Detection and Tracking of the Vanishing Point on a Horizon for Automotive Applications

Young-Woo Seo and Ragunathan (Raj) Rajkumar

GM-CMU Autonomous Driving Collaborative Research Lab
Carnegie Mellon University

Motivation

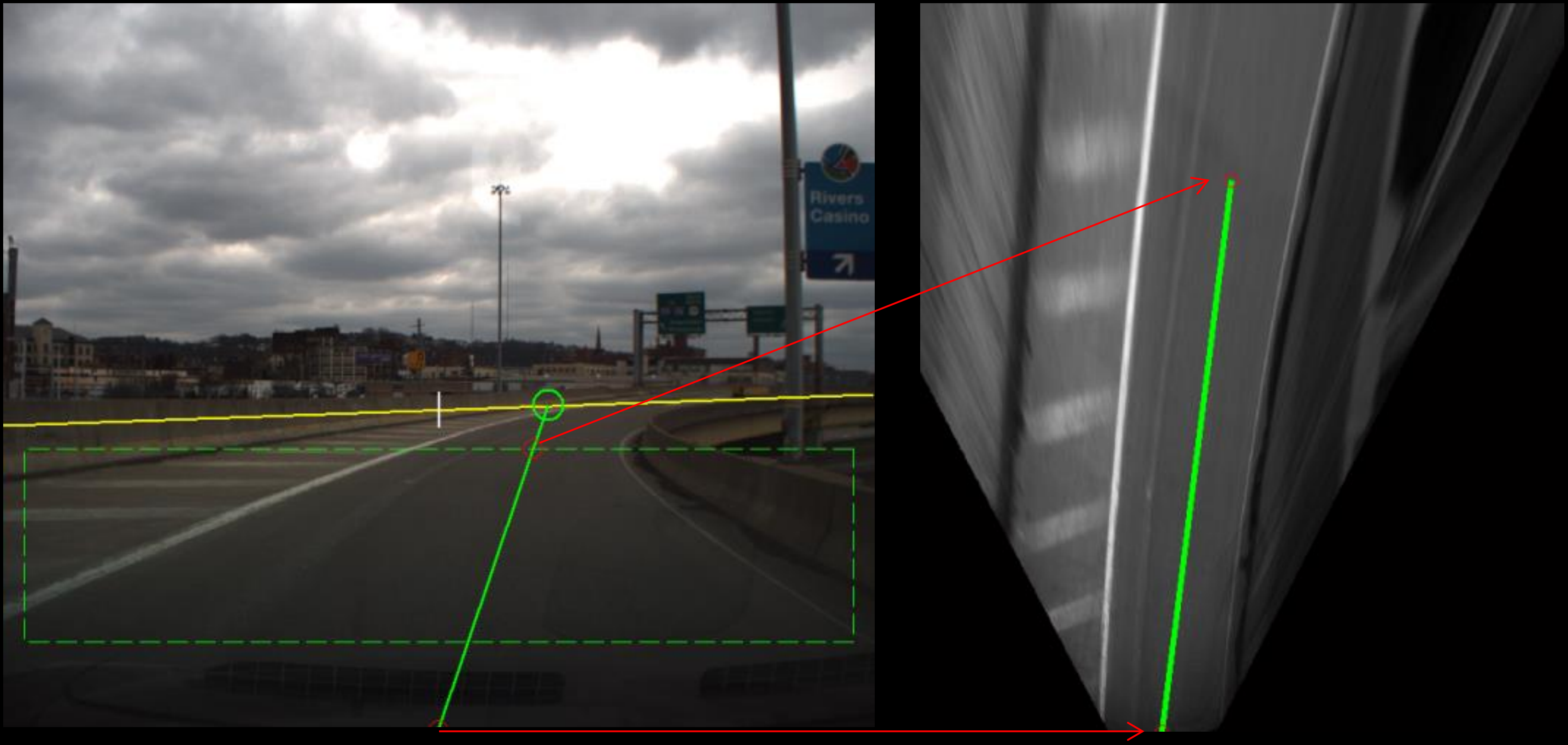
Knowledge of a horizon line and the vanishing point on the horizon line provides us with the the important information about driving environments

- Instantaneous driving direction of road
- Image sub-regions about drivable regions
- Search direction/region about road-occupants such as vehicles, pedestrians
- Geometric relation between image plane and road plane

Motivation

The location of the vanishing point on a horizon line provides important information about driving environments

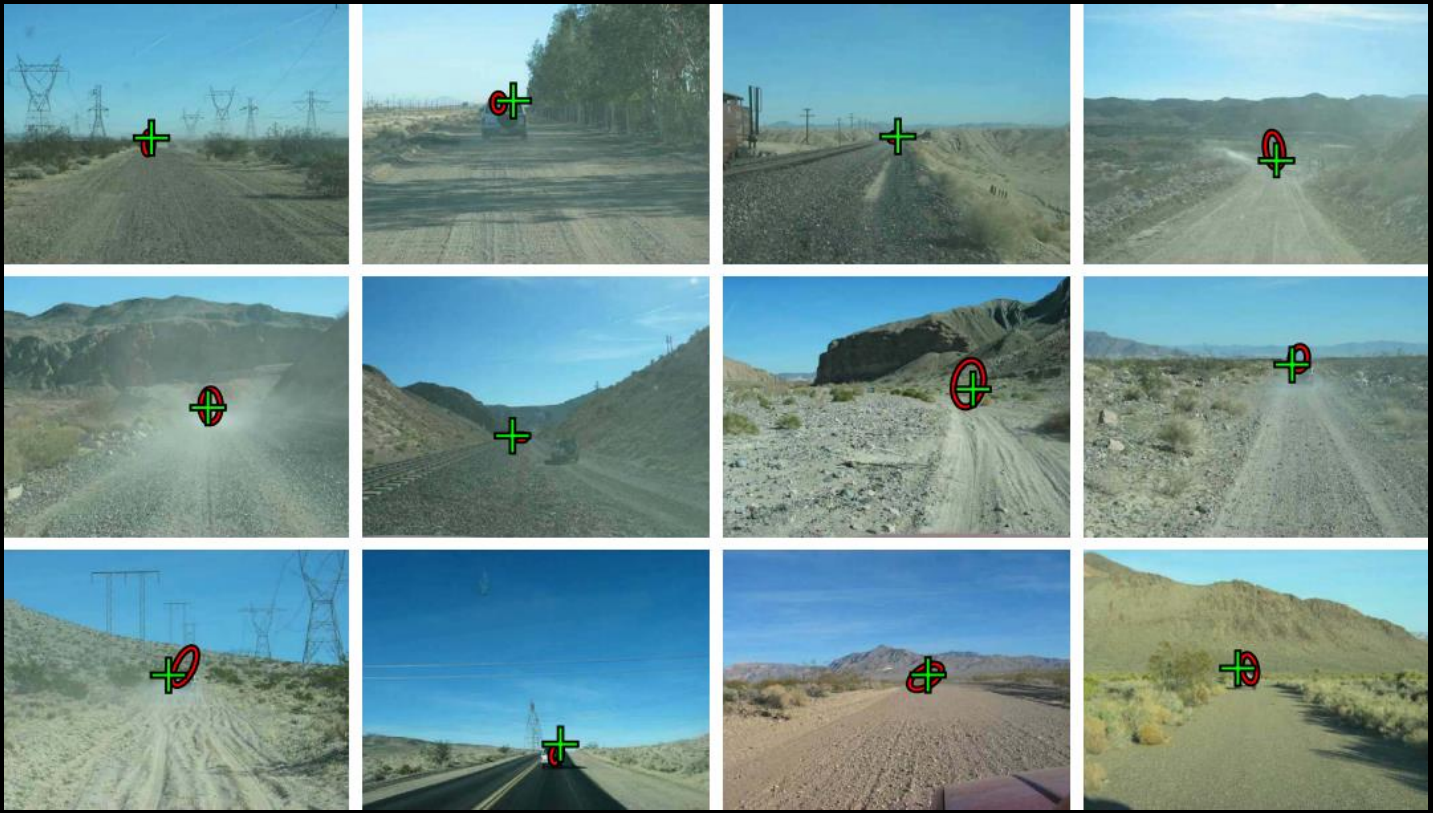
- Instantaneous driving direction of road
- Image sub-regions about drivable Regions
- Search direction of moving objects such as vehicles, pedestrians
- Geometric relation between image plane and road plane



Motivation

The location of the vanishing point on a horizon line provides important information about driving environments

- Instantaneous driving direction of road
- Image sub-regions about drivable Regions
- Search direction of moving objects such as vehicles, pedestrians
- Geometric relation between image plane and road plane

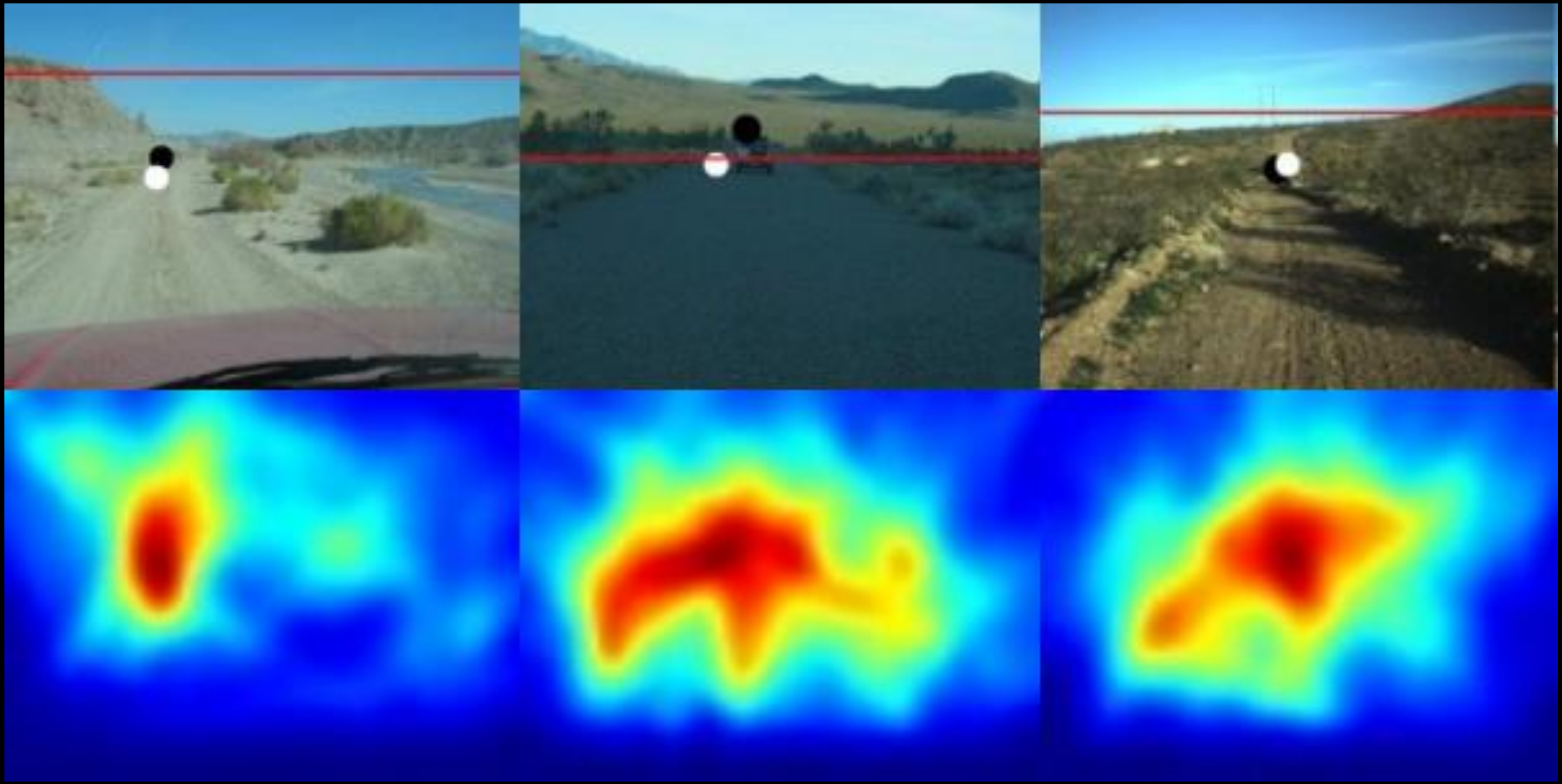


[Rasmussen, 2004] Grouping dominant orientations for ill-structured road following

Motivation

The location of the vanishing point on a horizon line provides important information about driving environments

- Instantaneous driving direction of road
- Image sub-regions about drivable Regions
- Search direction of moving objects such as vehicles, pedestrians
- Geometric relation between image plane and road plane

