for a single typical camera. Horizontal grey dashes show the two time instances for Figure 6.

Figure 9, 10 shows the mean FullDrift and SubDrift calculated and averaged over the time of the recording, for all cameras in the system, sorted by magnitude, for each direction of roadway travel. Note that not all cameras have fields of view defined for both sides of the roadway (hence there are more total westbound fields of view than eastbound fields of view). The mean SubDrift is Lower than the mean FullDrift over all cameras, and the maximum mean SubDrift (over all correspondence points and times) is lower than for FullDrift. However as seen in Figure 8a, for some time instances the maximum SubDrift can be higher than the maximum FullDrift.

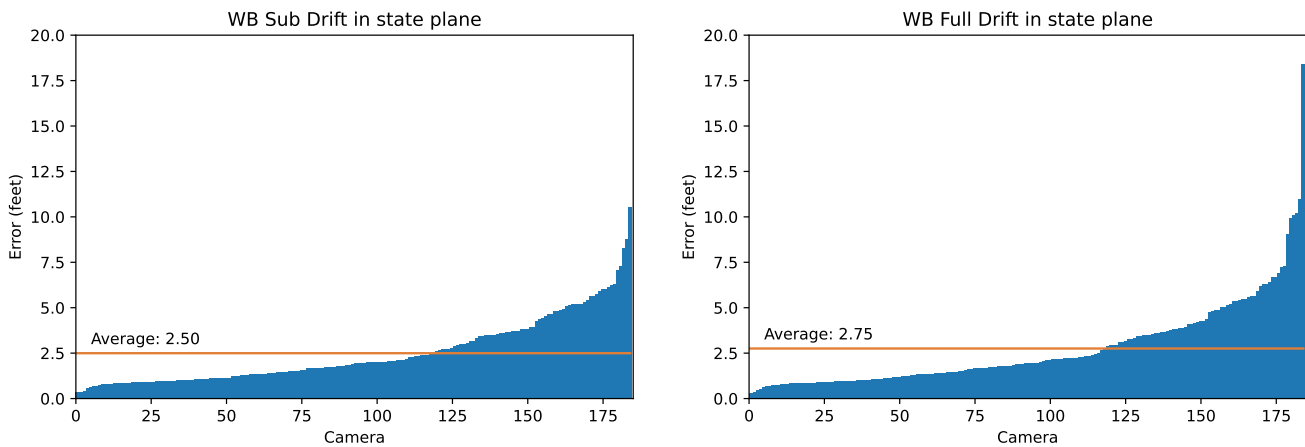


Figure 9. Mean averaged SubDrift and FullDrift calculated for all westbound cameras in the system, sorted by error magnitude.

The effectiveness of the homography re-estimation methods is illustrated on Figure 11, which shows the additional error above the error floor, utilizing the FullDrift metric, for two cameras. The reference (blue) indicates the resulting error without any mitigation, showing both long term (high mean) and short term (high variance) error. The SIFT-FLANN method (orange) illustrates the performance of an optical flow based "camera stabilization" solution. The static, all-day average method (green) removes the long term error, although it is mostly unable to remove the error caused by the sunflower effect. Finally since the dynamic homography (red) utilizes nearby (temporal) homography estimations for a given time instance, it can cope with short-term fluctuations. Note that in most cases the all-day average is superior to the SIFT-FLANN method, although there can be time instances where the former can produce better results. Figure 11 also illustrates, that long field of view cameras (e.g. P05C06) are more sensitive to camera movement, compared to mainly downwards-facing cameras (e.g. P05C04), thus the error caused by drift is more pronounced for them.

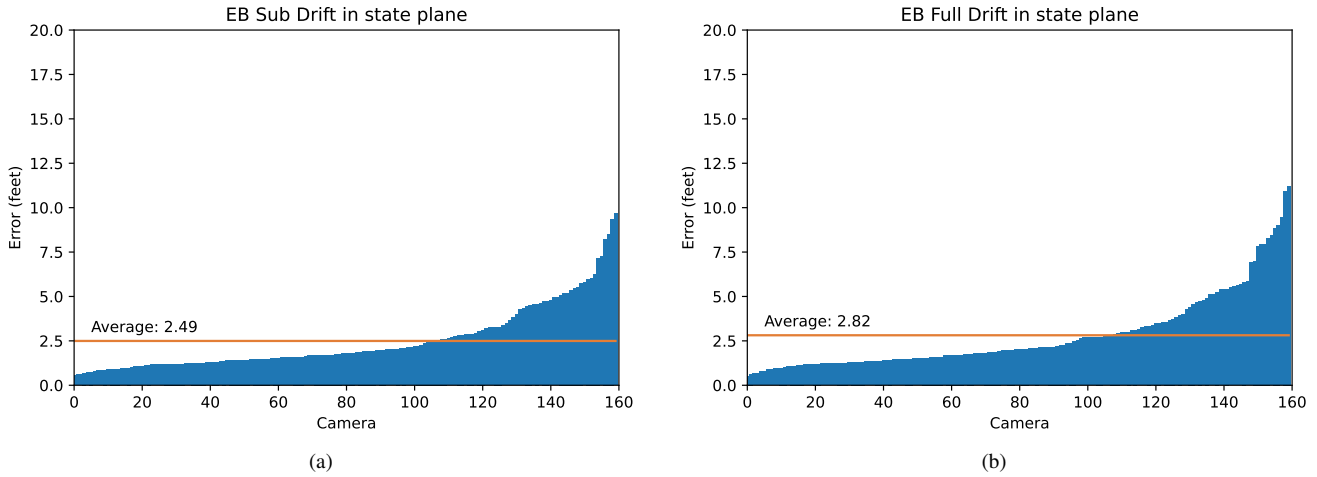


Figure 10. Mean averaged SubDrift and FullDrift calculated for all eastbound cameras in the system, sorted by error magnitude.

