Using Computer Vision and Street-Level Videos for Pedestrian-Vehicle Tracking and Behaviour Analysis

Roberto Ventura[a]*, Stella Roussou[b], Apostolos Ziakopoulos[b], Benedetto Barabino[a], and George Yannis[b]

[a] *Department of Civil, Environmental, Architectural Engineering and Mathematics (DICATAM) - University of Brescia - Via Branze, 43 – 25123 Brescia (Italy).*

[b]*Department of Transportation Planning and Engineering - National Technical University of Athens (NTUA) - 5, Iroon Polytechniou - GR-15773 Athens (Greece).*

SUPPLEMENTAL MATERIALS

1.Define Homography Matrix (Step 1.3)

This section provides more details about Step 1.3, which involves defining an homography matrix to map points from the video plane (video coordinates) to the ground plane (real-world coordinates).

Let:

- H be the homography matrix and h_{ij} be the element corresponding to row i and column j. H is a 3×3 matrix, which accounts for transformations such as rotation, translation, and scaling between these planes i and j.
- V be the set of points in the video plane (video coordinates) and $v \in V$ be a generic point of coordinates $(x_v, y_v, 1)$[1].
- G be the set of points in the ground plane (real-world coordinates) and $g \in G$ be a generic point of coordinates $(x_g, y_g, 1)$.

In this phase, the homography matrix H is generated through the following process:

1. **Selecting Reference Points:**

The user selects at least four points in a specific video frame (i.e., V), corresponding to known points on the ground plane (i.e., G). These points must be carefully chosen to match between the video and the real-world map, ensuring correct alignment.

2. **Fitting the Homography Matrix:**

Once the corresponding video and ground points are selected, the framework uses these pairs to compute the homography matrix H (MathWorks, 2024a). This is achieved by solving a system of linear equations that maps the video points to the ground points using a projective transformation. More formally, for each point correspondence, the relationships between video points and ground points via the homography matrix H can be written as:

[1] The last component in the coordinates is set to 1 to represent the affine coordinates in a projective space. This simplifies transformations such as translation, rotation, and scaling, which can then be handled using matrix multiplication. The use of 1 ensures that the transformations occur in a uniform way between the image plane and the ground plane.

$$\begin{pmatrix} x_g \\ y_g \\ 1 \end{pmatrix} = H \cdot \begin{pmatrix} x_v \\ y_v \\ 1 \end{pmatrix}; \quad \forall (v, g) \in V \times G\ ; \qquad \text{(SM 1)}$$

This relationship expands into two equations per each point correspondence:

$$h_{11}x_v + h_{12}y_v + h_{13} = x_g \cdot (h_{31}x_v + h_{32}y_v + 1); \quad \forall (v, g) \in V \times G\ ; \qquad \text{(SM 2)}$$

$$h_{21}x_v + h_{22}y_v + h_{23} = y_g \cdot (h_{31}x_v + h_{32}y_v + 1); \quad \forall (v, g) \in V \times G\ ; \qquad \text{(SM 3)}$$

Given at least four non-collinear points in V and G, this system of equations can be solved to determine the matrix H, which transforms all points from the video plane to their corresponding points on the ground plane.

2.Detect Objects and extract features (Step 2.1)

This section provides some details about the detection data generated in Step 2.1.

Let:

- $N(t)$ be the set of objects detected in frame $f(t)$ at time $t \in T$. These are the raw objects obtained from the object detection model (i.e., YOLOv8). Each element $n \in N(t)$ represent a detected object at time $t \in T$.
- $D(t)$ be the set of detection data in frame $f(t)$. Each element $d_n \in D(t)$ represents the detection data associated to the detected object $n \in N(t)$ at time $t \in T$.
- b_n be the bounding box of the detected object $n \in N(t)$.
- $x_{b_n,top}(t)$ and $y_{b_n,top}(t)$ be the video coordinates of the top-left corner of the bounding box b_n at time $t \in T$, expressed in pixels in the cartesian space of the frame $f(t)$.
- w_n and h_n be width and height of the bounding box b_n, respectively, in pixels.
- h_n be height of the bounding box b_n in pixels.
- c_n be the confidence score associated with the detection of the object $n \in N(t)$.