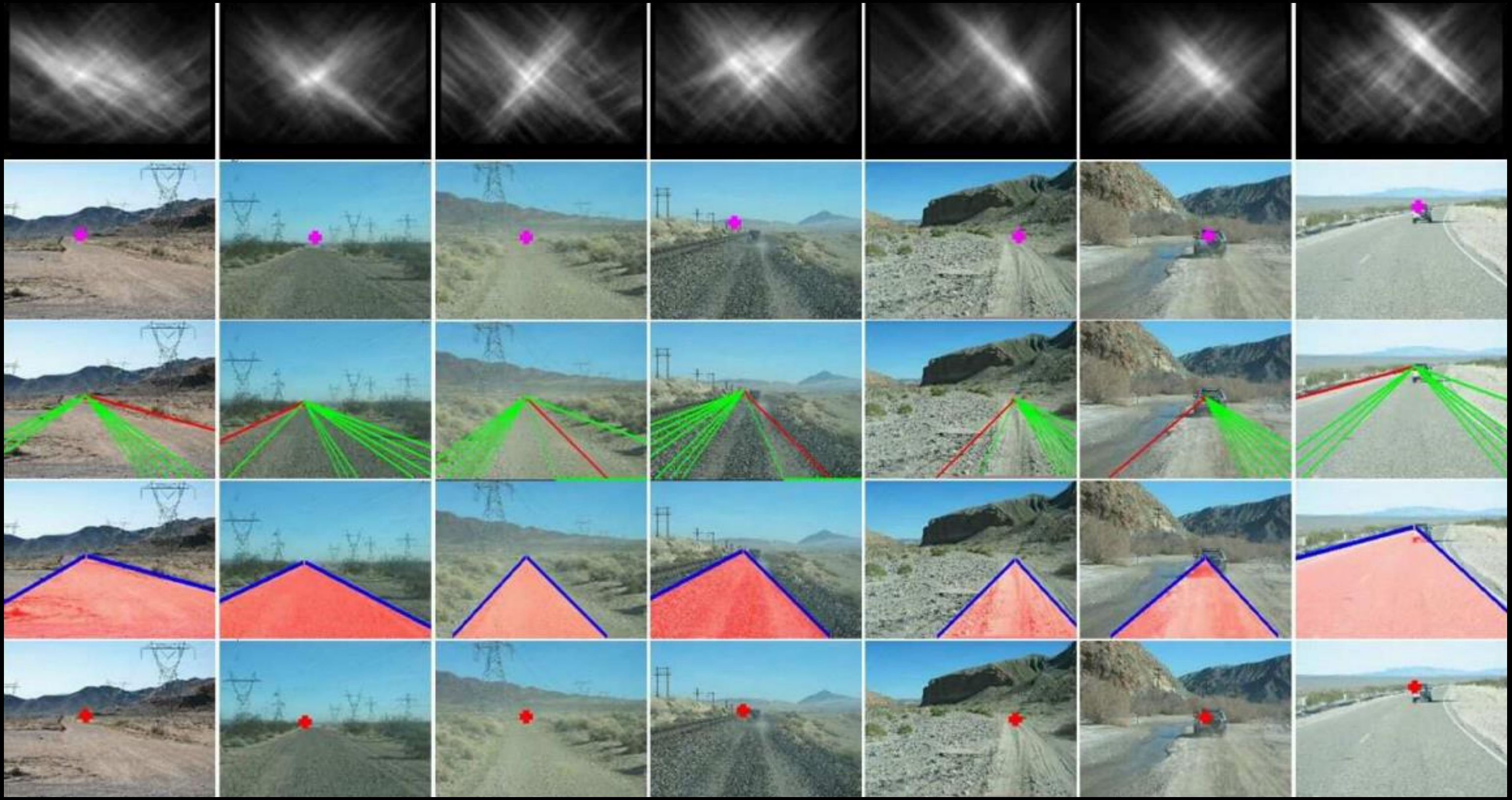


[Moghadam and Dong, 2012] Road region detection from unpaved road images

Motivation

The location of the vanishing point on a horizon line provides important information about driving environments

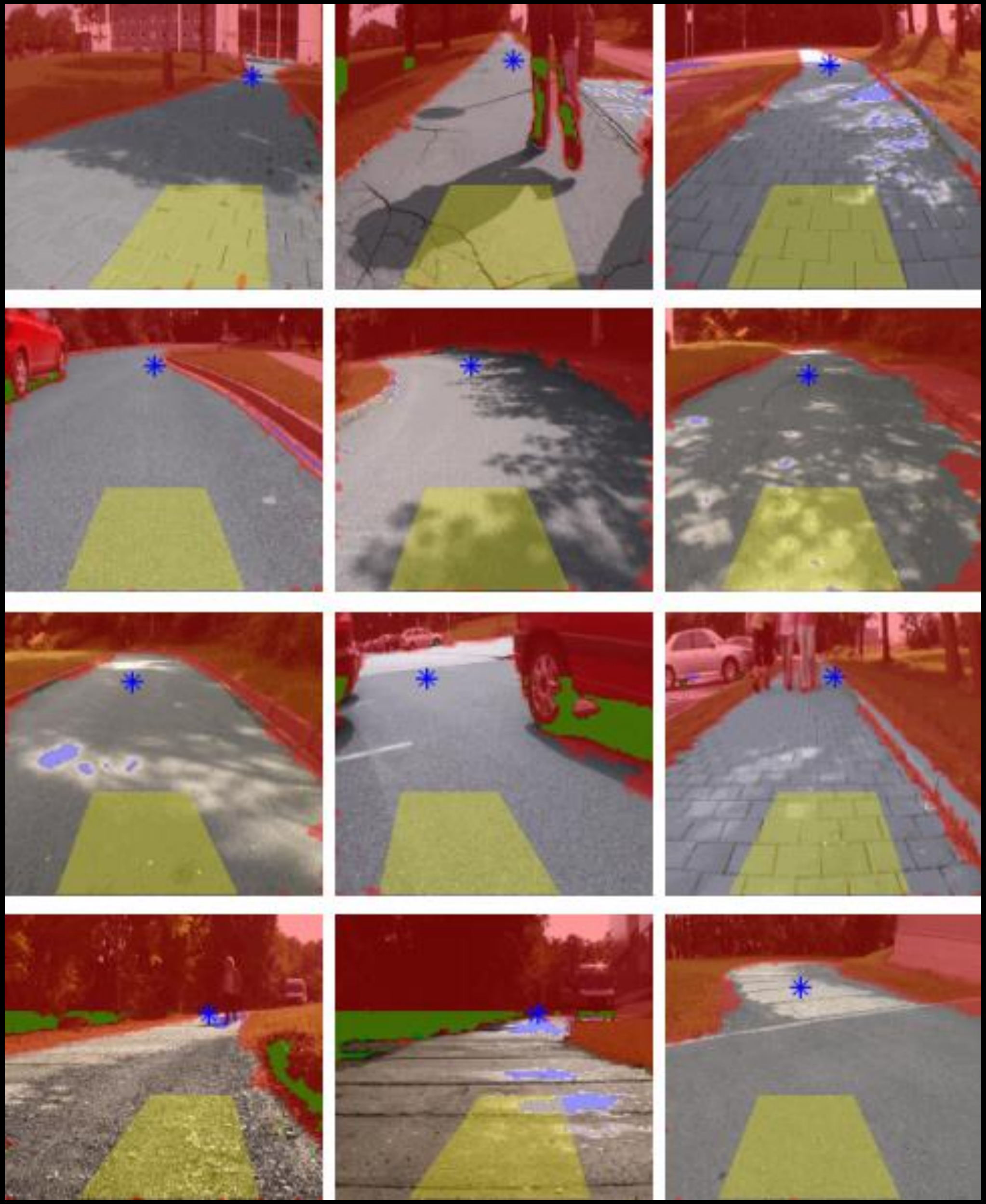
- Instantaneous driving direction of road
- Image sub-regions about drivable Regions
- Search direction of moving objects such as vehicles, pedestrians
- Geometric relation between image plane and road plane


[Kong et al., 2009] Vanishing point detection for road detection

Motivation

The location of the vanishing point on a horizon line provides important information about driving environments

- Instantaneous driving direction of road
- Image sub-regions about drivable Regions
- Search direction of moving objects such as vehicles, pedestrians
- Geometric relation between image plane and road plane

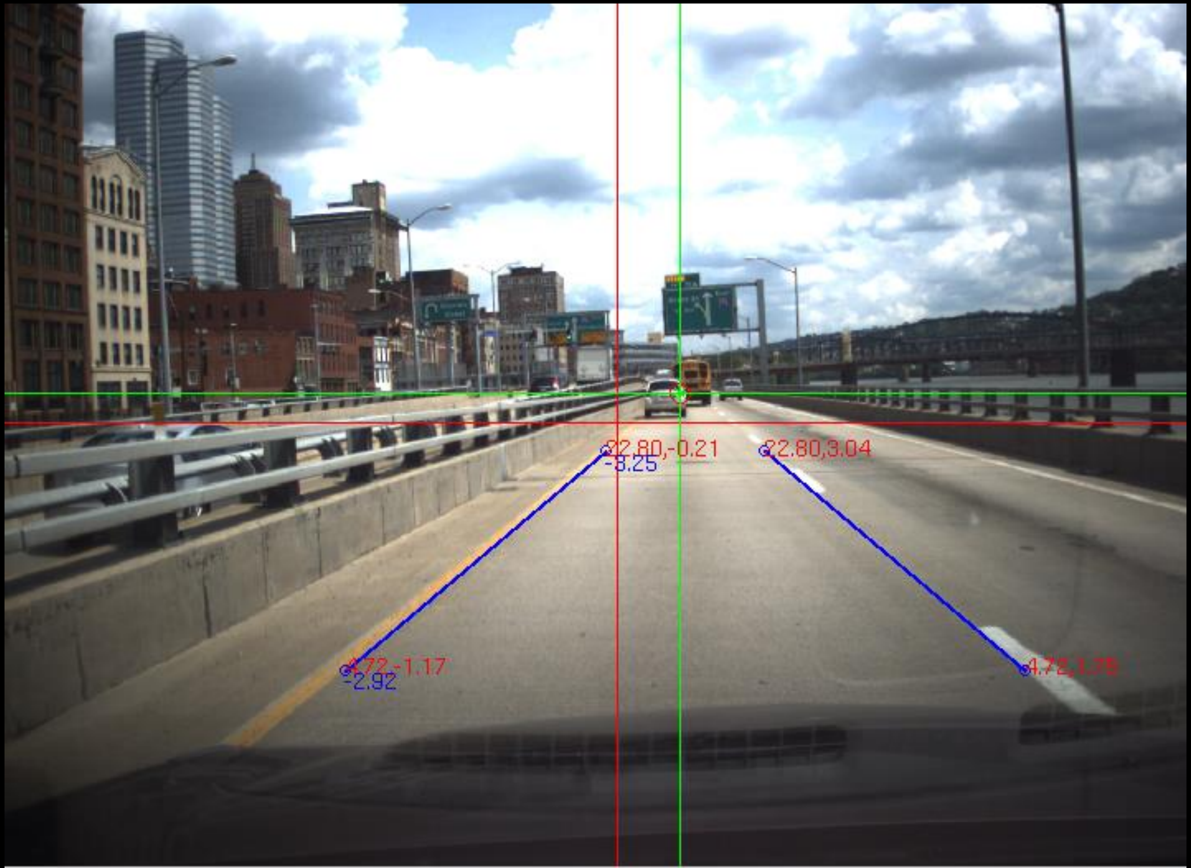


[Miksik et al., 2011] Road-detection based on vanishing point detection

Motivation

The location of the vanishing point on a horizon line provides important information about driving environments

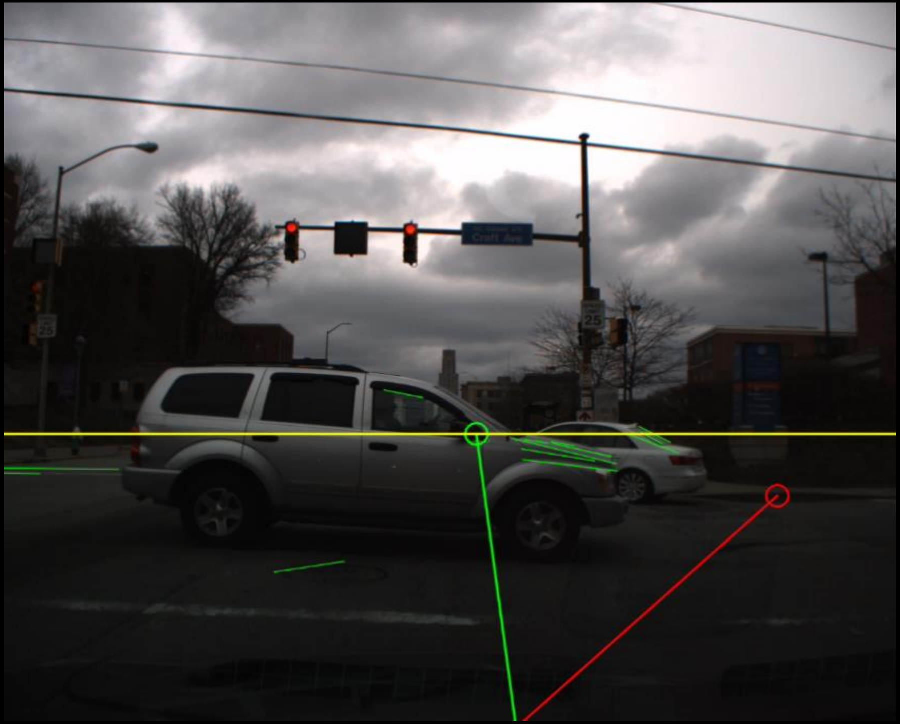
- Instantaneous driving direction of road
- Image sub-regions about drivable Regions
- Search direction of moving objects such as vehicles, pedestrians
- Geometric relation between image plane and road plane

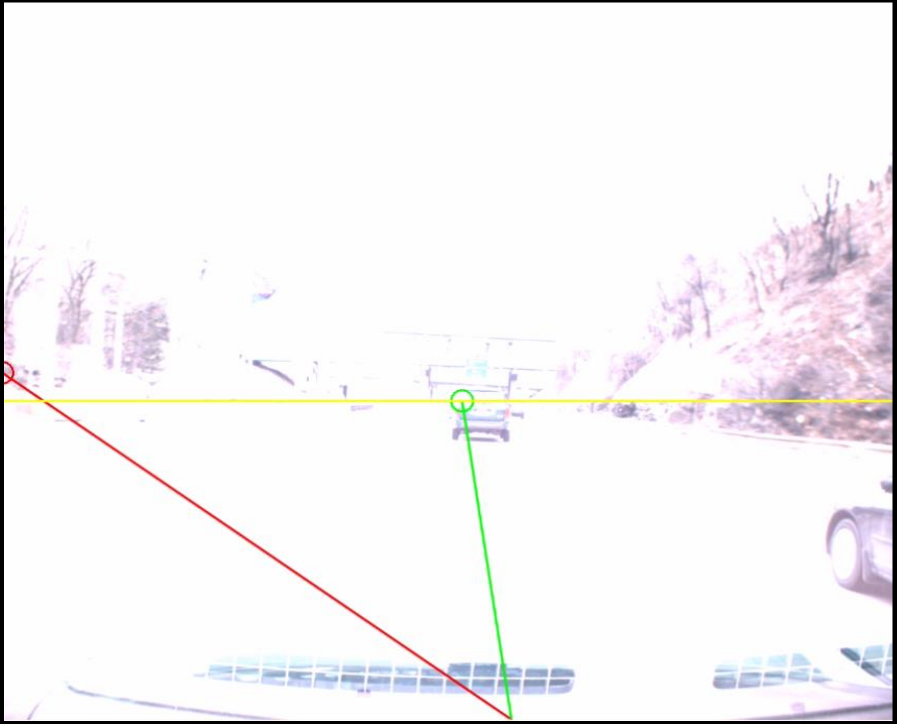


Motivation

Knowledge of a horizon line and the vanishing point on the horizon line provides us with the information about the important information about driving environments