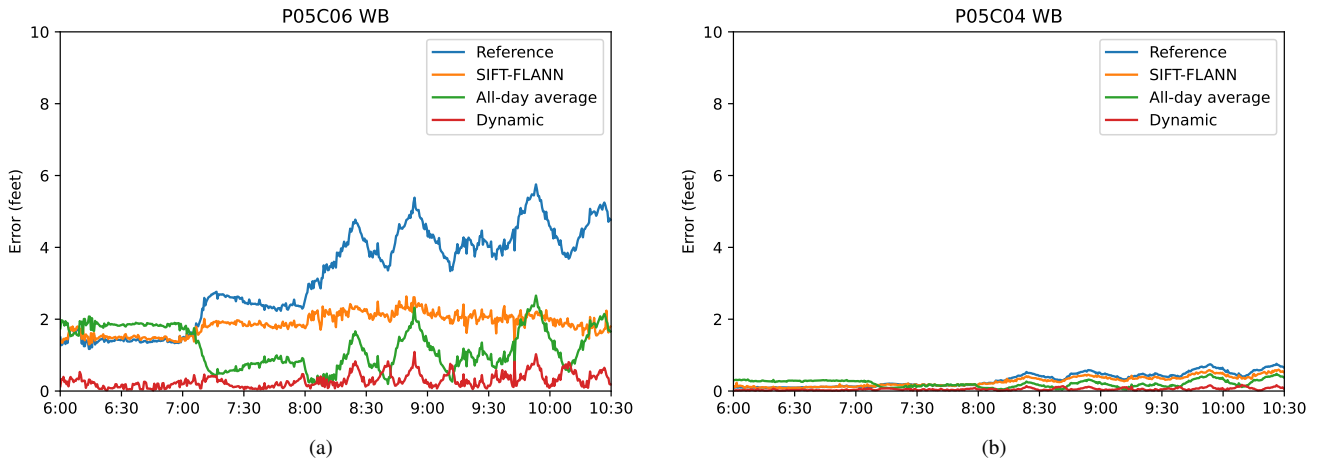


Figure 11. Error dynamics for two cameras over time with each homography re-estimation methods. Long field of view cameras e.g. 11a have more pronounced errors over down-looking cameras, e.g. 11b.

Lastly, figure 12 compares the remaining mean average SubDrift and FullDrift errors for each camera after (black) SIFT-FLANN feature-matching, (orange) one-day best fit homography re-estimation, and (red) dynamic homography re-estimation methods relative to original reference homography baseline (blue). The overall trends and relative performance amongst methods is unchanged; in most cases the SIFT-FLANN baseline outperforms the uncorrected reference homography, the all-day average outperforms SIFT-FLANN, and the dynamic method outperforms the all-day average.

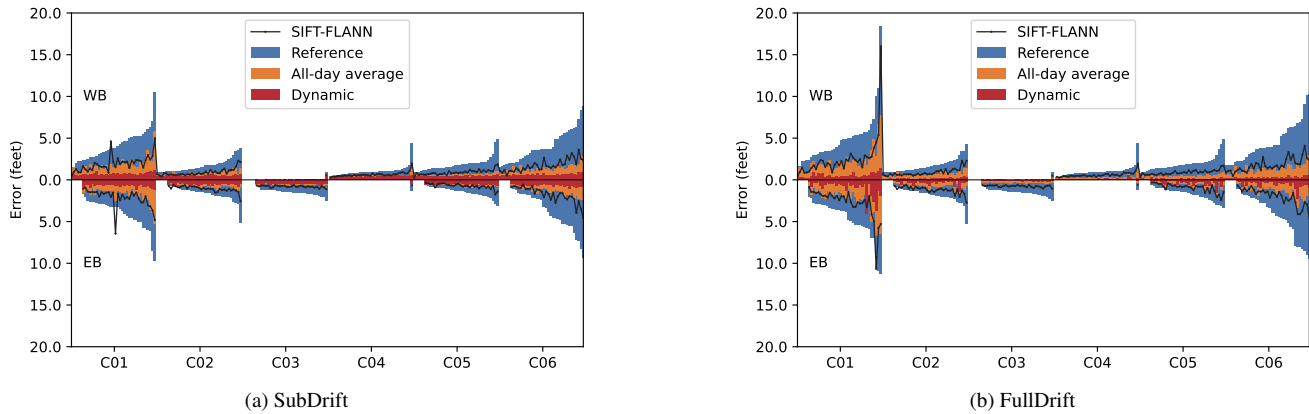


Figure 12. Remaining SubDrift and FullDrift errors for each camera after (black) SIFT-FLANN feature-matching, (orange) one-day best fit homography re-estimation, and (red) dynamic homography re-estimation methods relative to original reference homography baseline (blue). Cameras are grouped by position on pole (see Figure 2 in main text.) and by side of roadway (westbound homographies on top, eastbound on bottom.)