Each bounding box b_n is a rectangle in the cartesian space of the frame $f(t)$ and is fully described by the following vector:

$$b_n \stackrel{\text{def}}{=} \left[x_{b_n,top}(t), y_{b_n,top}(t), w_n, h_n\right]; \quad \forall n \in N(t); \qquad \text{(SM 4)}$$

Thus, the set of object detections $D(t)$ is defined as in Eqn. (SM 5):

$$D(t) \stackrel{\text{def}}{=} \{(b_n, c_n) \mid n \in N(t)\}; \quad \forall t \in T; \qquad \text{(SM 5)}$$

3.Predict Positions with Kalman Filter (Step 2.2)

This section provides more details about Step 2.2, which involves predicting the future positions of the detected objects by applying the Kalman filter.

Let:

- $M(t)$ be the set of tracked objects at time $t \in T$. These are objects whose identities are maintained over time through tracking. Each element $m \in M(t)$ represent a tracked object at time $t \in T$, which may correspond to a detection $n \in N(t)$ after the matching process.

- $s_m(t)$ and $\hat{s}_m$ be the observed and estimated state vector, respectively, for the tracked object $m \in M(t)$ at time $t \in T$.
- $x_{b_m,top}(t)$ and $y_{b_m,top}(t)$ be the video coordinates of the top-left corner of the bounding box b_m, for the tracked object $m \in M(t)$ at time $t \in T$.
- w_m and h_m be width and height of the bounding box b_m, respectively, in pixels.
- $x_{b_m,bot}(t)$ and $y_{b_m,bot}(t)$ be the video coordinates of the bottom-centre point of the bounding box b_m, for the tracked object $m \in M(t)$ at time $t \in T$.
- $v_{x_m}(t)$ and $v_{y_m}(t)$ be the velocities in the x and y directions, respectively, for the tracked object $m \in M(t)$ at time $t \in T$.
- A be the state transition matrix.
- $z_m(t)$ be the observation vector for the tracked object $m \in M(t)$ at time $t \in T$.
- $x^{obs}_{b_m,bot}(t)$ and $y^{obs}_{b_m,bot}(t)$ be the observed video coordinates of the bottom-centre point of the bounding box b_m, for the tracked object $m \in M(t)$ at time $t \in T$.
- $z_m(t)$ be the observation vector for the tracked object $m \in M(t)$ at time $t \in T$.
- O be the observation matrix.
- $P_m(t)$ be the error covariance matrix for the tracked object $m \in M(t)$ at time $t \in T$, representing the uncertainty in the state estimate.
- $Q_m(t)$ be the process noise covariance matrix for the tracked object $m \in M(t)$ at time $t \in T$, representing the uncertainty in the model (e.g., due to acceleration or other unmodeled dynamics).
- $R_m(t)$ be the measurement noise covariance matrix for the tracked object $m \in M(t)$ at time $t \in T$, representing the uncertainty in the observations.
- $K_m(t)$ be the Kalman Gain process for the tracked object $m \in M(t)$ at time $t \in T$, which determines how much the predictions are adjusted based on the new measurements.
- $\varepsilon_m(t)$ be the measurement residual (or error) for the tracked object $m \in M(t)$ at time $t \in T$.
- $smoothed(\varepsilon_m(t))$ be the "smoothed" residual (or error) for the tracked object $m \in M(t)$ at time $t \in T$, used in the adaptive noise tuning process.
- I be the identity matrix.
- $adaptiveFactor$ be a tuning parameter that influences the rate of adjustment for the covariance matrices $Q_m(t)$ and $R_m(t)$ in response to changes in the smoothed error, used in the adaptive noise tuning process. It determines how aggressively the filter adapts to changing conditions.
- $maxExpectedDeviation$ be the maximum expected deviation from the predicted state, which serves as a benchmark for assessing the magnitude of errors in the measurements during the adaptive noise tuning process.
- τ_1 and τ_2 be the thresholds used in the adaptive noise tuning process to decide whether increasing or decreasing $Q_m(t)$, respectively.

State Vector

In this algorithm, the Kalman filter state vector is applied to the bottom-centre point of the bounding box, which corresponds to the position where the object touches the ground (e.g., the feet of a pedestrian or the wheels of a vehicle). This point is more representative of ground plane projections and behaviour analysis.