However, the location of the vanishing point detected by frame-by-frame basis may be inconsistent over frames, due to, primarily, 1) overfitted image features and 2) absence of relevant image features





Contents

Vanishing Point Detection: Overview

Knowledge of a horizon line and the vanishing point on the horizon line provides us with the information about the important information about driving environments

Fact: Two parallel lines appearing on a perspective image meet at a point, vanishing point

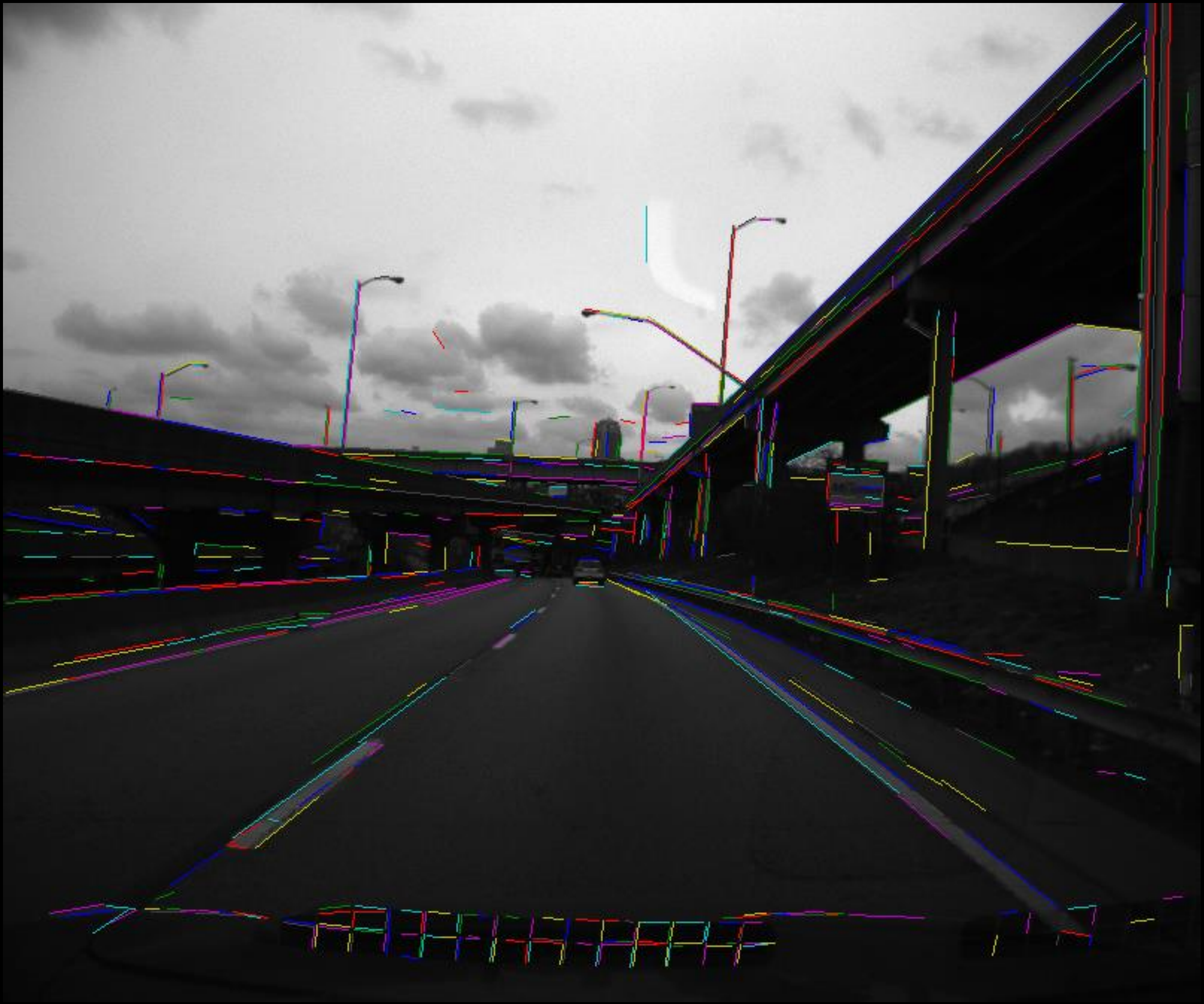
- **Line extraction**
- **Line classification** based on prior, [0, 0, 1] (horizontal), [0, 1, 0] (vertical)
- Find vanishing points through **RANSAC**
- Find **one vanishing point from vertical line class** and **more than one vanishing point from horizontal line class**

Vanishing Point Detection: Line Extraction

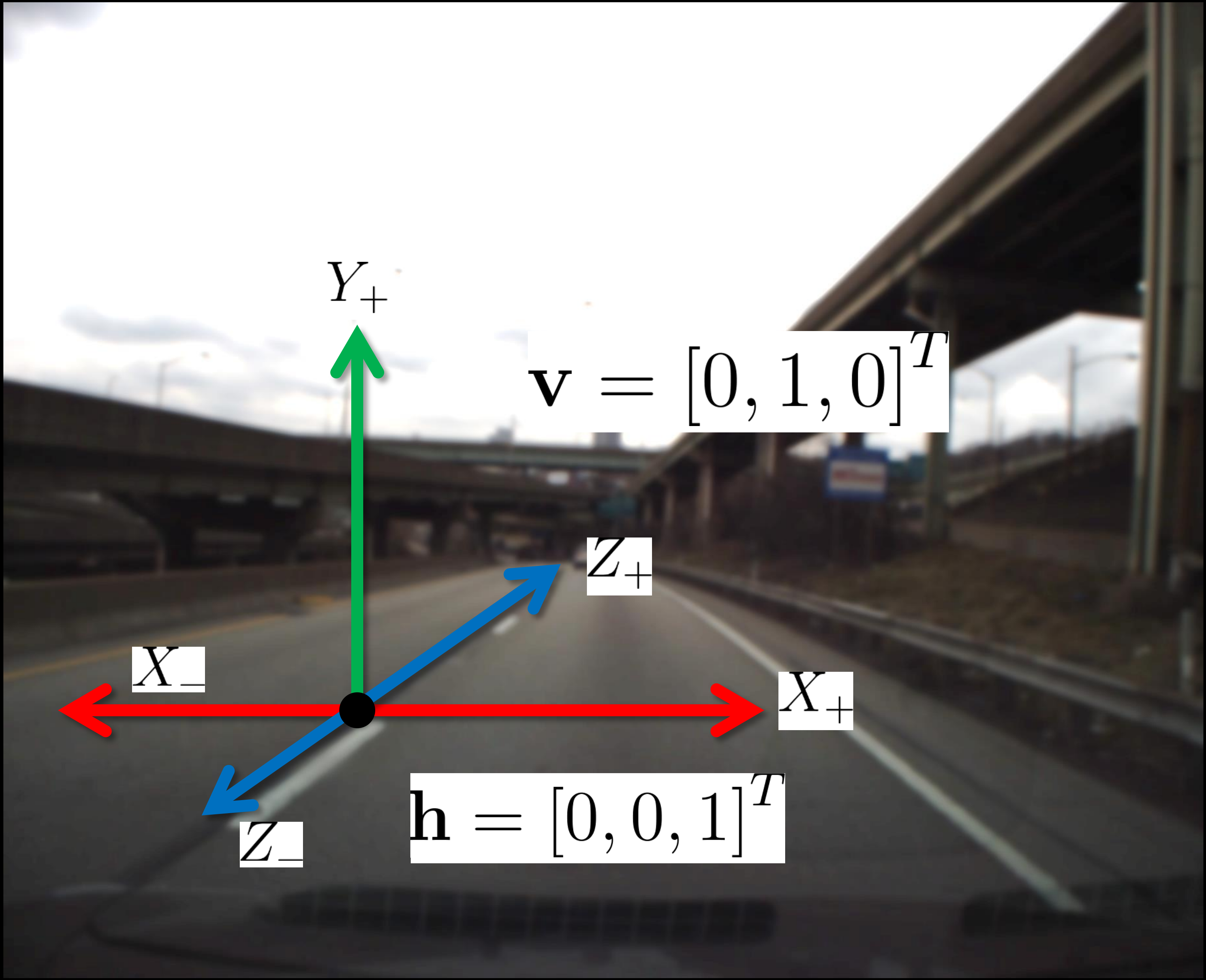
Algorithm: Line Extraction

1. Execute Histogram Equalization to normalize an input image's intensity
2. Smooth the image w/ a Gaussian kernel to suppress noises
3. Compute the gradients of the image, and magnitudes and orientations of the gradient
4. Execute a bilateral filtering to preserve natural edges
5. Compute Canny edges to collect pixel groups
6. Remove those pixel groups of which extents are too small or too large
7. Fit a line segment to each of the pixel groups

Vanishing Point Detection: Line Extraction



Vanishing Point Detection: Line Classification



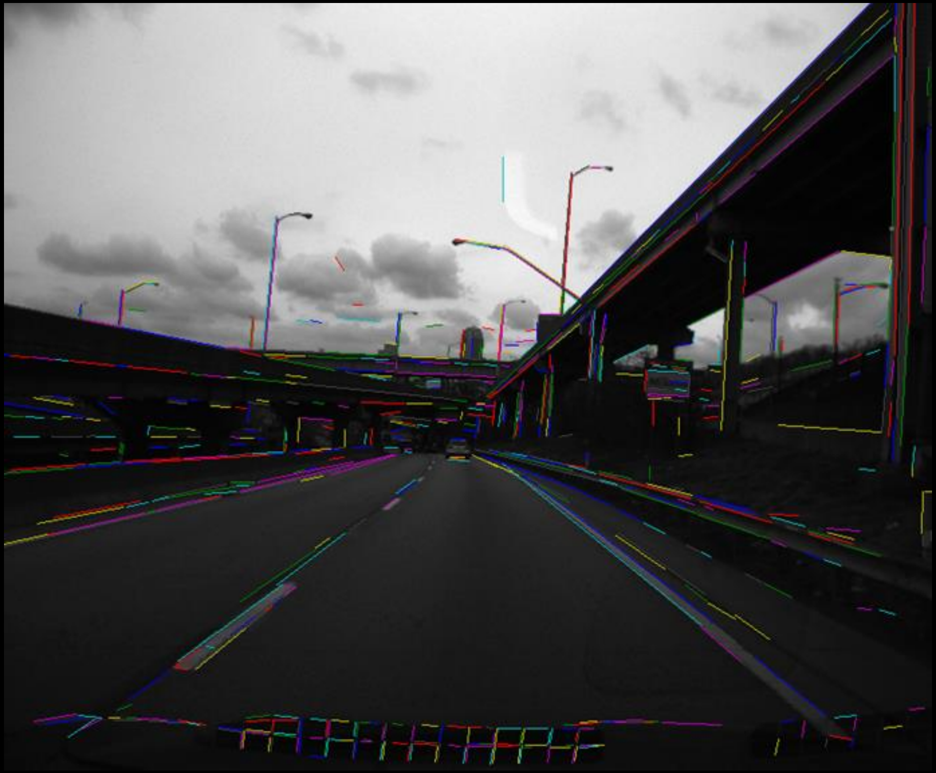
Vanishing Point Detection: Line Classification

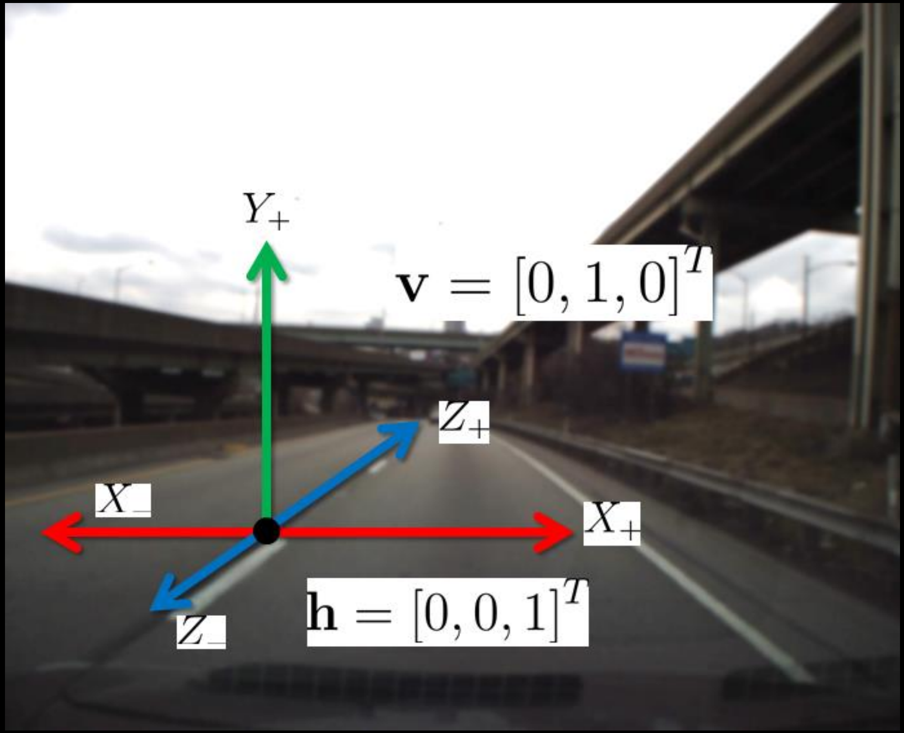
Given a line segment,

1) Compute the angle between the line and a vanishing point prior
2) Group the line into a vertical group if $\mathbf{l}_i^T \cdot \mathbf{v}_v^* < \mathbf{l}_i^T \cdot \mathbf{v}_h^*$