4. Additional GPS Trajectory Plots

Figure 13 shows plots for additional GPS tracks through the video scene. Manually annotated object positions (circles) and GPS positions (lines) for are shown for the overall scene, divided by westbound (top left) and eastbound (bottom left) direction of travel. (right) x-position relative to the corrected GPS track (top) and absolute y-position (bottom) are plotted for the highlighted GPS track. It can be seen that a.) the corrected GPS trajectories align more closely with object detections (black dots) and b.) there is significant deviation between the corrected and uncorrected GPS trajectories. Note that especially the Y-coordinate error in the uncorrected trajectories varies in character across different GPS trajectories.

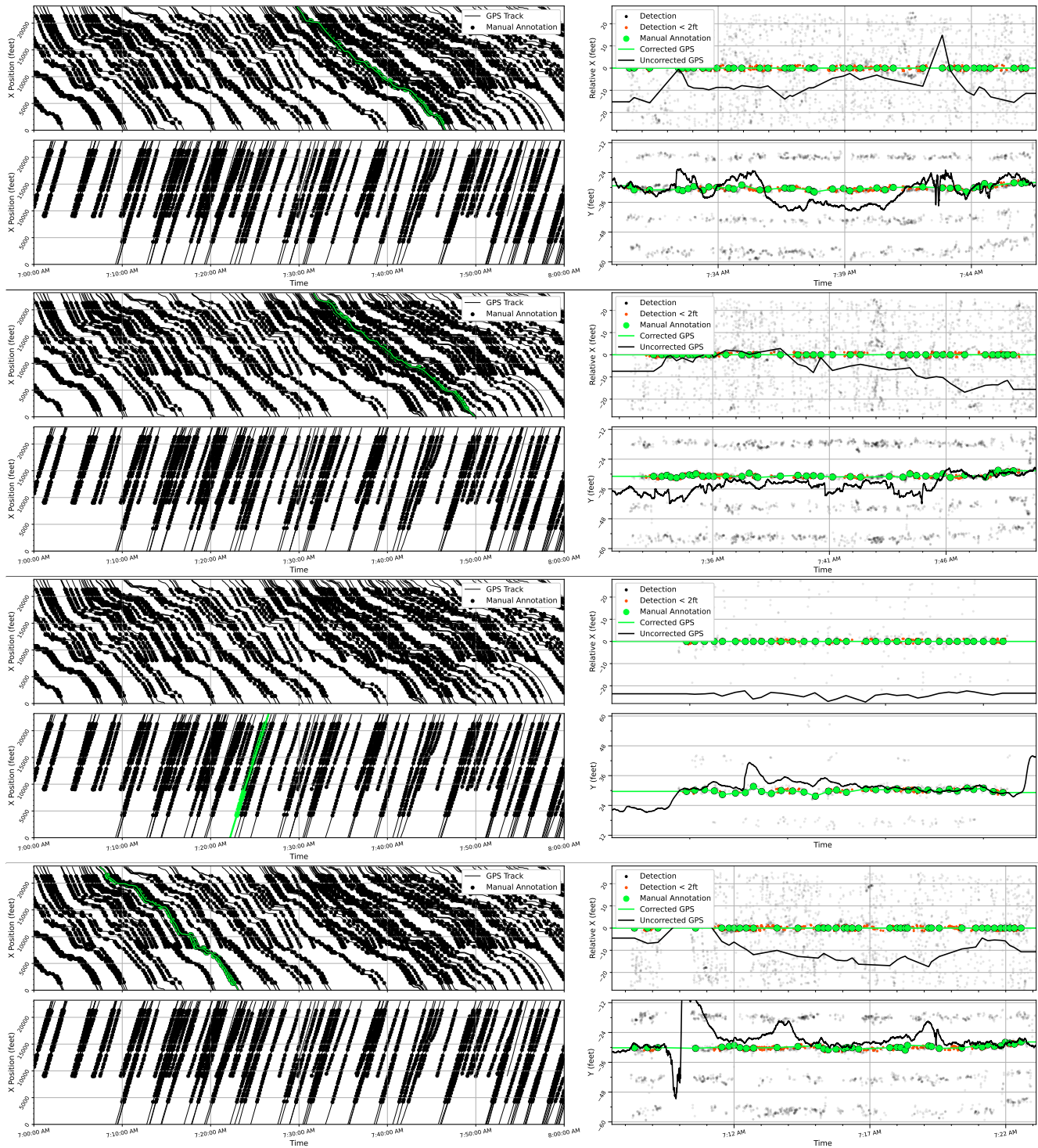


Figure 13. Plots for 4 individual GPS trajectories. In each sub-figure, **(top left)** shows westbound X-position and **(bottom left)** shows eastbound X-position for the whole scene duration. **(top right)** shows nearby (in relative X-coordinates) and **(bottom right)** shows nearby (in Y-coordinates) detections, manual annotations, and uncorrected GPS trajectory, relative to the corrected GPS trajectory.