The bottom-centre coordinates $x_{b_m,bot}(t)$ and $y_{b_m,bot}(t)$ are calculated by considering the width w_m and height h_m of the bounding box b_m, as follows:

$$x_{b_m,bot}(t) = x_{b_m,top}(t) + \frac{w_m}{2}; \quad \forall t \in T; \ \forall m \in M(t); \qquad \text{(SM 6)}$$

$$y_{b_m,bot}(t) = y_{b_m,top}(t) + h_m; \quad \forall t \in T; \ \forall m \in M(t); \qquad \text{(SM 7)}$$

Then, the state vector $s_m(t)$ is defined as in Eqn. (SM 8):

$$s_m(t) = \begin{bmatrix} x_{b_m,bot}(t) \\ y_{b_m,bot}(t) \\ v_{x_m}(t) \\ v_{y_m}(t) \end{bmatrix}; \quad \forall t \in T; \ \forall m \in M(t) \qquad \text{(SM 8)}$$

State Transition Model

The state transition model defines how the object's state evolves to a given time step $t \in T$ from the previous $(t - \Delta t) \in T$ in the absence of any control inputs. It captures the dynamics of the object's motion based on the assumption of constant velocity. The state transition matrix A is constructed to represent these dynamics, as shown in Eqn. (SM 9):

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \qquad \text{(SM 9)}$$

This matrix models the kinematics of the object under the assumption of constant velocity, where:

- The position at time $t \in T$ is updated based on the previous position and velocity:

$$x_{b_m,bot}(t) = x_{b_m,bot}(t - \Delta t) + v_{x_m}(t - \Delta t)\Delta t; \quad \forall t \in T; \forall m \in M(t); \qquad \text{(SM 10)}$$

$$y_{b_m,bot}(t) = y_{b_m,bot}(t - \Delta t) + v_{y_m}(t - \Delta t)\Delta t; \quad \forall t \in T; \ \forall m \in M(t); \qquad \text{(SM 11)}$$

- The velocity stays constant:

$$v_{x_m}(t) = v_{x_m}(t - \Delta t); \quad \forall t \in T; \ \forall m \in M(t); \qquad \text{(SM 12)}$$

$$v_{y_m}(t) = v_{y_m}(t - \Delta t); \quad \forall t \in T; \ \forall m \in M(t); \qquad \text{(SM 13)}$$

This linear model provides a reasonable approximation for short time intervals (i.e., $\Delta t \approx 0$), which fit the observation window of the present analysis.

Observation Model

The observation model relates the true state of the system to the observed measurements. In the case of this algorithm, the measurements are the observed positions of the object from the detections $D(t)$.

The observation vector $z_m(t)$ consists of the observed positions at time $t \in T$:

$$z_m(t) = \begin{bmatrix} x^{obs}_{b_m,bot}(t) \\ y^{obs}_{b_m,bot}(t) \end{bmatrix}; \quad \forall t \in T; \ \forall m \in M(t); \qquad \text{(SM 14)}$$

The observation matrix O maps the state vector $s_m(t)$ to the observation vector $z_m(t)$:

$$O = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}; \qquad \text{(SM 15)}$$

This matrix indicates that the model observes only the position components of the state vector, and the velocity components are not directly observed.

Kalman Filter Equations

The Kalman filter operates in two phases: Prediction and Update.

A. Kalman Prediction Phase

Firstly, this phase predicts the state estimate at time $t \in T$ based on the estimate from time $(t - \Delta t) \in T$:

$$\hat{s}_m(t|t-\Delta t) = A \cdot s_m(t-\Delta t); \quad \forall t \in T; \ \forall m \in M(t); \qquad \text{(SM 16)}$$

Eqn. (16) projects the previous state estimate forward in time using the state transition model.

Secondly, the phase predicts the error covariance matrix:

$$P_m(t|t-\Delta t) = A \cdot P_m(t-\Delta t) \cdot A^T + Q_m(t); \quad \forall t \in T; \ \forall m \in M(t); \qquad \text{(SM 17)}$$

B. Kalman Update Phase

Firstly, this phase computes the Kalman Gain $K_m(t)$, which determines how much the predictions are adjusted based on the new measurements:

$$K_m(t) = P_m(t|t-\Delta t) \cdot O^T \cdot \left(O \cdot P_m(t|t-\Delta t) \cdot = O^T + R_m(t)\right)^{-1}; \quad \forall t \in T; \ \forall m \in M(t); \text{(SM 18)}$$

The Kalman Gain balances the uncertainties in the prediction and the measurement.

Secondly, the phase updates the state estimate $s_m(t)$ using the new measurement $z_m(t)$:

$$s_m(t) = \hat{s}_m(t|t-\Delta t) + K_m(t) \cdot \left(z_m(t) - O \cdot \hat{s}_m(t|t-\Delta t)\right); \quad \forall t \in T; \ \forall m \in M(t); \qquad \text{(SM 19)}$$

In Eqn. (SM 19), the term $z_m(t) - O \cdot \hat{s}_m(t|t-\Delta t)$ represents the measurement residual $\varepsilon_m(t)$, i.e., the discrepancy between the predicted measurement and the actual measurement. Thus, the state estimate is adjusted by the Kalman Gain multiplied by the measurement residual.