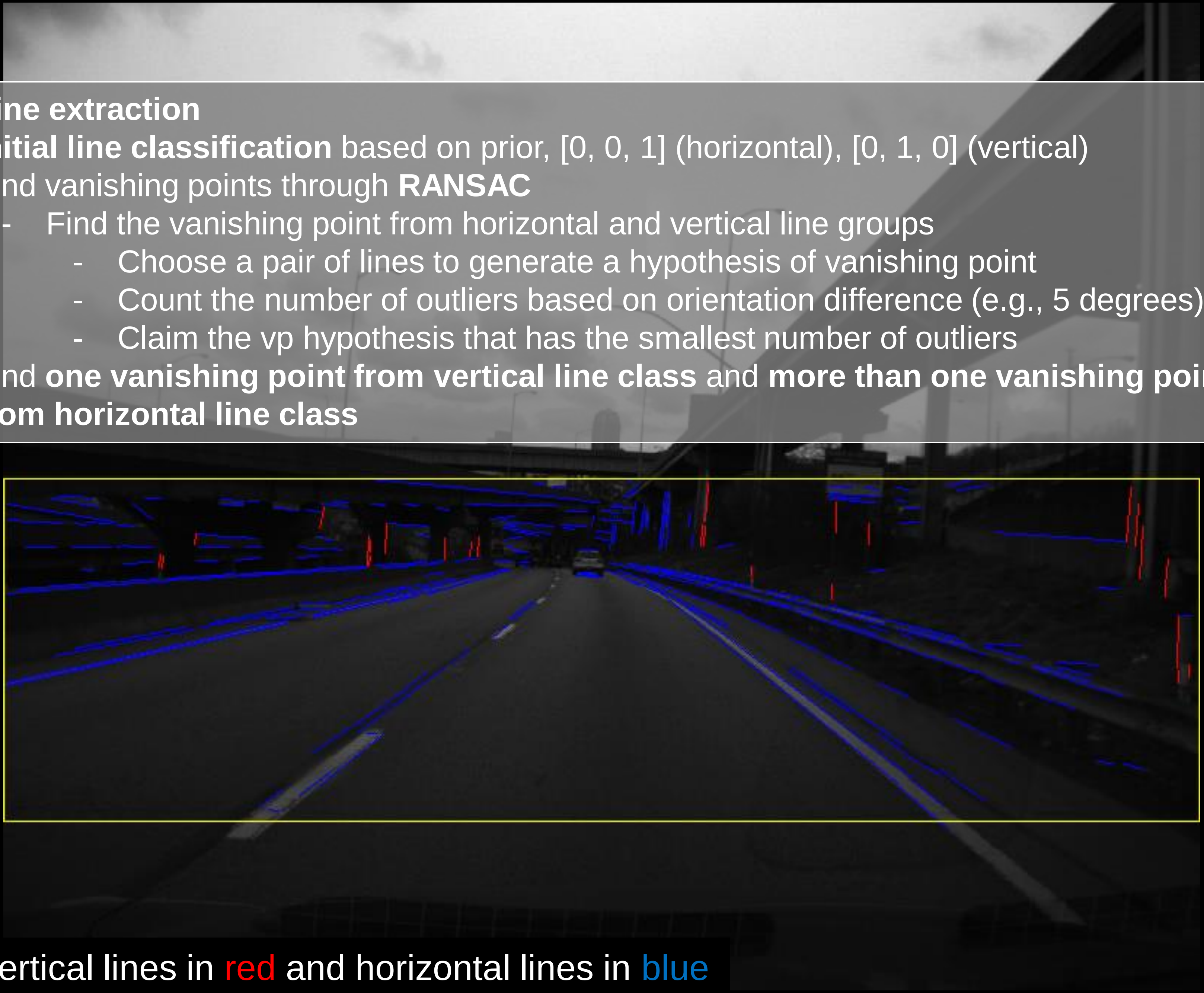
$$\mathbf{l}_i = a_i x + b_i y + c_i = \begin{bmatrix} a_i \\ b_i \\ c_i \end{bmatrix}$$

$$\mathbf{l}_i^T \cdot \mathbf{v}_j^* = \frac{[a_i, b_i, c_i]^T [v_{j,1}, v_{j,2}, v_{j,3}]}{\sqrt{a_i^2 + b_i^2 + c_i^2}}$$





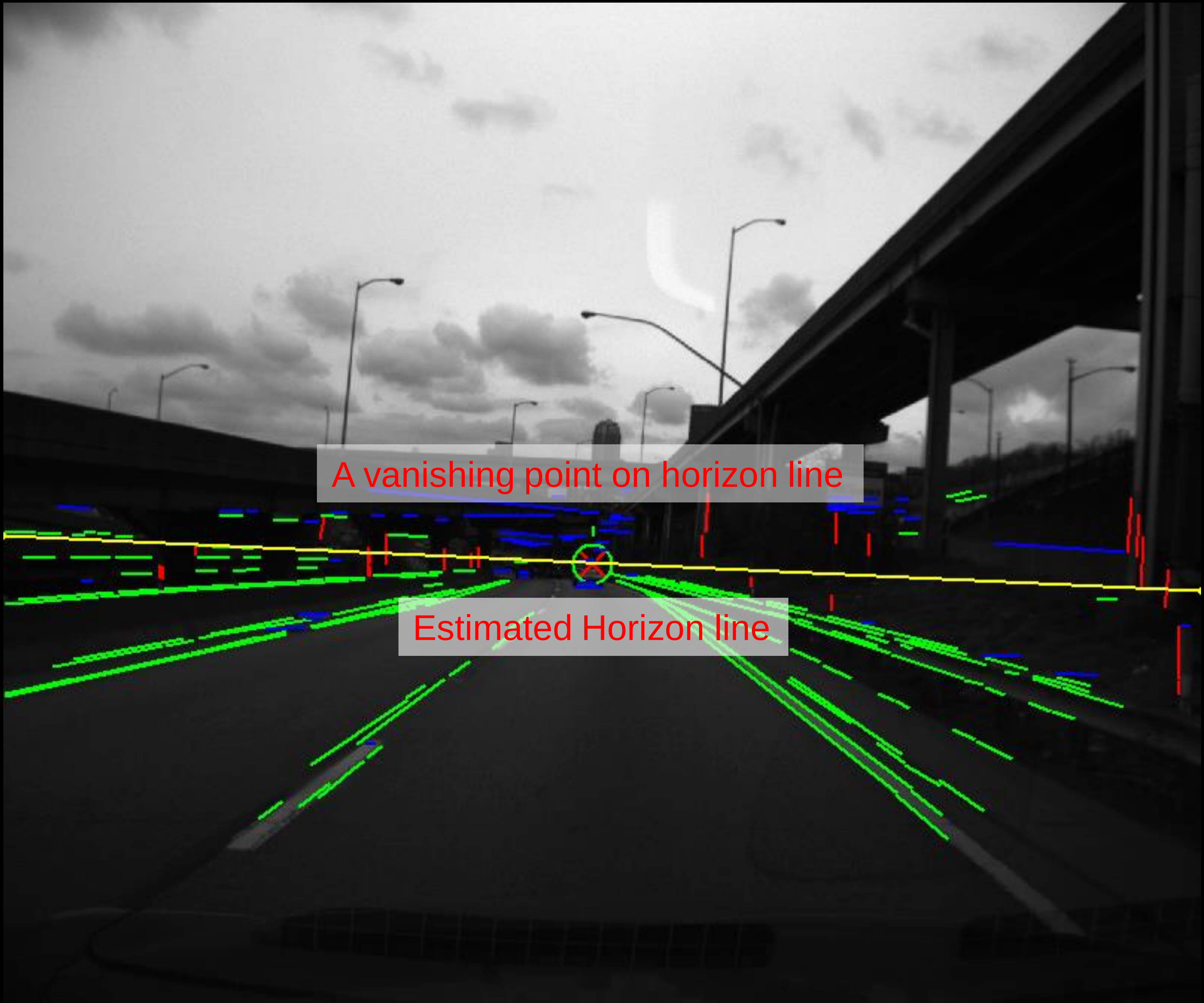
Vanishing Point Detection: Line Classification



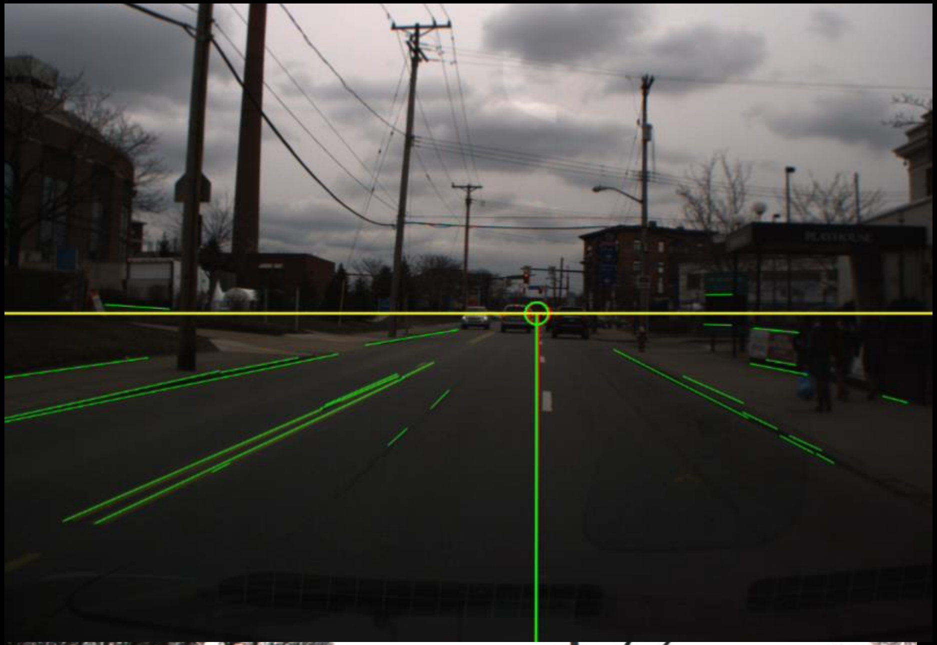
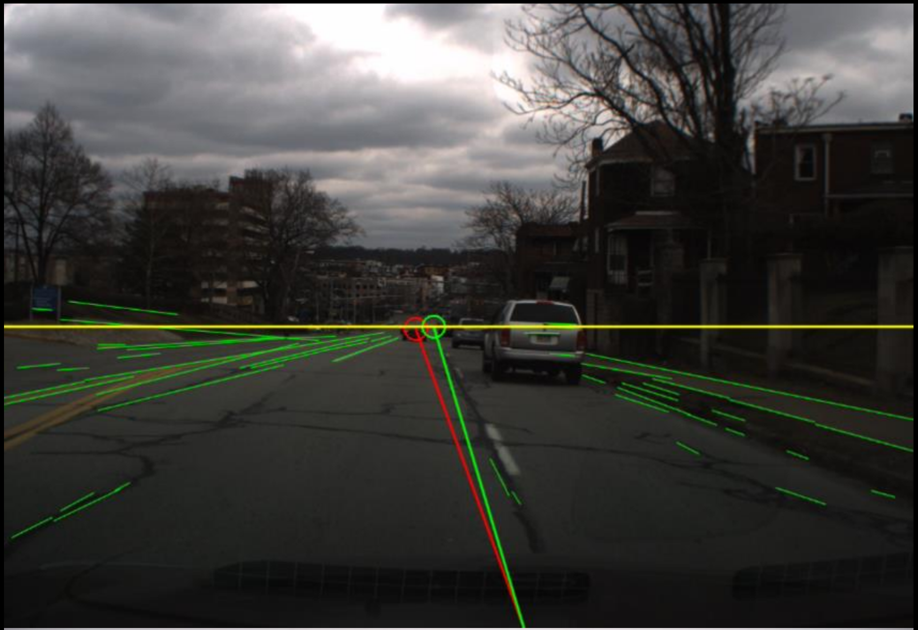
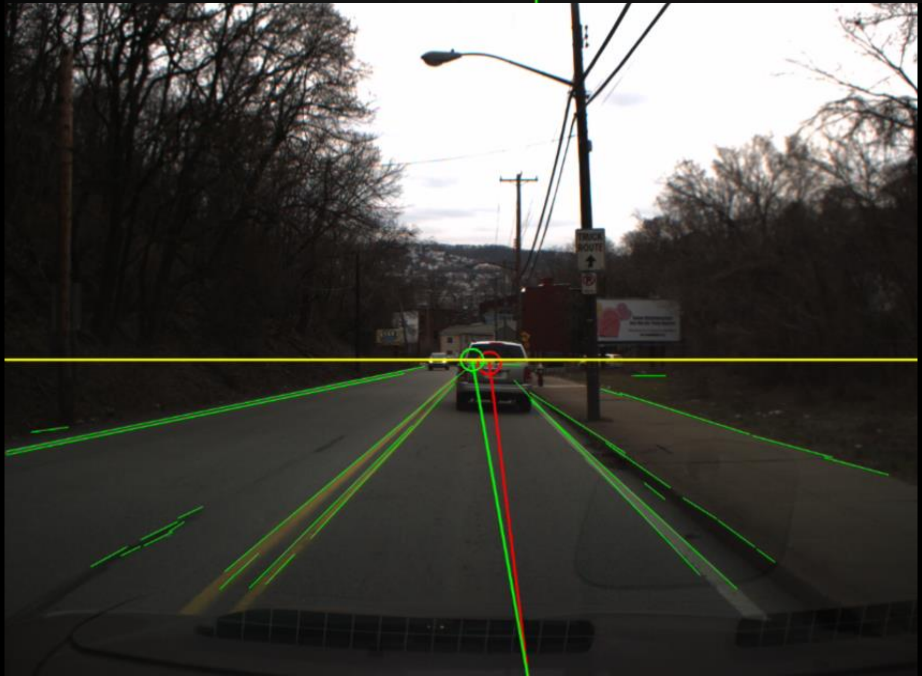
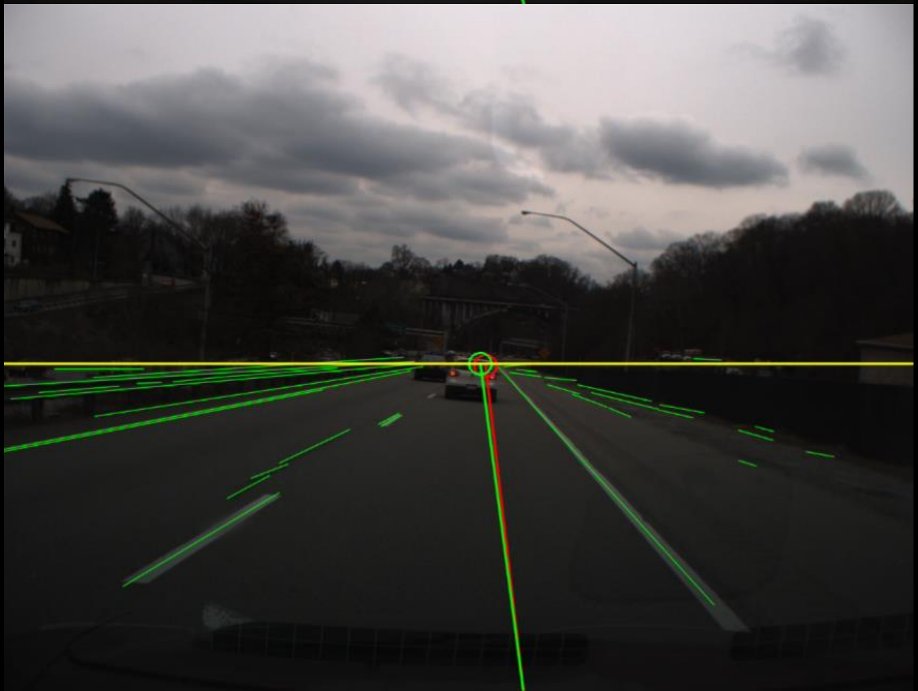
- **Line extraction**
- **Initial line classification** based on prior, [0, 0, 1] (horizontal), [0, 1, 0] (vertical)
- Find vanishing points through **RANSAC**
    - Find the vanishing point from horizontal and vertical line groups
        - Choose a pair of lines to generate a hypothesis of vanishing point
        - Count the number of outliers based on orientation difference (e.g., 5 degrees)
        - Claim the vp hypothesis that has the smallest number of outliers
- Find **one vanishing point from vertical line class** and **more than one vanishing point from horizontal line class**