5. Object Detector

5.1. 3D Bounding Box Parameterization

Object detections are produced by a Retinanet object detector with Resnet-50 FPN backbone [12]. We extend the first convolutional layer of this network to allow two input frames (the current and a rolling average frame). This network has two output heads: a classification head and a regression head, each of which outputs a set of outputs per anchor box. The shape of the classification head output C is $[n, c]$ where n is the number of anchor boxes in the network and c is the number of classes. For an anchor box i , the final output object class and object confidence are taken as:

$$\text{class}_i = \arg \max_c (C[i, :]) \quad (27)$$

$$\text{confidence}_i = \max(C[i, :]) \quad (28)$$

The shape of the regression head output R is $[n, 8]$. Let A_i parameterize a single anchor box i according to:

$$A[i, :] = [x_a, y_a, w_a, h_a] \quad (29)$$

denoting the x-coordinate, y-coordinate, width and height of the anchor box, respectively. The regression output corresponding to that anchor box $R[i, :]$ parameterizes a rectangular prism (ignoring the effects of perspective foreshortening) according to:

$$R[i, :] = [x_c, y_c, x_l, y_l, x_w, y_w, x_h, y_h] \quad (30)$$

where (x_c, y_c) indicates the center of the rectangular prism relative to the anchor box top left corner, x_l and y_l indicate the x-pixel and y-pixel components of the rectangular prism's length, x_w and y_w indicate the x-pixel and y-pixel components of the rectangular prism's width, and x_h and y_h indicate the x-pixel and y-pixel components of the rectangular prism's height. All components (e.g. x_l, y_l) are relative to the anchor box dimensions (in this case w_a and h_a). By convention, each dimension is directional and is measured relative to the rear bottom left corner. Figure 14 provides a visual overview of each component.

The (\mathbf{X}, \mathbf{Y}) pixel coordinates for back bottom left coordinate (x_{rbl}, y_{rbl}) within the overall image can then be written as:

$$\begin{aligned} x_{rbl} &= x_a + (x_c - x_l/2 - x_w/2 + x_h/2) * w_a \\ y_{rbl} &= y_a + (y_c - y_l/2 - y_w/2 + y_h/2) * h_a \end{aligned} \quad (31)$$

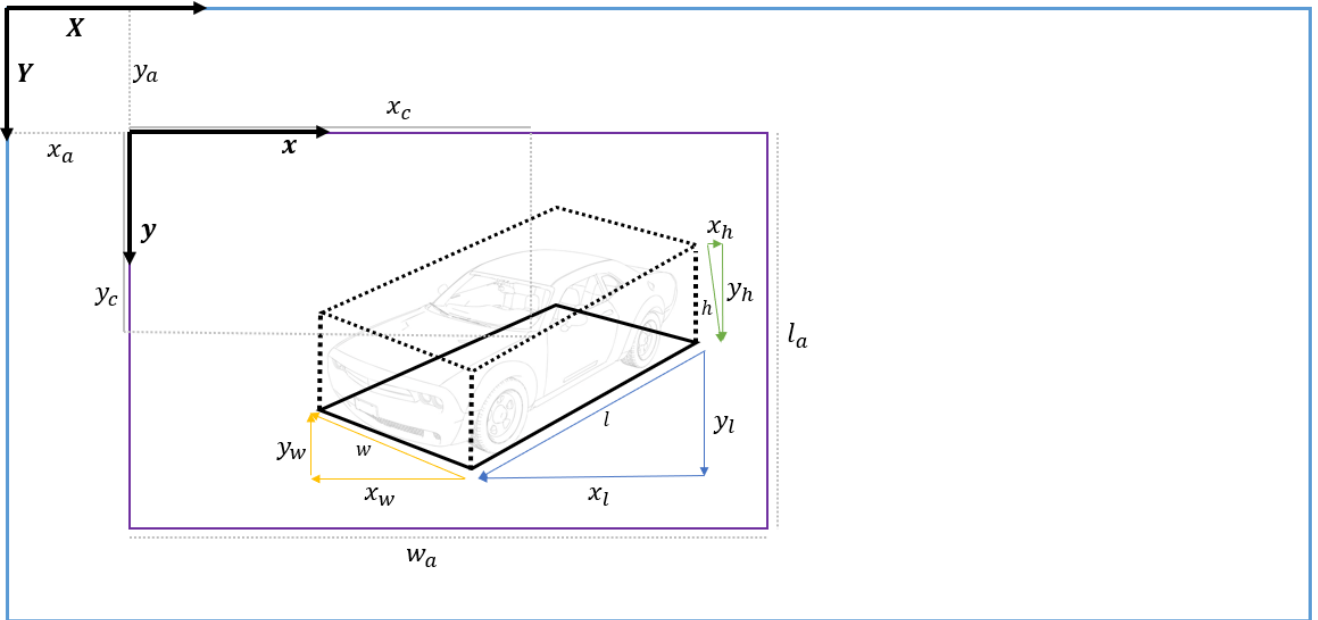


Figure 14. Bounding box *rectangular prism* parameterization. Anchor box (purple) reference coordinate system denoted as (\mathbf{x}, \mathbf{y}) and global frame (blue) reference coordinate system denoted as (\mathbf{X}, \mathbf{Y}) . Angle of h is exaggerated to show sub-components x_h and y_h .

The 7 other corner coordinates can be similarly written. The resulting rectangular prism can then be converted into state plane and roadway coordinates via the methods discussed in Appendix 1. Readers familiar with 3D detection in the autonomous vehicle context will likely question the use of 3D bounding boxes produced explicitly in state space, as in e.g. [9, 14]. In these contexts, a single camera is statically mapped to the world coordinate plane (i.e. the same pixel within an image always corresponds to the same point on the ground plane in world space). This is not the case in our application, in which a single CNN is desired to produce object detections for frames from all 234 cameras. Thus, a *viewpoint agnostic* anchor box / bounding box formulation such as the one used here is required. We also consider the direct regression of 3D bounding box corner coordinates as in [11] but empirically this results in poor performance.