Thirdly, the phase updates the error covariance matrix $P_m(t)$ to reflect the reduced uncertainty after incorporating the measurement:

$$P_m(t) = (I - K_m(t) \cdot O) \cdot P_m(t|t-\Delta t); \quad \forall t \in T; \ \forall m \in M(t); \qquad \text{(SM 20)}$$

These equations enable the Kalman filter to optimally combine the prediction and measurement, accounting for uncertainties, to produce the best possible estimate of the object's current state.

Adaptive Noise Tuning

To enhance the Kalman filter's performance in dynamic environments, adaptive noise tuning is employed. This involves dynamically adjusting the process noise covariance matrix $Q_m(t)$ and measurement noise covariance matrix $R_m(t)$ based on the observed tracking error. This adjustment allows the Kalman filter to better handle variations in object motion and measurement uncertainties (Simon, 2006).

Specifically, the measurement residual $\varepsilon_m(t)$ is calculated to represent the difference between the observed measurement and the predicted state:

$$\varepsilon_m(t) = z_m(t) - O \cdot \hat{s}_m(t|t-\Delta t); \quad \forall t \in T; \ \forall m \in M(t); \qquad \text{(SM 21)}$$

Exponential smoothing is then applied to stabilise the impact of the noise by deriving a smoothed residual:

$$smoothed\left(\varepsilon_m(t)\right) = 0.8 \cdot smoothed\left(\varepsilon_m(t-\Delta t)\right) + 0.2 \cdot \varepsilon_m(t); \quad \forall t \in T; \ \forall m \in M(t); \qquad \text{(SM 22)}$$

When the smoothed error ε_m increases, it indicates that the current estimates are becoming less reliable, suggesting a need for greater flexibility in the state estimation. Therefore, $Q_m(t)$ is updated as follows:

$$Q_m(t) = Q_m(t) + \min\left(adaptiveFactor \cdot \left(smoothed\left(\varepsilon_m(t)\right) - smoothed\left(\varepsilon_m(t-1)\right)\right); 0.1\right) \cdot I; \quad \forall t \in T; \ \forall m \in M(t); \qquad \text{(SM 23)}$$

Allowing for fluctuations in $Q_m(t)$ denotes the capability of the framework to adapt to inherent uncertainty in the data. By lowering $Q_m(t)$, the algorithm allows the Kalman filter to place more trust in the process model (model's predictions) rather than the observed measurements, which is useful when measurement data may be less reliable. The use of the minimum function ensures that the update remains bounded, preventing excessively large adjustments that could destabilise the filter.

Conversely, when the smoothed error decreases, the measurement data may be more reliable, and thus, the measurement noise covariance $R_m(t)$ should be adjusted:

$$R_m(t) = \max\big(0.1 \cdot I;\ R_m(t) - \min\big(adaptiveFactor \cdot \big|smoothed\big(\varepsilon_m(t)\big) - smoothed\big(\varepsilon_m(t-1)\big)\big|; 0.1\big) \cdot I\big);\quad \forall t \in T;\ \forall m \in M(t); \qquad \text{(SM 24)}$$

Lowering $R_m(t)$ implies an increased confidence in the measurement data. The threshold of $0.1 \cdot I$ prevents $R_m(t)$ from becoming too small, which would lead to overconfidence in the measurements and potentially destabilise the estimates. Next, adaptive thresholds based on the smoothed error provide additional mechanisms for adjusting the covariance matrices. Particularly, the threshold for increasing $Q_m(t)$ is defined as:

$$\tau_1 = 0.1 \cdot maxExpectedDeviation; \qquad \text{(SM 25)}$$

This threshold ensures that a significant increase in process noise covariance occurs only when the smoothed error exceeds 10% of the maximum expected deviation, allowing the filter to adapt to large deviations without being overly sensitive to minor fluctuations (Simon, 2006). The condition for adjusting the process noise covariance matrix $Q_m(t)$ can expressed mathematically as:

$$\text{if } smoothed\big(\varepsilon_m(t)\big) > \tau_1, \quad \text{then } Q_m(t) = Q_m(t) + (2 * adaptiveFactor, 0.1) \cdot I;$$
$$\forall t \in T;\ \forall m \in M(t); \qquad \text{(SM 26)}$$