Vertical lines in red and horizontal lines in blue

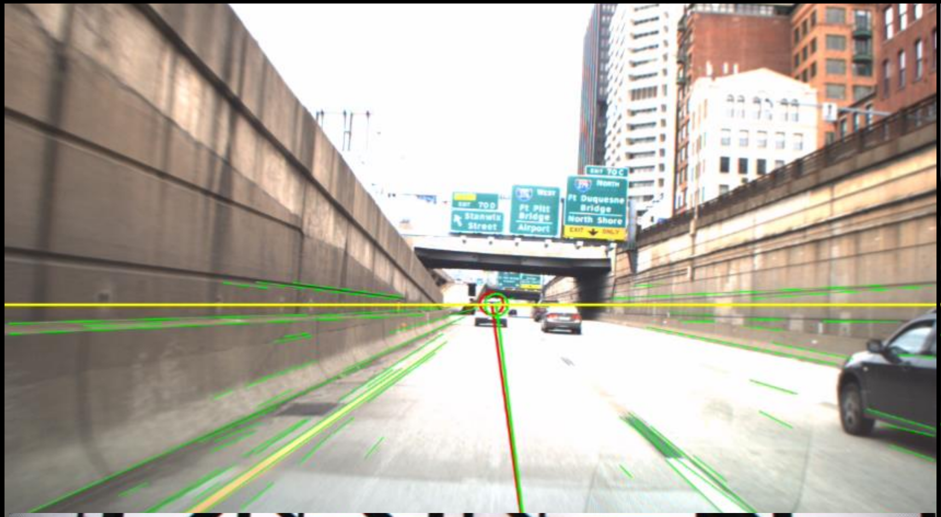
Vanishing Point Detection: An Example

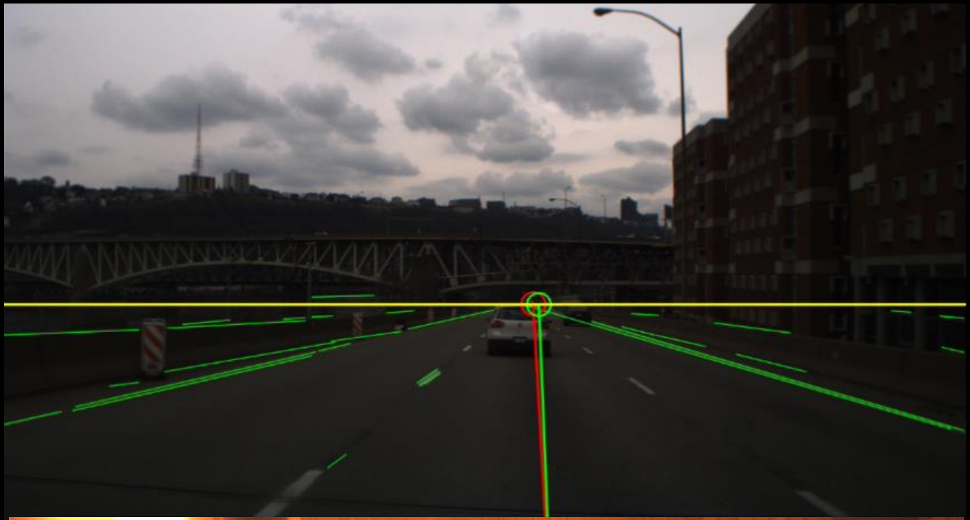


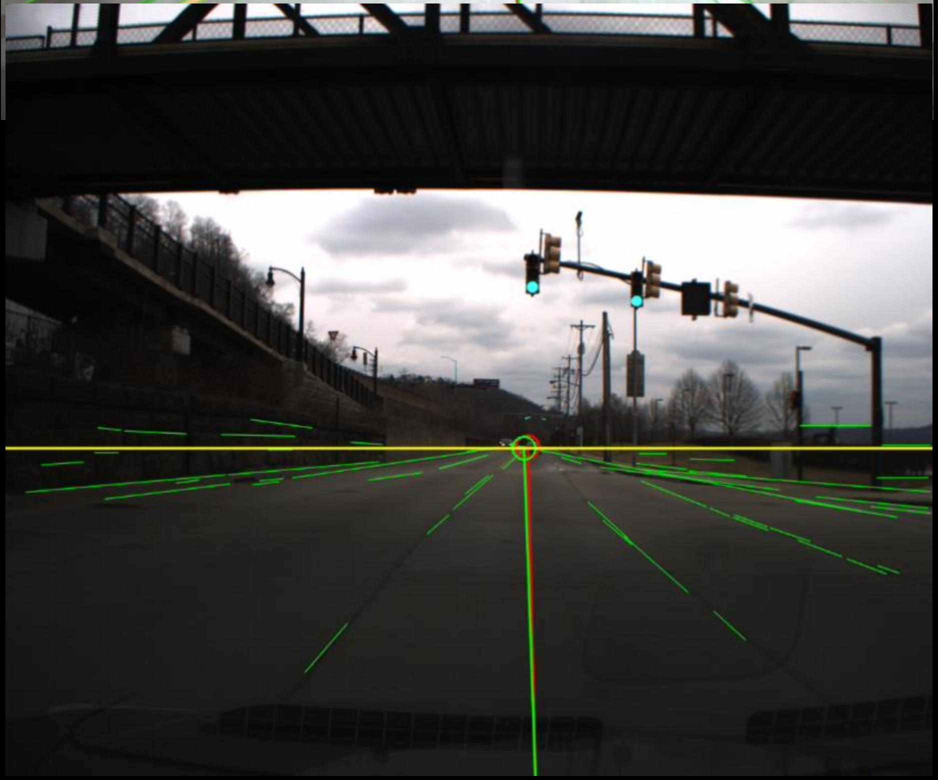
Vanishing Point Detection: Detection Results

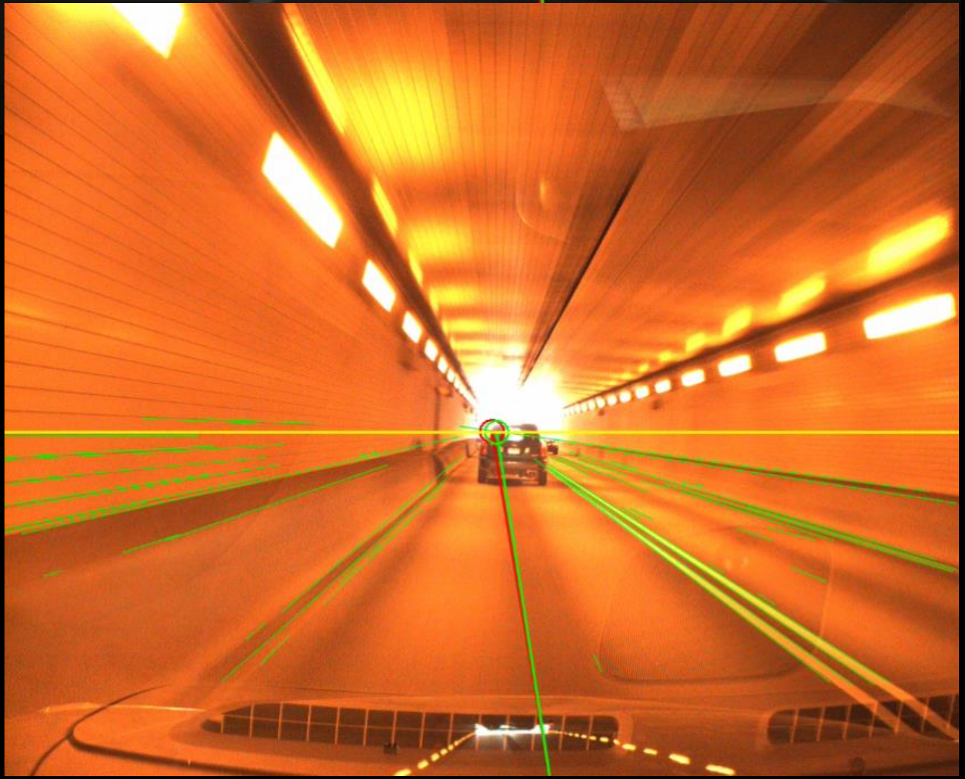





Vanishing Point Detection: Detection Results

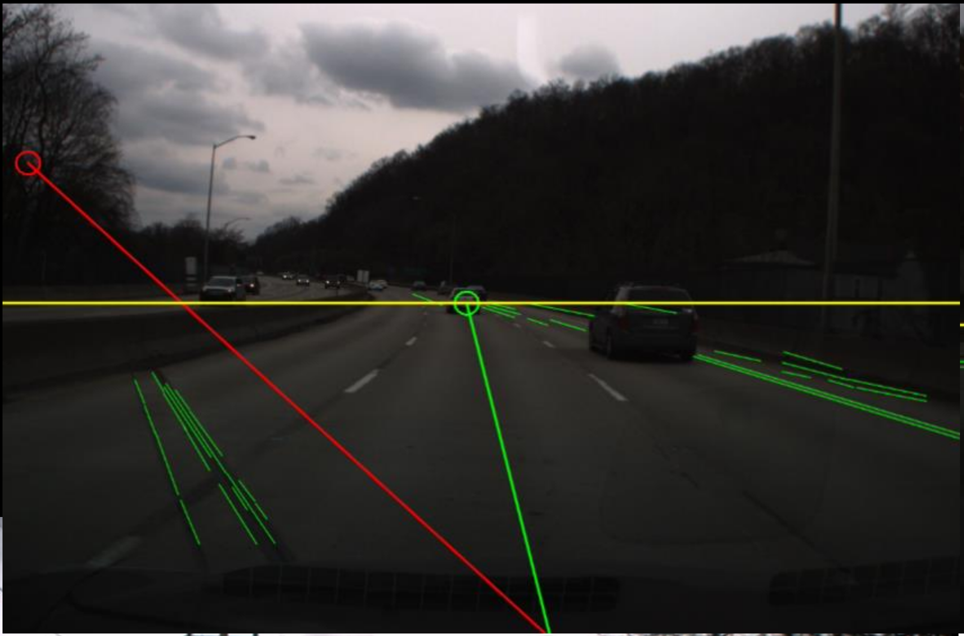
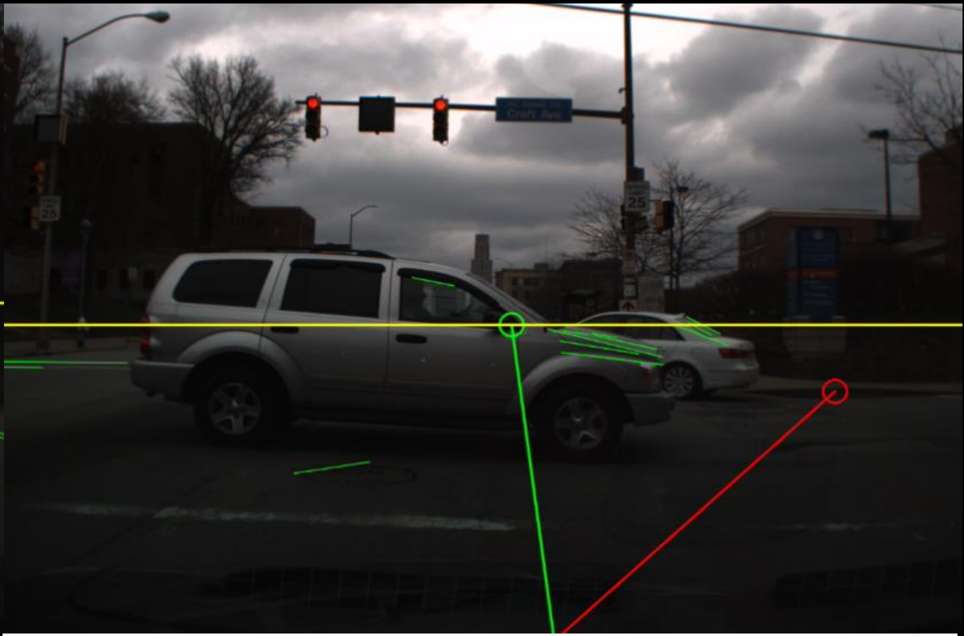
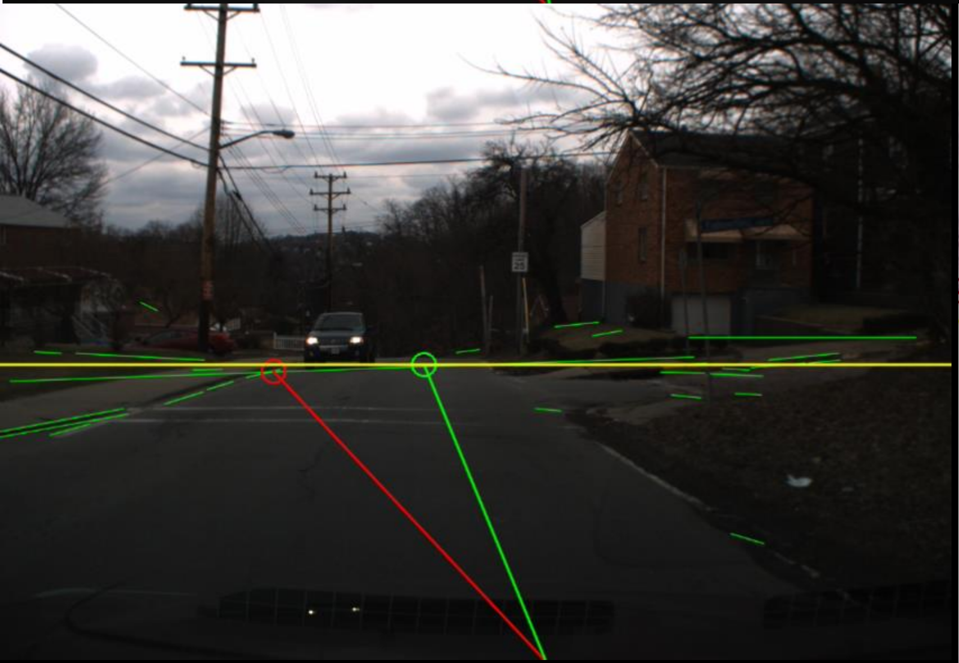
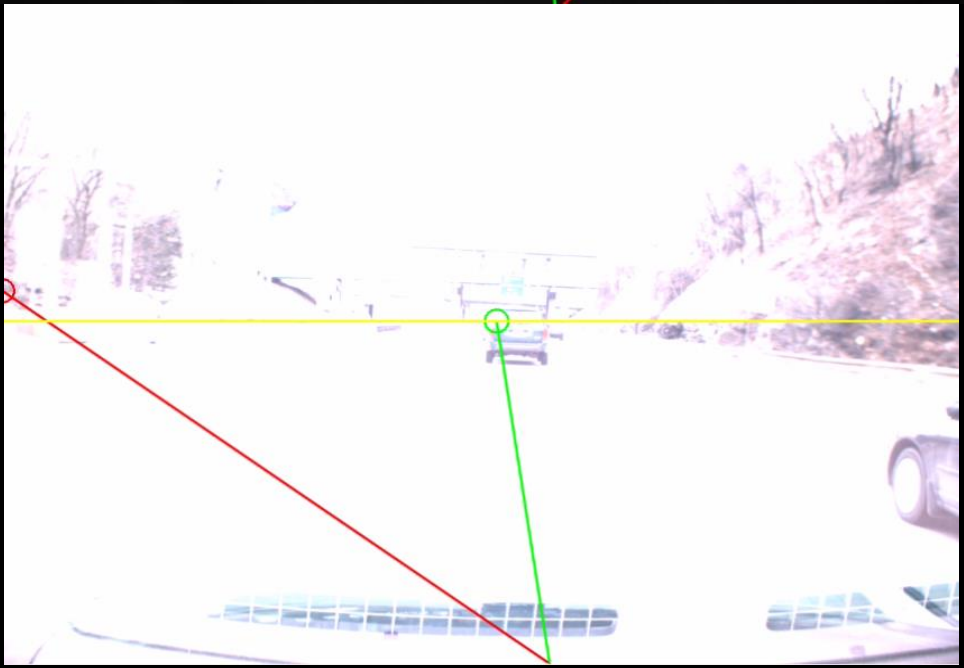