5.2. Training

We train the above object detector on the I24-3D dataset [6]. We use the following loss formulation:

$$Loss = Loss_{2D} + \beta \times Loss_{3D} + Loss_{cls} + \gamma Loss_{vp} \tag{32}$$

where:

- $Loss_{2D}$ is intersection-over-union loss [13] for the minimum enclosing 2D bounding boxes for each of the predicted and target bounding box
- $Loss_{3D}$ is MSE loss computed between the target and predicted bounding box corners
- $Loss_{cls}$ is classification focal loss [12]
- β is a weighting coefficient, here set to 2.
- γ is a weighting coefficient, here set to 1/3

and $Loss_{vp}$ is a term designed to enforce the length, width, and height components of the predicted rectangular prism to align closely to the corresponding vanishing points by penalizing the angle between these vectors and the vanishing point directions (relative to the center of the predicted bounding box). **Succinctly, $Loss_{vp}$ is 0 when each bounding box is perfectly axis-aligned relative to the vanishing points, and is 1 when the axes of the bounding box are perfectly orthogonal to their respective vanishing points.** In this way, the training enforces the CNN to utilize visual cues from the image and align bounding boxes along these visual axes.

$$Loss_{vp} = 1/2 - \alpha_l((y_l(y_{vpl} - y_c)) + x_l(x_{vpl} - x_c)) / (2(x_{vpl} - x_c)(y_{vpl} - y_c)y_lx_l) + \tag{33}$$

$$1/2 - \alpha_w((y_w(y_{vpw} - y_c)) + x_w(x_{vpw} - x_c)) / (2(x_{vpw} - x_c)(y_{vpw} - y_c)y_wx_w) +$$

$$1/2 - \alpha_h((y_h(y_{vph} - y_c)) + x_h(x_{vph} - x_c)) / (2(x_{vph} - x_c)(y_{vph} - y_c)y_hx_h)$$

where (x_{vpl}, y_{vpl}) is the vanishing point, in pixel coordinates, aligned with the roadway direction of travel, (x_{vpw}, y_{vpw}) is the vanishing point, in pixel coordinates, perpendicular with the roadway direction of travel, and (x_{vph}, y_{vph}) is the vanishing point, in pixel coordinates for vertically aligned lines within the frame, α_l is -1 if the rear of the bounding box is closer to (x_{vpl}, y_{vpl}) than the front of the bounding box and 1 otherwise, α_w is -1 if the left side of bounding box is closer to (x_{vpw}, y_{vpw}) than the right side of the bounding box and 1 otherwise, and α_h is -1 if the bottom of the bounding box is closer to (x_{vph}, y_{vph}) than the front of the bounding box and 1 otherwise.

Figures 15 and 16 show the resulting detection set in roadway coordinates. Each diagram plots roadway X-position versus time for all detections on the given roadway direction. In some cases, more than one detection may be mapped to a single pixel because they correspond to two detections occupying nearly the same X-position at the same time, in different lateral (lane) positions. In these figures, horizontal bands without detections correspond either to missing camera poles (see Appendix 8 or overpasses. Full-resolution images are included in supplementary material as separate files.