This adjustment reflects a substantial increase in model uncertainty when the smoothed error surpasses the defined threshold τ_1. Such a scenario necessitates that the filter adaptively handles greater deviations between predicted and actual measurements, thus enhancing the robustness of the state estimates. Similarly, the threshold for decreasing $Q_m(t)$ is defined as:

$$\tau_2 = 0.05 \cdot maxExpectedDeviation; \qquad \text{(SM 27)}$$

This threshold ensures that the process noise covariance is reduced when the smoothed error falls below 5% of the maximum expected deviation, promoting filter stability and convergence during low-error conditions (Simon, 2006). The condition for adjusting the process noise covariance matrix $Q_m(t)$ can expressed mathematically as:

$$\text{elseif } smoothed\big(\varepsilon_m(t)\big) < \tau_2, \text{ then } Q_m(t) = \max(Q_m(0);\ Q_m(t) - \min(0.5 \cdot adaptiveFactor; 0.1) \cdot I);\quad \forall t \in T;\ \forall m \in M(t); \qquad \text{(SM 28)}$$

Reducing $Q_m(t)$ when the smoothed error is below τ_2 signals increased confidence in the model predictions. By stabilizing the process noise covariance, the filter promotes convergence of the estimates to the actual values, minimizing excessive deviations.

4.Average Features for Tracked Objects (Step 2.3)

This section provides details about Step 2.3, which involves averaging the feature vectors (generated by ResNet) and the embeddings (generated by Re-ID) of the tracked objects over time.

Let:

- $r_m(t)$ be the feature vector for tracked object $m \in M(t)$ at time $t \in T$. It is computed as in Step 2.1, where n is replaced with m.
- $e_m(t)$ be the embedding vector for re-identifying the tracked object $m \in M(t)$ at time $t \in T$. It is computed as in Step 2.1, where n is replaced with m.
- $\bar{r}_m(t)$ be the average feature vector for tracked object $m \in M(t)$ at time $t \in T$.
- $\bar{e}_m(t)$ be the average embedding vector for re-identifying the tracked object $m \in M(t)$ at time $t \in T$.

- $matchCount_m(t)$ be the number of consecutive frames in which object $m \in M(t)$ has been successfully matched at time $t \in T$. This count increases with each matched frame, allowing for more reliable tracking over time.
- $baseHistLength$ be the initial minimum history length (expressed as a number of frames or, equivalently, as a number of timestamps) used for the averaging process.
- $maxHistLength_m(t)$ be the dynamic history length (expressed as a number of frames or, equivalently, as a number of timestamps) for the tracked object $m \in M(t)$ at time $t \in T$, used for the averaging process. It grows with $matchCount_m(t)$.
- $upperBound$ be the be the maximum allowable history length (expressed as a number of frames or, equivalently, as a number of timestamps) used for the averaging process.

The dynamic history length $maxHistLength_m(t)$ is computed as follows:

$$maxHistLength_m(t) = \min(baseHistLength + matchCount_m(t); upperBound);$$
$$\forall t \in T;\ \ \forall m \in M(t); \text{ (SM 29)}$$

Eqn. (SM 29) ensures that the history length starts from a base value and grows as the object's $matchCount_m(t)$ increases, but it is capped by the upper bound to prevent excessive growth.

Therefore, the average feature and embedding vectors for tracked object $m \in M(t)$ at time $t \in T$, denoted as $\bar{r}_m(t)$ and $\bar{e}_m(t)$, respectively, are computed by averaging the features over the most recent $maxHistLength_m(t)$ frames:

$$\bar{r}_m(t) = \frac{1}{maxHistLength_m(t)} \sum_{t'=t-leng\_aver+}^{t} r_m(t') \qquad \text{(SM 30)}$$

$$\bar{e}_m(t) = \frac{1}{maxHistLengt\ _m(t)} \sum_{t'=t-le\ \ \_aver+}^{t} e_m(t') \qquad \text{(SM 31)}$$

These averaged quantities smooth variations in object appearance and help improve matching, especially during occlusions or erratic movements.

5. Apply Savitzky-Golay Filter for Unmatched Objects (Step 2.6)

This section provides details about Step 2.6, which involves applying the Savitzky-Golay filter to smooth the trajectory of tracked pedestrians and vehicles that have temporarily gone unmatched.

Let :

- $LengthSavi$ be the history length (expressed in number of frames or, equivalently, in number of timestamps) used for the Savitzky-Golay filtering process.
- ϕ be the Savitzky-Golay filter coefficient matrix, where ϕ_{ij} represents the coefficient for the j-th element in the i-th row of the matrix. The middle row of the matrix serves as the primary smoothing filter applied to the signal during steady-state conditions. The rows before and after the middle row act as transitional filters to handle edge effects, accommodating the beginning and end of the data series.
- $\check{p}_{world,m}(t)$ be the smoothed position for the centroid $\hat{p}_{world,m}(t)$ of the unmatched tracked object $m \in M(t)$ at time $t \in T$.