Vanishing Point Detection: Detection Results

Vanishing Point Tracking: Overview

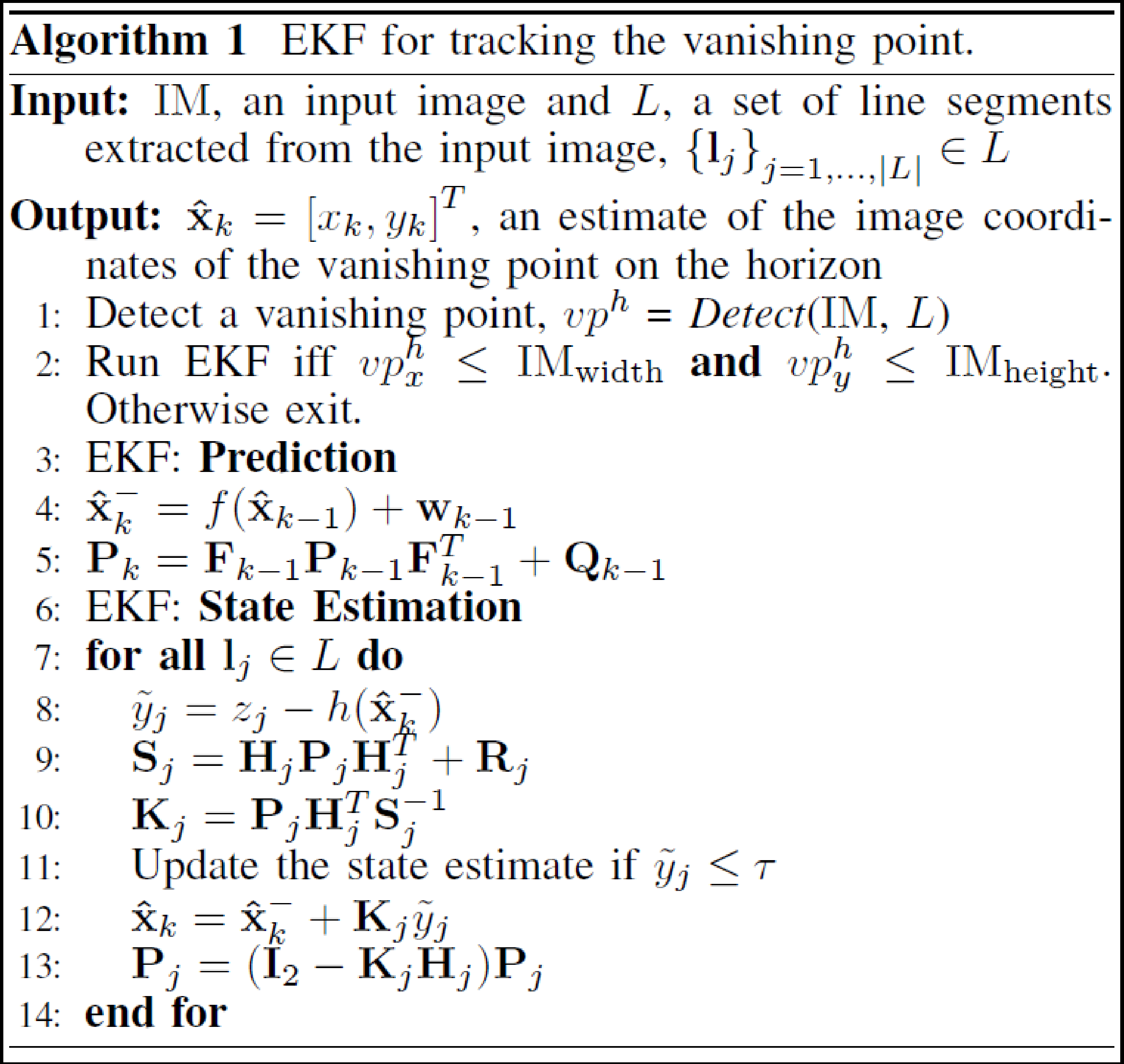
Extended Kalman Filter for tracking the vanishing point on the horizon:

- The locations of the vanishing point detected frame-by-frame basis may be inconsistent over the frames

- Track the image coordinates of a vanishing point using the extracted lines, which are used for detecting the vanishing point

- Smooth the detected locations of the vanishing point appearing on the horizon line, even with absence of relevant image features

Vanishing Point Tracking: Overview



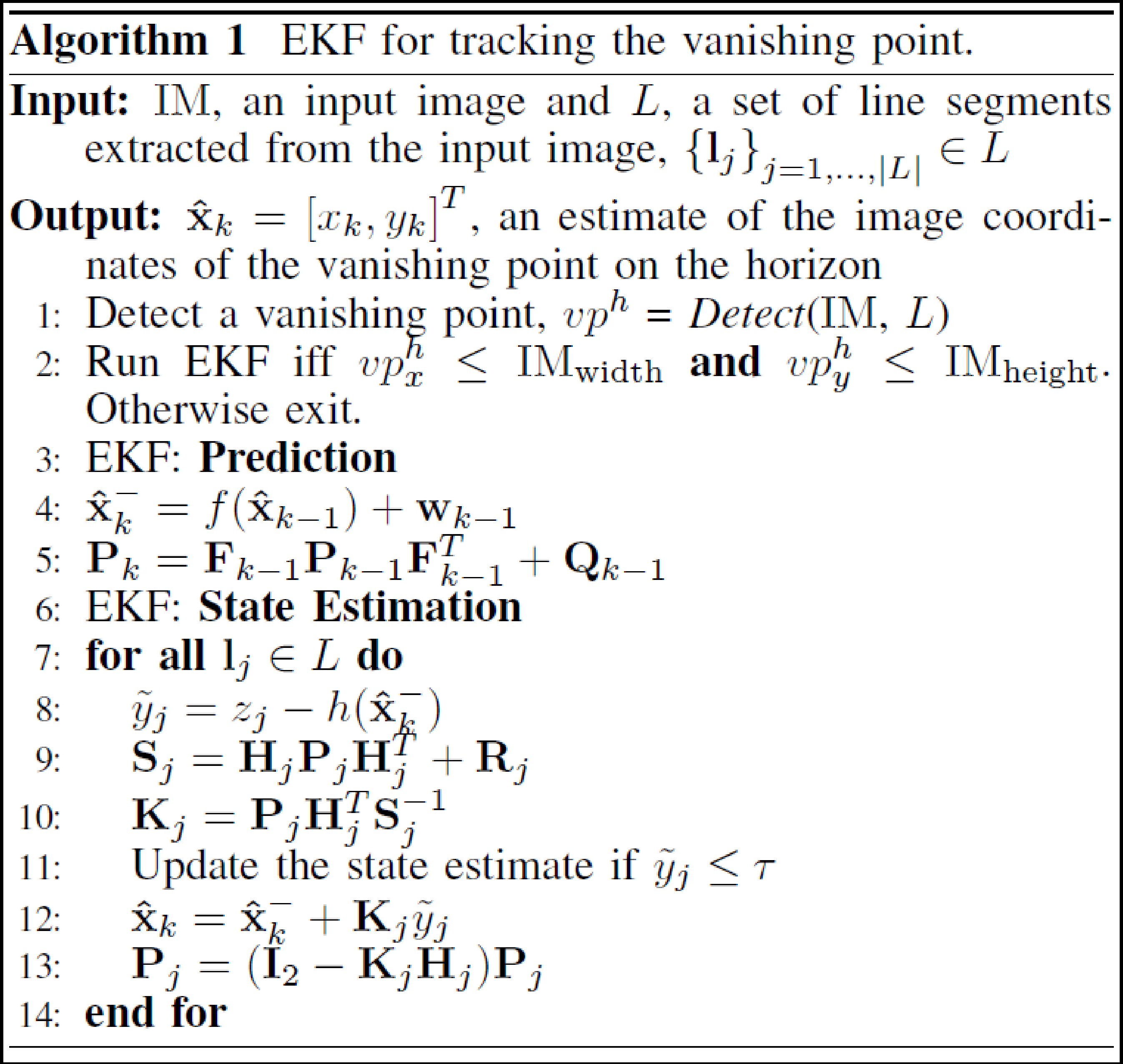
Algorithm 1 EKF for tracking the vanishing point.

Input: IM, an input image and L, a set of line segments extracted from the input image, $\{\mathbf{l}_j\}_{j=1,\ldots,|L|} \in L$

Output: $\hat{\mathbf{x}}_k = [x_k, y_k]^T$, an estimate of the image coordinates of the vanishing point on the horizon

```
 1: Detect a vanishing point, $vp^h$ = $Detect$(IM, $L$)
 2: Run EKF iff $vp^h_x \leq \mathrm{IM_{width}}$ and $vp^h_y \leq \mathrm{IM_{height}}$.
    Otherwise exit.
 3: EKF: Prediction
 4: $\hat{\mathbf{x}}_k^- = f(\hat{\mathbf{x}}_{k-1}) + \mathbf{w}_{k-1}$
 5: $\mathbf{P}_k = \mathbf{F}_{k-1}\mathbf{P}_{k-1}\mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1}$
 6: EKF: State Estimation
 7: for all $\mathbf{l}_j \in L$ do
 8:     $\tilde{y}_j = z_j - h(\hat{\mathbf{x}}_k^-)$
 9:     $\mathbf{S}_j = \mathbf{H}_j\mathbf{P}_j\mathbf{H}_j^T + \mathbf{R}_j$
10:     $\mathbf{K}_j = \mathbf{P}_j\mathbf{H}_j^T\mathbf{S}_j^{-1}$
11:     Update the state estimate if $\tilde{y}_j \leq \tau$
12:     $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_j\tilde{y}_j$
13:     $\mathbf{P}_j = (\mathbf{I}_2 - \mathbf{K}_j\mathbf{H}_j)\mathbf{P}_j$
14: end for
```

Vanishing Point Tracking: Overview



Algorithm 1 EKF for tracking the vanishing point.

Input: IM, an input image and L, a set of line segments extracted from the input image, $\{\mathbf{l}_j\}_{j=1,\ldots,|L|} \in L$

Output: $\hat{\mathbf{x}}_k = [x_k, y_k]^T$, an estimate of the image coordinates of the vanishing point on the horizon

```
1: Detect a vanishing point, $vp^h = Detect(\mathrm{IM}, L)$
2: Run EKF iff $vp^h_x \leq \mathrm{IM}_{\mathrm{width}}$ and $vp^h_y \leq \mathrm{IM}_{\mathrm{height}}$. Otherwise exit.
3: EKF: Prediction
4: $\hat{\mathbf{x}}_k^- = f(\hat{\mathbf{x}}_{k-1}) + \mathbf{w}_{k-1}$
5: $\mathbf{P}_k = \mathbf{F}_{k-1}\mathbf{P}_{k-1}\mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1}$
6: EKF: State Estimation
7: for all $\mathbf{l}_j \in L$ do
8:     $\tilde{y}_j = z_j - h(\hat{\mathbf{x}}_k^-)$
9:     $\mathbf{S}_j = \mathbf{H}_j\mathbf{P}_j\mathbf{H}_j^T + \mathbf{R}_j$
10:    $\mathbf{K}_j = \mathbf{P}_j\mathbf{H}_j^T\mathbf{S}_j^{-1}$
11:    Update the state estimate if $\tilde{y}_j \leq \tau$
12:    $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_j\tilde{y}_j$
13:    $\mathbf{P}_j = (\mathbf{I}_2 - \mathbf{K}_j\mathbf{H}_j)\mathbf{P}_j$
14: end for
```

State?