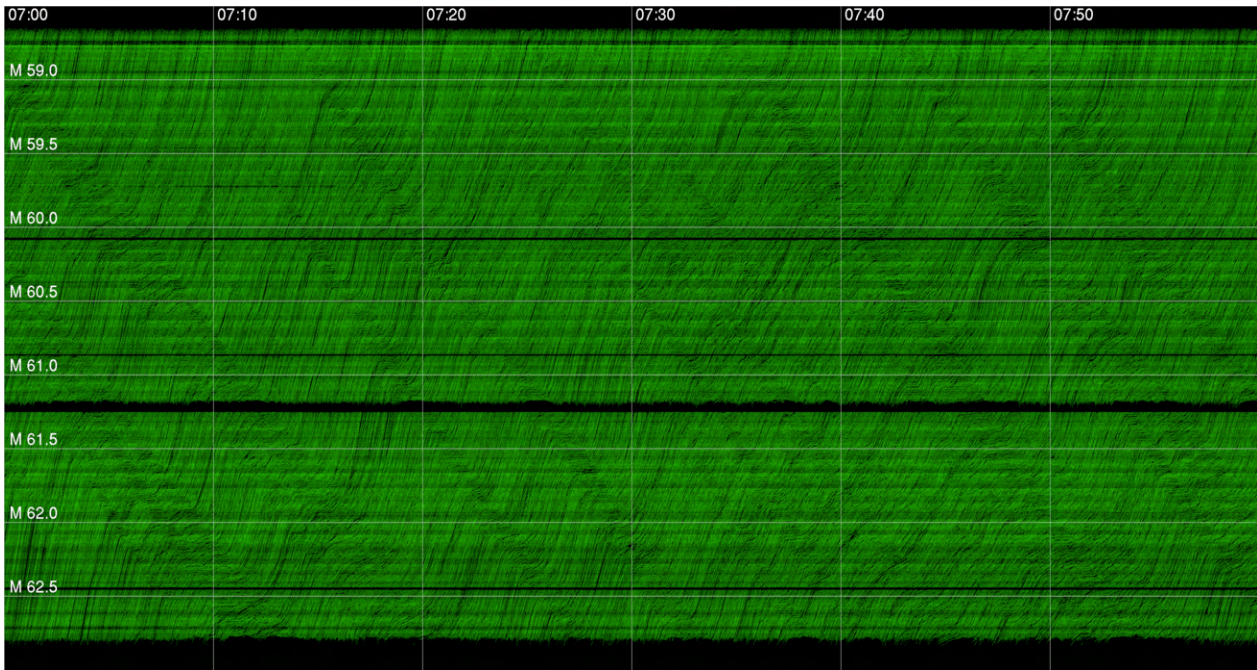


Figure 15. Longitudinal (X) position versus time for all detections on the westbound side. Each colored pixel (a total of 123,768,540 though with some overlaps for vehicles in different lanes but the same X position at the same time) represents a detected vehicle in a particular location and time. (A full resolution image is included in supplementary files, with scale 1 pixel = 2 feet and 0.16 sec).

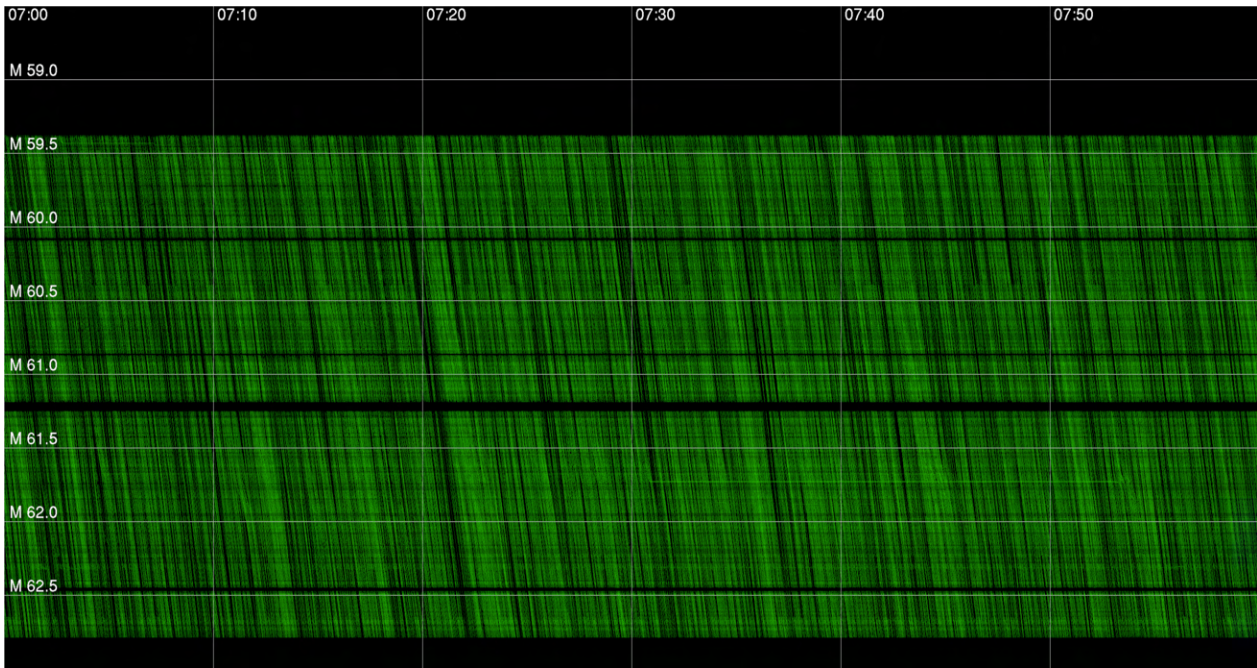


Figure 16. Longitudinal (X) position versus time for all detections on the eastbound side. Each colored pixel (a total of 123,768,540 though with some overlaps for vehicles in different lanes but the same X position at the same time) represents a detected vehicle in a particular location and time. (A full resolution image is included in supplementary files, with scale 1 pixel = 2 feet and 0.16 sec).

6. Experimental Details

6.1. Evaluation Protocol

Each object tracker is run using the detection set from Appendix 5. GPS trajectories and detections from each camera are obtained at slightly different times. To account for this, tracking evaluation is performed at fixed 0.1 second intervals, and each GPS trajectory and object tracklet position is linearly interpolated at each evaluation time. GPS trajectories extend somewhat outside of the temporal duration for which detections are available. Evaluation is performed only over the temporal range for which both GPS trajectories and detections exist. Moreover, each GPS trajectory is clipped in the X-range $[0, 23000]$ such that the trajectory is always visible within the field of view of the overall camera network. Evaluation is performed as in [1]. For all metrics other than HOTA metrics, a lax IOU of 0.1 is required for an object tracklet and GPS trajectory to be matched, and matching is performed with the Hungarian Algorithm for bipartite matching [10]. For each object tracklet, we use the median reported dimension (l, w , and h , in feet) over all reported tracklet dimension measurements.

6.2. Parameter Settings

Table 3 lists relevant parameter settings for each implemented algorithm. Kalman filter parameters were empirically fine-tuned using the I24-3D dataset [6].