The smoothed centroid position $\check{p}_{world,m}$ is calculated by applying the filter over a window of the past $LengthSavi$ frames:

$$\check{p}_{world,m}(t) = \sum_{t'=t-LengthSavi+1}^{t} \phi_{(\max(1,\min(LengthSav,t'-(t-LengthSavi)+1))),(t-t'+1)} \cdot \hat{p}_{world,m}(t');$$

$$\forall t \in T;\ \forall m \in M(t) |\ m = unmatched; \qquad \text{(SM 32)}$$

Eqn. (SM 32) ensures that:

1) For the early part of the history length, the first rows (handling start transients) of ϕ are used.
2) For the middle part of the history length, the centre row of ϕ (handling steady-state filtering) is used.
3) For the end part of the history length, the last rows of ϕ (handling end transients) are used.
4) The column index corresponds to the position difference between t' and the current time t.

The coefficients ϕ_{ij} used in the Savitzky-Golay filter are derived from fitting a polynomial of a specified order to the local window of data. Specifically, they represent the weights applied to each centroid position $\hat{p}_{world,m}(t')$ in the window defined by $LengthSavi$. These coefficients are computed based on the least-squares polynomial fitting, ensuring that the filtered output $\check{p}_{world,m}(t)$ preserves trends while smoothing short-term noise (MathWorks, 2024b).

6.Count Pedestrians and Vehicles (Step 2.10)

This section provides details about Step 2.10, which involves counting pedestrians and vehicles crossing specific gates.

Counting Mechanism

Let:

- GA be the set of gates, and $ga \in GA$ be a generic gate, i.e., a line segment in the ground plane, disposed perpendicularly to the expected trajectory of the object to be counted.
- $P_{1,ga} = (x_1, y_1)$ and $P_{2,ga} = (x_2, y_2)$ be the two endpoints of gate $ga \in GA$.
- $projection_{ga,m}(t)$ be the projection of the object's centroid $p_{world,m}$ onto the gate $ga \in GA$.
- $d_{ga,m}(t)$ be the perpendicular distance between the gate line $ga \in GA$ and the projected ground positions $p_{world,m}$ of the tracked object $m \in M(t)$ at time $t \in T$.
- $d_{threshold}$ be the threshold for the perpendicular distance $d_{ga,m}(t)$.
- c_min be the minimum confidence threshold used for the counting.
- $N_{legal,ga}(t)$, $N_{illegal,ga}(t)$ and $N_{unkown,ga}(t)$ be the cumulative number of legal, illegal and unknown crossings at the gate $ga \in GA$ at time $t \in T$.

For each gate $ga \in GA$ and tracked objected $m \in M(t)$ with centroid $p_{world,m}$, a gate crossing event is detected by considering two key factors: projection test and perpendicular distance test.

As for the projection test, the position vector from the object centroid to the start of the gate (i.e., $p_{world,m} - P_{1,ga}$) is projected onto the gate vector itself (i.e., $P_{2,ga} - P_{1,ga}$). This projection, denoted as $projection_{ga,m}(t)$, determines whether the object $m \in M(t)$ is within the boundaries defined by the gate's endpoints. It is computed as follows:

$$projection_{ga,m}(t) = \frac{(p_{world,m} - P_{1,ga}) \cdot (P_{2,ga} - P_{1,ga})}{\|P_{2,ga} - P_{1,ga}\|};\ \forall t \in T;\ \forall (ga, m) \in GA \times M(t); \qquad \text{(SM 33)}$$

The object must satisfy the following condition to be within the gate bounds:

$$projection_{ga,m}(t) \geq 0;\ \forall t \in T;\ \forall (ga, m) \in GA \times M(t); \qquad \text{(SM 34)}$$

As for the perpendicular distance test, the distance $d_{ga,m}(t)$ between the object's centroid $p_{world,m}$ and the gate line is computed as follows:

$$d_{ga,m}(t) = \frac{\left|\frac{(P_{2,ga}-P_{1,ga})^{\perp}}{\|P_{2,ga}-P_{1,ga}\|}\cdot(p_{world,m}-P_{1,ga})\right|}{\|P_{2,ga}-P_{1,ga}\|};\ \forall t \in T;\ \forall (ga,m) \in GA\ \times M(t); \qquad \text{(SM 35)}$$

The perpendicular distance must fall below a predefined threshold for the object to be considered close enough to cross the gate:

$$d_{ga,m}(t) \le d_{threshold};\ \forall t \in T;\ \forall (ga,m) \in GA\ \times M(t); \qquad \text{(SM 36)}$$

If both the conditions in Eqns. (SM 34) and (SM 36) are satisfied, the object $m \in M(t)$ is considered to have crossed the gate $ga \in GA$.