Initialization?

Process Model?

Measurement Model?

Vanishing Point Tracking: State Definition and Initialization

$$\mathbf{x}_k = [x_k, y_k]^T, \mathbf{P}_k = \begin{bmatrix} \sigma_{x,k} & \sigma_{xy,k} \\ \sigma^{-1}_{xy,k} & \sigma_{y,k} \end{bmatrix}$$

$$\mathbf{x}_0 = \left[\mathbf{I}_{width/2}, \mathbf{I}_{height/2}\right]$$

$$\mathbf{P}_0 = \begin{bmatrix} \left(\frac{x_{img}}{f_x}\right)^2 & 0 \\ 0 & \left(\frac{y_{img}}{f_y}\right)^2 \end{bmatrix}$$

where

$$x_{img} = f_x + c_x$$

The coordinates of the vanishing point are represented in the (normalized) camera coordinates

Re-Initialization: Re-initialize the state when the coordinates of the estimated vanishing point are projected out of the image coordinate

Vanishing Point Tracking: Process Model

$$\begin{aligned}\hat{\mathbf{x}}_k^- &= f(\hat{\mathbf{x}}_{k-1}) + \mathbf{Q}_k, \text{ where, } \mathbf{Q}_k \sim \mathbf{N}(0, \Sigma) \\ &= \mathbf{I}_2\hat{\mathbf{x}}_{k-1} + \mathbf{Q}_k\end{aligned}$$

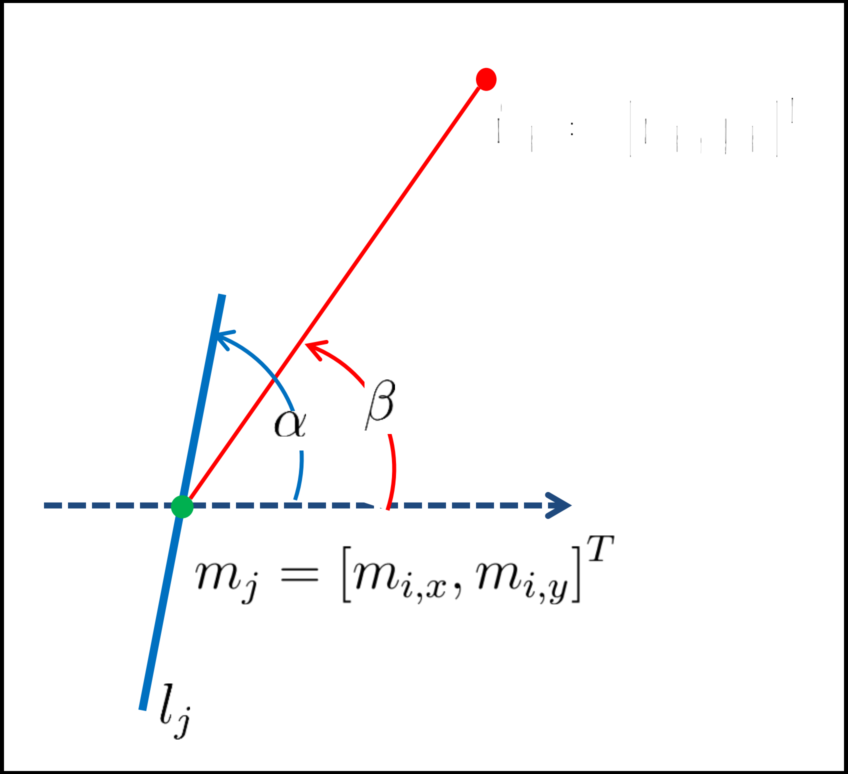
Predict the coordinates of the vanishing point at the next frame

No motion model (for now)

Vanishing Point Tracking: Measurement Model

$$\hat{\mathbf{z}}_k = h(\hat{\mathbf{x}}_k^-) + \mathbf{R}_k, \text{ where, } \mathbf{R}_k \sim \mathbf{N}(0, \Sigma)$$

Predict the expected line from the predicted state



$$h(\hat{\mathbf{x}}_k^-) = \tan^{-1}\left(\frac{x_k - m_{i,y}}{y_k - m_{i,x}}\right)$$

$$\frac{\partial h(\mathbf{x}_k)}{\partial \mathbf{x}} = \left[\frac{\partial h(\mathbf{x}_k)}{\partial x_k}, \frac{\partial h(\mathbf{x}_k)}{\partial y_k}\right]$$

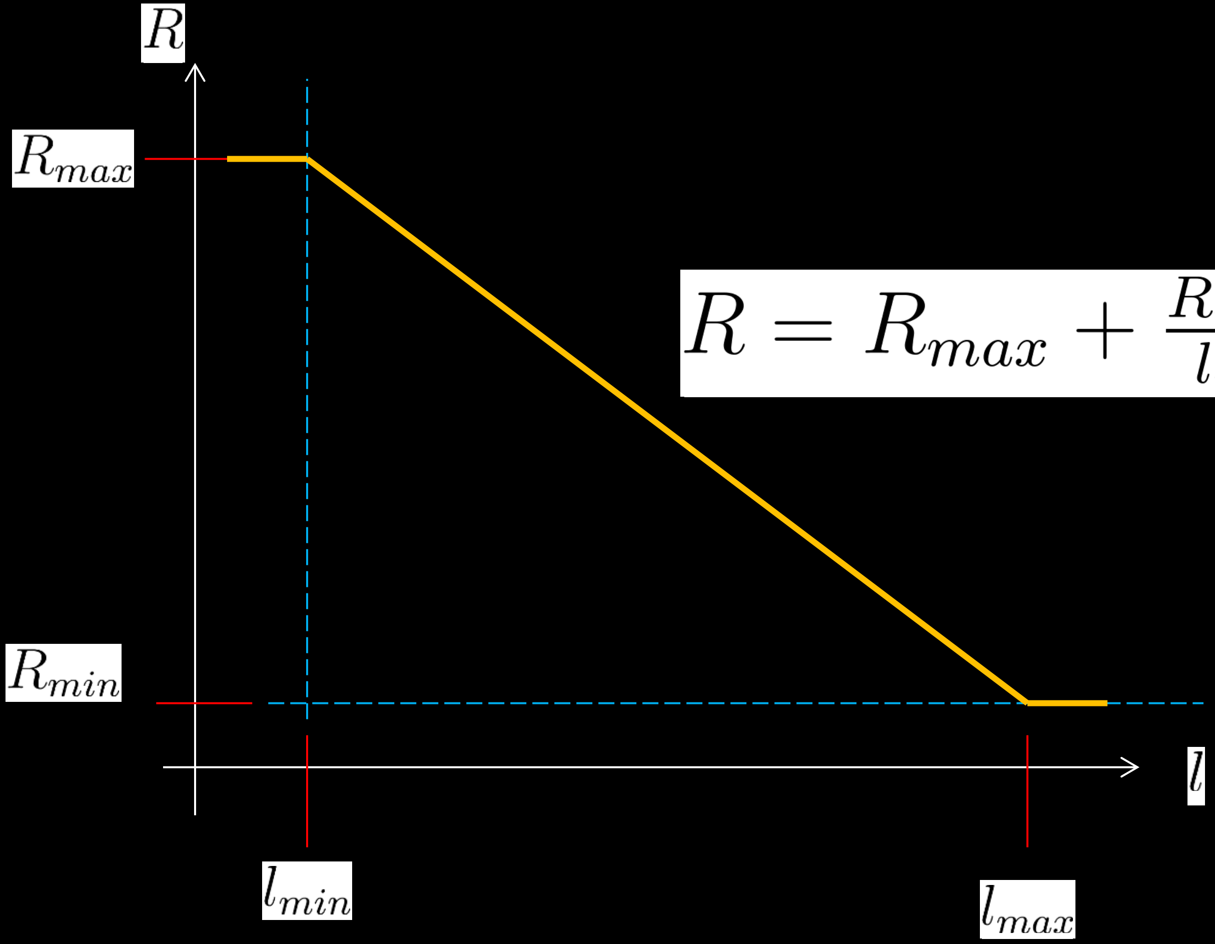
$$= \left[\frac{-(x_k - m_{i,y})}{d^2}, \frac{(y_k - m_{i,x})}{d^2}\right]$$

where,

$$d^2 = \sqrt{(x_k - m_{i,x})^2 + (y_k - m_{i,y})^2}$$

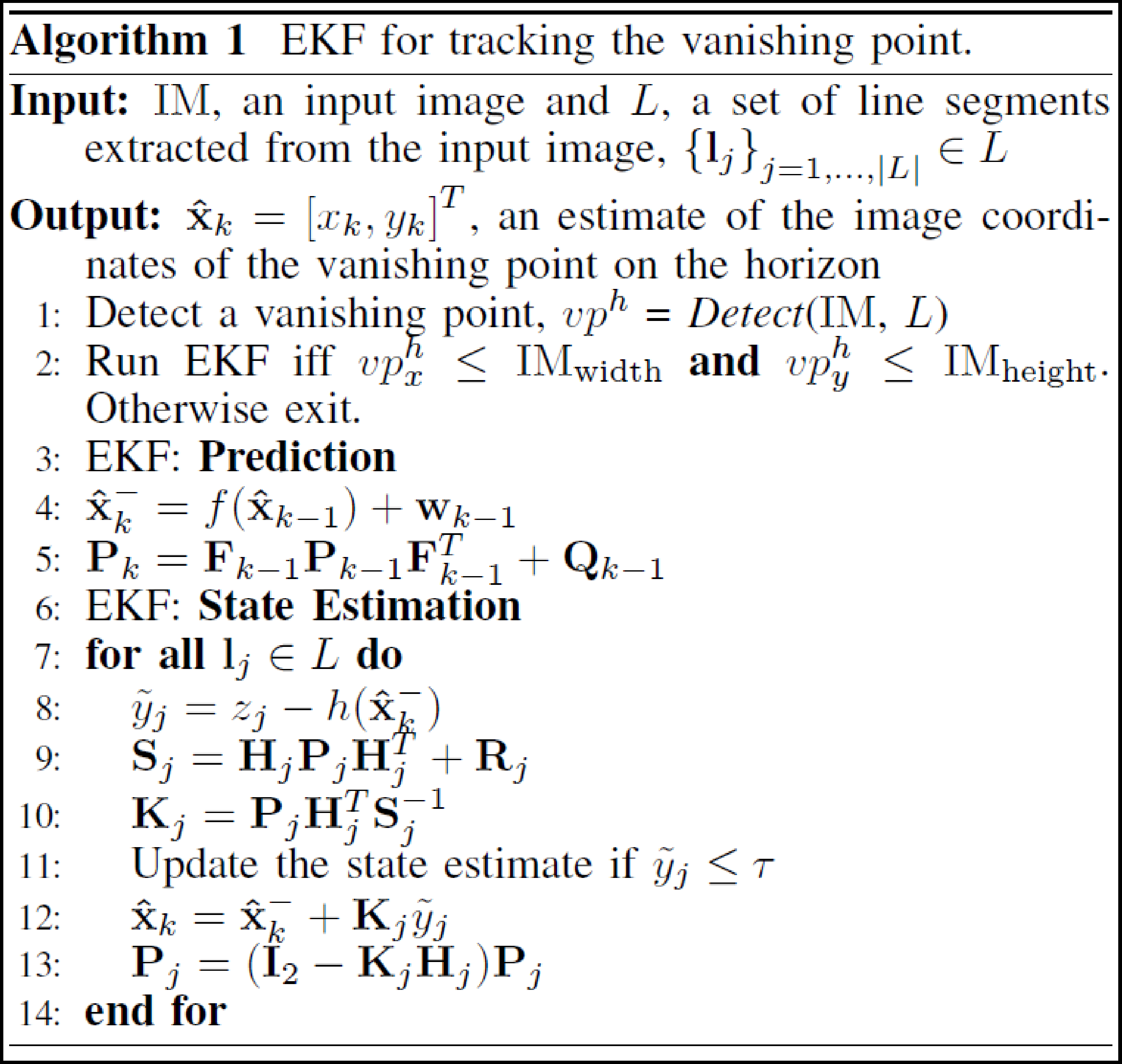
Vanishing Point Tracking: Measurement Model

Measurement update based on a line' fidelity to the current vanishing point: The longer a line the lower chance it is an outlier



$$R = R_{max} + \frac{R_{min} - R_{max}}{l_{max} - l_{min}} l$$

Vanishing Point Tracking: Summary



Algorithm 1 EKF for tracking the vanishing point.