Algorithm	Parameter	Value	Description
ALL	t_{max}	2	Maximum time (sec) between detections before track is terminated
	t_{min}	2	Minimum track length (sec)
	f_{eval}	10	Evaluation time step (Hz)
SORT [2]	σ_{high}	0.5	Required object confidence to be included in detection set
	ϕ_{nms}	0.1	Non-maximal suppression IOU threshold
	d_{max}	10	Maximum allowable distance for a match (ft)
	f_{track}	10	Tracking time step (Hz)
IOUT [3]	σ_{high}	0.5	Required object confidence to be included in detection set
	ϕ_{nms}	0.1	Non-maximal suppression IOU threshold
	ϕ_{min}	0.1	Minimum IOU for a match
	f_{track}	15	Tracking time step (Hz)
KIOU [5]	σ_{high}	0.5	Required object confidence to be included in detection set
	ϕ_{nms}	0.1	Non-maximal suppression IOU threshold
	ϕ_{min}	0.1	Minimum IOU for a match
	f_{track}	10	Tracking time step (Hz)
ByteTrack (L2) [17]	σ_{high}	0.01	Required object confidence to be included in detection set
	ϕ_{nms}	0.1	Non-maximal suppression IOU threshold
	d_{max}	10	Maximum allowable distance for a match (ft)
	τ_{high}	0.4	Required confidence to be included in first matching step
	f_{track}	10	Tracking time step (Hz)
ByteTrack (IOU) [17]	σ_{high}	0.01	Required object confidence to be included in detection set
	ϕ_{nms}	0.1	Non-maximal suppression IOU threshold
	ϕ_{min}	0.1	Minimum IOU for a match
	τ_{high}	0.4	Required confidence to be included in first matching step
	f_{track}	10	Tracking time step (Hz)
Oracle	ϕ_{min}	0.1	Minimum IOU for a match
	f_{track}	10	Tracking time step (Hz)

Table 3. Caption

7. Treatment of Personally Identifiable Information

As stated in [7], the primary purpose of the I-24 system is not to produce video data, but rather is to produce vehicle trajectory data. The data management plan for the system states that in general, video data is not released to the public or stored for long periods of time, but occasionally video data may be released to enable work on training, testing, and validation of trajectory generation algorithms. Thus, the video released in this work represents an edge case for the system,

Nevertheless, the release of video data (or any data with individuals represented) carries with it the risk of releasing *personally identifiable information* (PII) on the included individuals. In this work we make every effort to prevent the leakage of PII to the public, and furthermore make it infeasible to automatically extract PII such that doing so becomes extremely onerous to potential bad actors. A three-tiered approach is used: 1.) We release, more or less, a random hour of data, such that the potential for capturing a discrete event of interest to a third party is near zero. 2.) We automatically redact all license plate information from all visible vehicles. 3.) We manually redact any regions containing private property or visible people.

Even in un-redacted video data, license plates are in almost all cases impossible to read. Figure 17 provides a typical camera field of view, with the license plate only about 15 pixels wide and subject to significant blurring from vehicle motion. (The vehicle used in this image is part of the GPS instrumented vehicle fleet and does not belong to an individual). Nevertheless, we run an off-the-shelf license plate redaction algorithm [15] on all frames and cover all detected license plates with a black rectangle.

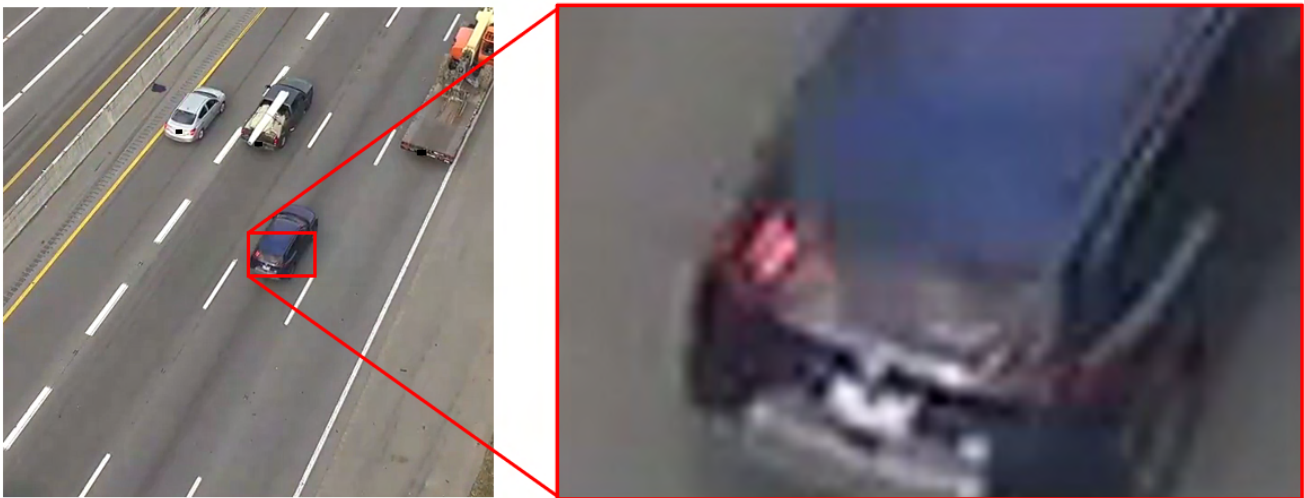


Figure 17. Example license plate from this dataset before redaction. License plate information is unrecoverable; for reference, the orange numbers on the rear window of the vehicle are about 5 times as large as license plate text and are barely discernible. (The pictured vehicle is part of the GPS instrumented vehicle fleet and does not belong to an individual).