Flag and Confidence-Based Overcounting Prevention

To prevent overcounting, gate-crossing flags are assigned to each tracked pedestrian and vehicle. These flags are stored in the respective history structures and indicate whether the object has already crossed a particular gate. Once a flag is set, the object will not be counted again at the same gate. More formally, let $flag_{ga,m}$ be the flag related to the object $m \in M(t)$ and gate $ga \in GA$. The flag takes the value $true$ if the object has already crossed the gate and $false$ otherwise.

Furthermore, a filter based on the confidence score of each tracked object is applied to avoid counting for low-reliability detections. More formally, the objects with a confidence lower than c_min are excluded by the counting.

Managing the Count

At each time $t \in T$, for each gate $ga \in GA$ and object $m \in M(t)$, a straightforward logic is used to compute the cumulative number of legal, illegal, and unknown crossings. More formally, if the object $m \in M(t)$ has crossed the gate $ga \in GA$, $flag_{ga,m} = false$, and $c_m > c\_min$, the respective count is updated based on the object status $S_m(t)$:

$$\text{if } \left(projection_{g,m}(t) \ge 0\right) \text{ and } \left(d_{ga,m}(t) \le d_{threshold}\right) \text{ and } \left(flag_{ga,m} = false\right) \text{ and } (c_m > c\_min) \text{ and } (S_m(t) = legal), \quad \text{then: } N_{legal,ga}(t) = N_{legal,ga}(t-\Delta t) + 1;$$

$$\forall t \in T;\ \forall (ga,m) \in GA\ \times M(t); \text{ (SM 37)}$$

$$\text{if } \left(projection_{ga,m}(t) \ge 0\right) \text{ and } \left(d_{ga,m}(t) \le d_{threshold}\right) \text{ and } \left(flag_{ga,m} = false\right) \text{ and } (c_m > c\_min) \text{ and } (S_m(t) = illegal), \quad \text{then: } N_{illegal,ga}(t) = N_{illegal,ga}(t-\Delta t) + 1;$$

$$\forall t \in T;\ \forall (ga,m) \in GA\ \times M(t); \text{ (SM 38)}$$

$$\text{if } \left(projection_{ga,m}(t) \ge 0\right) \text{ and } \left(d_{ga,m}(t) \le d_{threshold}\right) \text{ and } \left(flag_{ga,m} = false\right) \text{ and } (c_m > c\_min) \text{ and } (S_m(t) = unknown), \text{ then: } N_{unknown,ga}(t) = N_{unknown,ga}(t-\Delta t) + 1;$$

$$\forall t \in T;\ \forall (ga,m) \in GA\ \times M(t); \text{ (SM 39)}$$

7. Calculate Velocity and Speed (Step 2.11)

This section provides more details about Step 2.11, which involves calculating the velocity and the speed of tracked objects and removing the outlier speeds.

Velocity Calculation

For each tracked object $m \in M(t)$, the velocity vector is calculated based on the change in position over time.

More formally, let $v_m(t)$ be the velocity vector of object $m \in M(t)$ at time $t \in T$. It is calculated as follows:

$$v_m(t) = \frac{p_{world,m}(t) - p_{world,m}(t-1)}{\Delta t};\quad \forall t \in T;\ \forall m \in M(t); \quad \text{(SM 40)}$$

Speed Calculation

For each tracked object $m \in M(t)$, the speed is calculated as the magnitude of the velocity vector $v_m(t)$.

More formally, let $speed_m(t)$ be the speed of object $m \in M(t)$ at time $t \in T$. It is calculated as follows:

$$speed_m(t) = \|v_m(t)\| = \sqrt{\left[v_{x_m}(t)\right]^2 + \left[v_{y_m}(t)\right]^2}; \quad \forall t \in T;\ \forall m \in M(t); \qquad \text{(SM 41)}$$

Outlier Removal Using Median Absolute Deviation (MAD) and maximum plausible speed

To ensure accuracy, outliers in speed calculations are removed using the Median Absolute Deviation method and considering a maximum plausible speed (Iglewicz & Hoaglin, 1993; Leys et al., 2013).

More formally, let be:

- n_frames be the number of frames considered for computing the MAD.
- $median\big(speed_m(t)\big)$ be the median speed of object $m \in M(t)$ over the most recent n_frames.
- $MAD\big(speed_m(t)\big)$ be the median absolute deviation of the speed of object $m \in M(t)$ over the most recent n_frames.
- k be a constant threshold.
- $speed_{max,ped}$ and $speed_{max,veh}$ be the maximum plausible speed for pedestrians and vehicles, respectively.

Speed values satisfying at least one of the following conditions are considered outliers and are discarded:

$$\left|speed_m(t) - median\big(s_m(t)\big)\right| > k \cdot MAD\big(speed_m(t)\big); \quad \forall t \in T;\ \forall m \in M(t); \quad \text{(SM 42)}$$

$$speed_m(t) > speed_{max,ped}; \quad \text{if } m = ped \in PED(t); \quad \forall t \in T;\ \forall m \in M(t); \qquad \text{(SM 43)}$$

$$speed_m(t) > speed_{max,veh}; \quad \text{if } m = veh \in VEH(t); \qquad \forall t \in T;\ \forall m \in M(t); \quad \text{(SM 44)}$$

8. Calculate Time-to-Collision (Step 2.12)

This section provides more details about Step 2.12, which involves calculating the TTC between pedestrian and vehicle pairs.

Let:

- $d_{rel,ped,veh}(t)$ be the relative position vector for the pair $ped \in PED_t$ and $veh \in VEH_t$ at time $t \in T$.
- $v_{rel,ped,veh}(t)$ be the relative velocity vector for the pair $ped \in PED_t$ and $veh \in VEH_t$ at time $t \in T$.
- $speed_{rel\_proj,ped,veh}(t)$ be the relative speed projection, i.e., the component of the relative velocity in the direction of the relative position.
- $TTC_{ped,veh}(t)$ be the Time-to-Collision (TTC) for the pair $ped \in PED_t$ and $veh \in VEH_t$ at time $t \in T$.

For each pair $ped \in PED(t)$ and $veh \in VEH(t)$, the relative position, velocity and speed projection are computed as follows:

$$d_{rel,ped,veh}(t) = p_{world,ped}(t) - p_{world,veh}(t);$$
$$\forall t \in T;\ \forall (ped, veh) \in PED(t) \times VEH(t); \qquad \text{(SM 45)}$$
$$v_{rel,ped,veh}(t) = v_{ped}(t) - p_{veh}(t);$$

$$\forall t \in T;\ \forall (ped, veh) \in PED(t)\ \times VEH(t); \qquad \text{(SM 46)}$$

$$speed_{rel\_proj,ped,ve}\ (t) = \frac{v_{rel,ped,veh}(t) \cdot d_{rel,ped,veh}(t)}{\|d_{rel,ped,ve}\ (t)\|};$$

$$\forall t \in T;\ \forall (ped, veh) \in PED(t)\ \times VEH(t); \qquad \text{(SM 47)}$$

Hence, $TTC_{ped,veh}(t)$ is computed according to the following equation (Vogel, 2003):

$$TTC_{ped,veh}(t) = \frac{\|d_{rel,ped,veh}(t)\|}{speed_{rel_{proj},ped,veh}(t)};$$

$$\forall t \in T;\ \forall (ped, veh) \in PED(t)\ \times VEH(t); \qquad \text{(SM 48)}$$

If $speed_{rel\_proj,ped,ve}\ (t) \leq 0$, then $TTC_{ped,veh}(t) = \infty$, indicating no imminent collision.

References

Iglewicz, B., & Hoaglin, D. C. (1993). *Volume 16: how to detect and handle outliers*. Quality Press.

Leys, C., Ley, C., Klein, O., Bernard, P., & Licata, L. (2013). Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4), 764-766. https://doi.org/10.1016/j.jesp.2013.03.013

MathWorks. (2024a). *fitgeotform2d* [Documentation]. MathWorks. https://it.mathworks.com/help/images/ref/fitgeotform2d.html (Accessed on September 27, 2024)

MathWorks. (2024b). *sgolayfilt* [Documentation]. MathWorks. https://it.mathworks.com/help/signal/ref/sgolayfilt.html (Accessed on September 27, 2024)

Simon, D. (2006). *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. Wiley-Interscience. https://doi.org/10.1002/0470045345.

Vogel K. A comparison of headway and time to collision as safety indicators. Accid Anal Prev. 2003 May;35(3):427-33. doi: 10.1016/s0001-4575(02)00022-2. PMID: 12643960.