We then manually inspect each video and redact the following sets of information. These areas are blurred in the released videos. Polygons defining the redacted regions are also released such that data users can replace these pixel values as desired for computer vision applications. Figure 18 shows an example of redacted regions.

- All visible people
- All private property
- Any stopped law enforcement vehicle (we leave ~ 1 sec of the stopped vehicle visible to allow for graceful handling of these vehicles by tracking algorithms).



Figure 18. Example redacted regions (red outline, blurred) for one camera field of view .

8. Known Data Artifacts and Anomalies

Here, we report a list of known anomalies and artifacts in the data as of submission time:

- Visible in some cameras on poles 3 and 4, there is a region that is blacked out in all frames and camera fields of view. This region corresponds to a private residence. Different from other redaction areas in this dataset, a virtual mask is applied to this region in all cameras in which it is visible, such that no visual information from this region is ever recorded. Other regions are redacted after recording.
- All cameras from Pole 25 are missing; this pole was struck by a vehicle a few days prior to the recording day and could not be restored in time.
- Homographies on the eastbound roadway side are not defined for cameras on poles 1-7. Construction work required that temporary solid lane markings be painted, for which i.) corner points were not uniquely distinguishable and ii.) no matching aerial imagery exists due to the short term nature of the construction work.
- Reference homographies for cameras on poles 1 and 2 are defined, but homographies are not re-estimated for these cameras. The lane markings were altered between the reference homography day and the recording day meaning that SIFT-FLANN matching and lane marking re-detection fail.
- Camera P22C04 has a black ring visible on the left portion of the frame due to a mechanical misalignment of the camera lens and body.
- Occasionally, GPS trajectories have missing recordings for time periods on the order of ~ 1 sec. The onboard sensor filtering attempts to compensate for this missing data, producing “sawtooth” artifacts in the trajectory. Figure 19 shows an example. The majority of these artifacts were removed during data refinement, but due some artifacts may still remain.

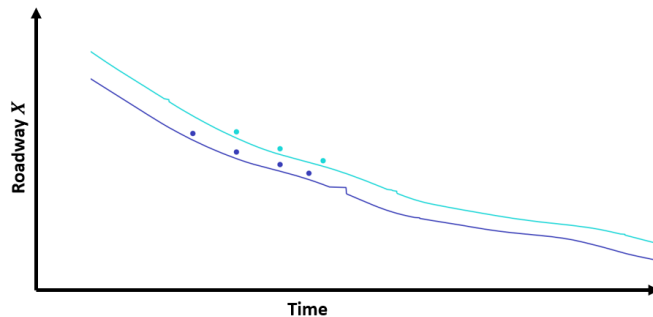


Figure 19. “Sawtooth” artifact in uncorrected GPS trajectories (lines). Circles depict corresponding manually annotated vehicle positions.

References

- [1] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008:1–10, 2008. 23
- [2] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE, 2016. 23
- [3] Erik Bochinski, Volker Eiselein, and Thomas Sikora. High-speed tracking-by-detection without using image information. In *2017 14th IEEE international conference on advanced video and signal based surveillance (AVSS)*, pages 1–6. IEEE, 2017. 23
- [4] Gary Bradski. The opencv library. *Dr. Dobb’s Journal: Software Tools for the Professional Programmer*, 25(11):120–123, 2000. 3, 10
- [5] S. Chen and C. Shao. Python implementation of the kalman-iou tracker., 2017. <https://github.com/siyuanc2/kiout>. 23
- [6] Derek Gloudemans, Gracie Gumm, Yanbing Wang, William Barbour, and Daniel B. Work. The interstate-24 3d dataset: a new benchmark for 3d multi-camera vehicle tracking. *arXiv preprint arXiv:2308.14833*, 2023. 2, 21, 23
- [7] Derek Gloudemans, Yanbing Wang, Junyi Ji, Gergely Zachar, Will Barbour, and Daniel B Work. I-24 motion: An instrument for freeway traffic science. *arXiv preprint arXiv:2301.11198*, 2023. 2, 24
- [8] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 2, 4
- [9] Hou-Ning Hu, Qi-Zhi Cai, Dequan Wang, Ji Lin, Min Sun, Philipp Krahenbuhl, Trevor Darrell, and Fisher Yu. Joint monocular 3d vehicle detection and tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5390–5399, 2019. 21
- [10] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 23
- [11] Peixuan Li, Huaici Zhao, Pengfei Liu, and Feidao Cao. Rtm3d: Real-time monocular 3d detection from object keypoints for autonomous driving. In *European Conference on Computer Vision*, pages 644–660. Springer, 2020. 21
- [12] 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*, pages 2980–2988, 2017. 20, 21
- [13] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 658–666, 2019. 21
- [14] Thomas Roddick, Alex Kendall, and Roberto Cipolla. Orthographic feature transform for monocular 3d object detection. *arXiv preprint arXiv:1811.08188*, 2018. 21
- [15] Sergio M Silva and Cláudio Rosito Jung. A flexible approach for automatic license plate recognition in unconstrained scenarios. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):5693–5703, 2021. 24
- [16] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. 7
- [17] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*, pages 1–21. Springer, 2022. 23