Input: IM, an input image and L, a set of line segments extracted from the input image, $\{\mathbf{l}_j\}_{j=1,\ldots,|L|} \in L$

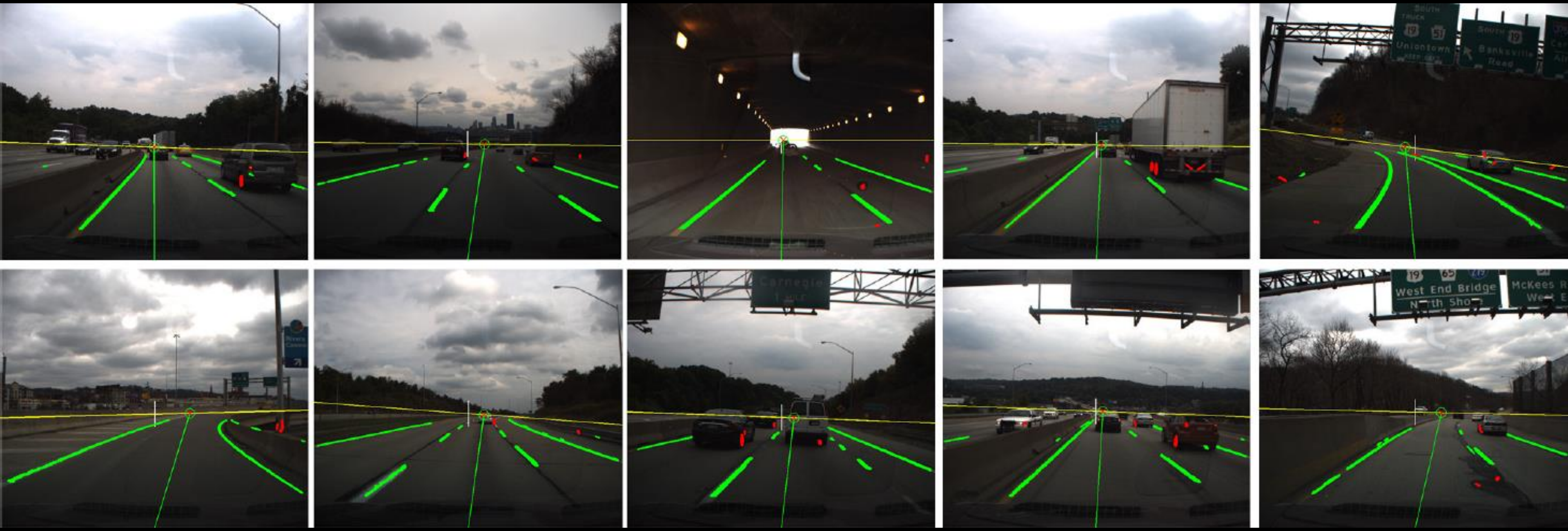
Output: $\hat{\mathbf{x}}_k = [x_k, y_k]^T$, an estimate of the image coordinates of the vanishing point on the horizon

```
1:  Detect a vanishing point, $vp^h$ = $Detect(\text{IM}, L)$
2:  Run EKF iff $vp^h_x \leq \text{IM}_{\text{width}}$ and $vp^h_y \leq \text{IM}_{\text{height}}$. Otherwise exit.
3:  EKF: Prediction
4:  $\hat{\mathbf{x}}^-_k = f(\hat{\mathbf{x}}_{k-1}) + \mathbf{w}_{k-1}$
5:  $\mathbf{P}_k = \mathbf{F}_{k-1}\mathbf{P}_{k-1}\mathbf{F}^T_{k-1} + \mathbf{Q}_{k-1}$
6:  EKF: State Estimation
7:  for all $\mathbf{l}_j \in L$ do
8:      $\tilde{y}_j = z_j - h(\hat{\mathbf{x}}^-_k)$
9:      $\mathbf{S}_j = \mathbf{H}_j\mathbf{P}_j\mathbf{H}^T_j + \mathbf{R}_j$
10:     $\mathbf{K}_j = \mathbf{P}_j\mathbf{H}^T_j\mathbf{S}^{-1}_j$
11:     Update the state estimate if $\tilde{y}_j \leq \tau$
12:     $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}^-_k + \mathbf{K}_j\tilde{y}_j$
13:     $\mathbf{P}_j = (\mathbf{I}_2 - \mathbf{K}_j\mathbf{H}_j)\mathbf{P}_j$
14: end for
```

Vanishing Point Detection and Tracking: Applications

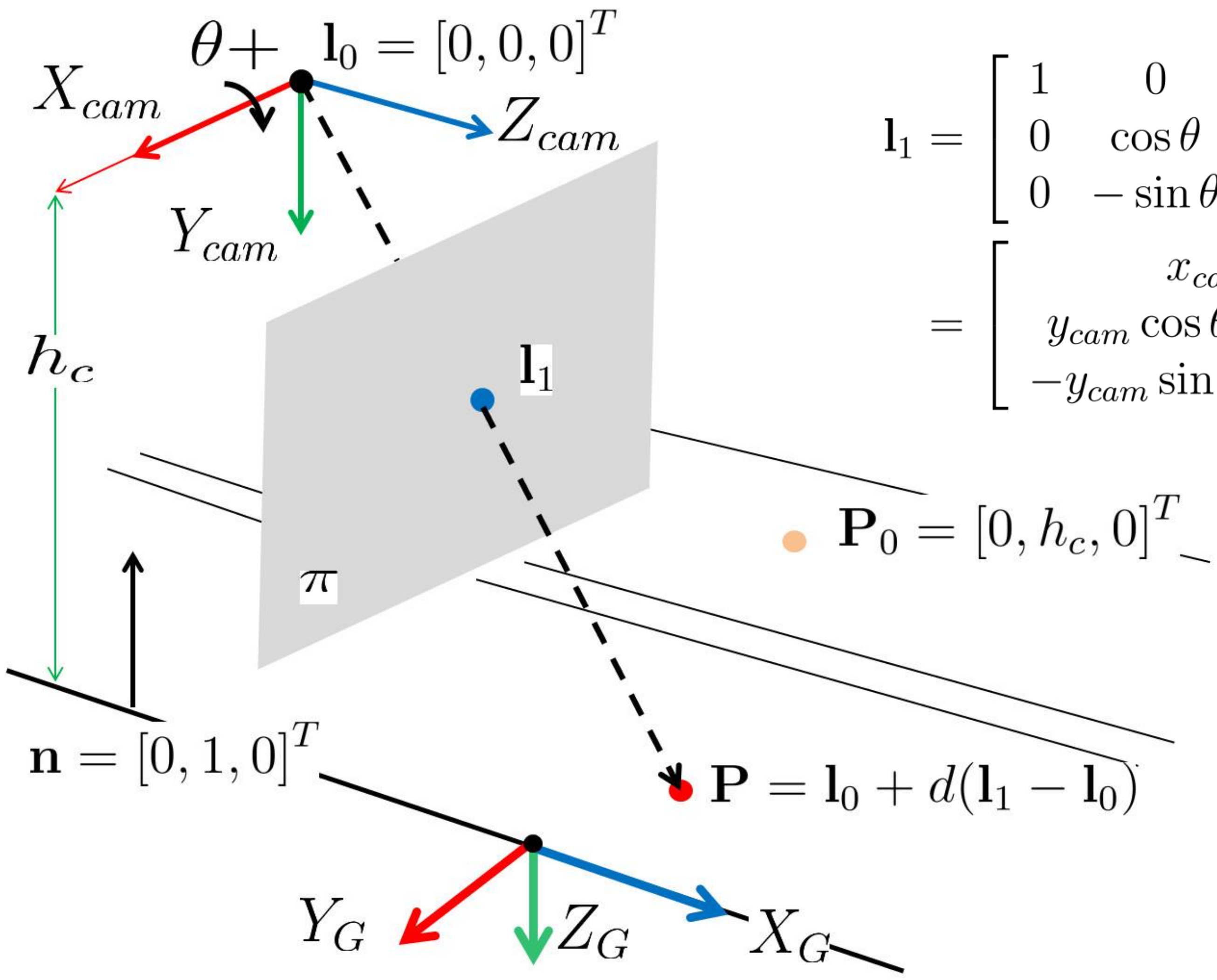
Estimation of road driving direction: To improve the performance of lane-marking detection [Seo and Rajkumar, 2014a] (IV-2014)

Estimation of pitch angle: To compute metric information of interesting objects on ground plane [Seo and Rajkumar, 2014b] (ITSC-14)



Metric Measurement: Homography

Estimation of pitch angle: To compute metric information of interesting objects on ground plane [Seo and Rajkumar, 2014b]


$\theta+$
$\mathbf{l}_0 = [0, 0, 0]^T$
X_{cam}
Z_{cam}
Y_{cam}
h_c
$\mathbf{l}_1$
π
$\mathbf{P}_0 = [0, h_c, 0]^T$
$\mathbf{n} = [0, 1, 0]^T$
$\mathbf{P} = \mathbf{l}_0 + d(\mathbf{l}_1 - \mathbf{l}_0)$
Y_G
Z_G
X_G

$$\mathbf{l}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_{cam} \\ y_{cam} \\ 1 \end{bmatrix} = \begin{bmatrix} x_{cam} \\ y_{cam}\cos\theta + \sin\theta \\ -y_{cam}\sin\theta + \cos\theta \end{bmatrix} \qquad (1)$$

Metric Measurement: Homography

$$\mathbf{P} = [X, Y, Z]^T = [Y_R, h_c, X_R]^T$$

$$\mathbf{P} = \mathbf{l}_0 + d(\mathbf{l}_1 - \mathbf{l}_0) = d\mathbf{l}_1 + \mathbf{l}_0 = d\mathbf{l}_1$$

$$(\mathbf{P} - \mathbf{P}_0) \cdot \mathbf{n} = 0$$

$$(d\mathbf{l}_1 + \mathbf{l}_0 - \mathbf{P}_0) \cdot \mathbf{n} = 0$$

$$d = \frac{(\mathbf{P}_0 - l_0) \cdot \mathbf{n}}{l_1 \cdot \mathbf{n}} = \frac{h_c}{y_{cam} \cos\theta + \sin\theta} \qquad (2)$$

$$\mathbf{P} = dl_1 \qquad (1) \times (2)$$

$$= \frac{h_c}{y_{cam} \cos\theta + \sin\theta} \begin{bmatrix} x_{cam} \\ y_{cam} \cos\theta + \sin\theta \\ -y_{cam} \sin\theta + \cos\theta \end{bmatrix}$$

$$= \left[h_c \frac{x_{cam}}{y_{cam} \cos\theta + \sin\theta}, h_c, h_c \frac{-y_{cam} \sin\theta + \cos\theta}{y_{cam} \cos\theta + \sin\theta} \right]^T$$

Metric Measurement: Pitch Angle Estimation

The underlying idea is to compute the pitch (or yaw) angles from the computation of the difference of coordinates between the camera center and the vanishing point on a horizon line

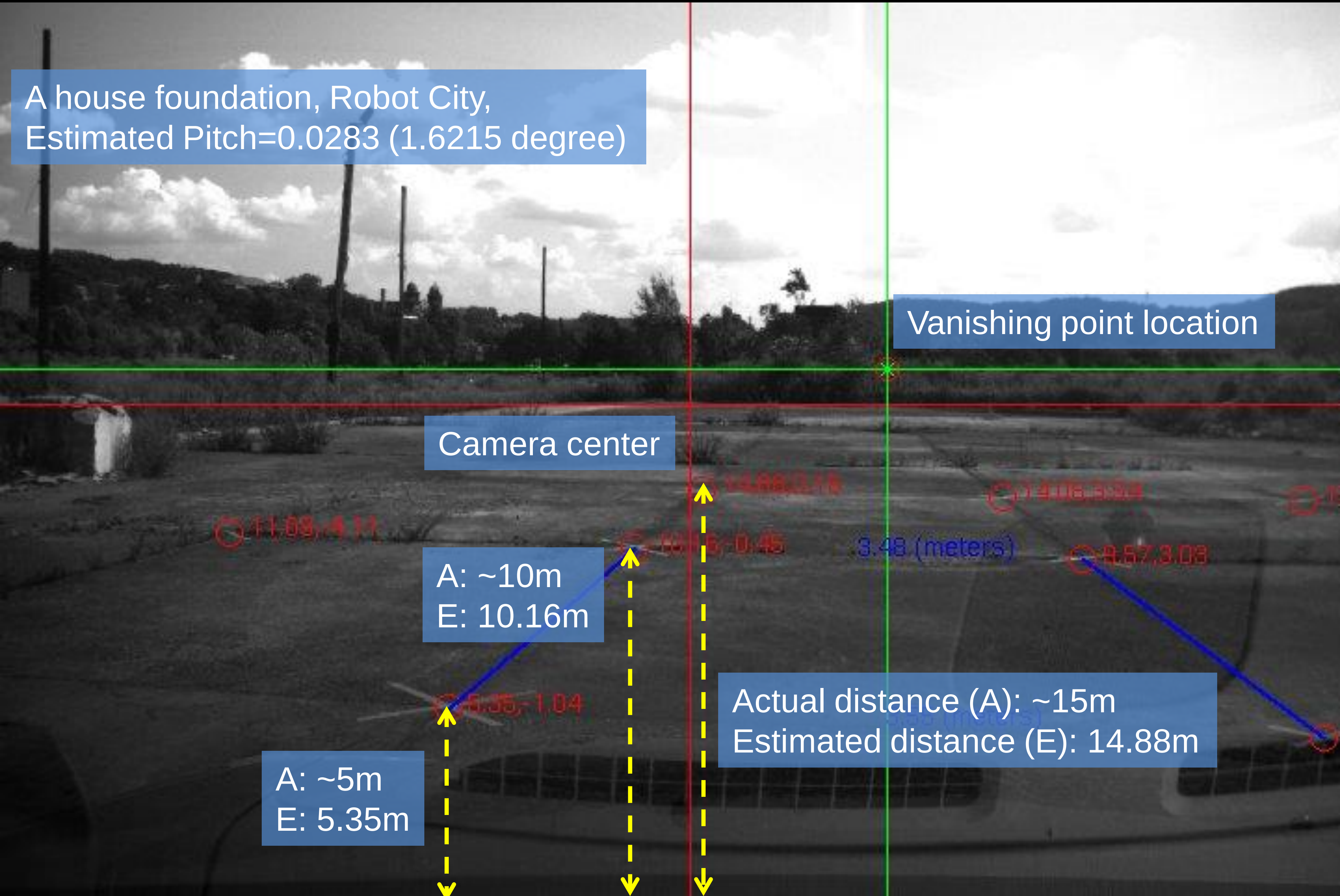
$$vp_h^*(\phi, \theta\,\varphi) = \left[\frac{\cos\phi\sin\varphi - \sin\phi\sin\theta\cos\varphi}{\cos\theta\cos\varphi}, \frac{-\sin\phi\sin\varphi - \cos\phi\sin\theta\cos\varphi}{\cos\theta\cos\varphi}\right]$$

$$(\phi, \theta\,\varphi) = (\mathrm{roll}, \mathrm{pitch}, \mathrm{yaw})$$

$$vp_h^*(0, \theta\,0) = \left[\frac{0}{\cos\theta}, \frac{-\sin\theta}{\cos\theta}\right]$$

$$vp_h^*(0, \theta\,\varphi) = \left[\frac{\sin\varphi - 0}{\cos\theta\cos\varphi}, \frac{-\sin\theta}{\cos\theta}\right]$$

Metric Measurement: Model Verification



A house foundation, Robot City,
Estimated Pitch=0.0283 (1.6215 degree)

Vanishing point location

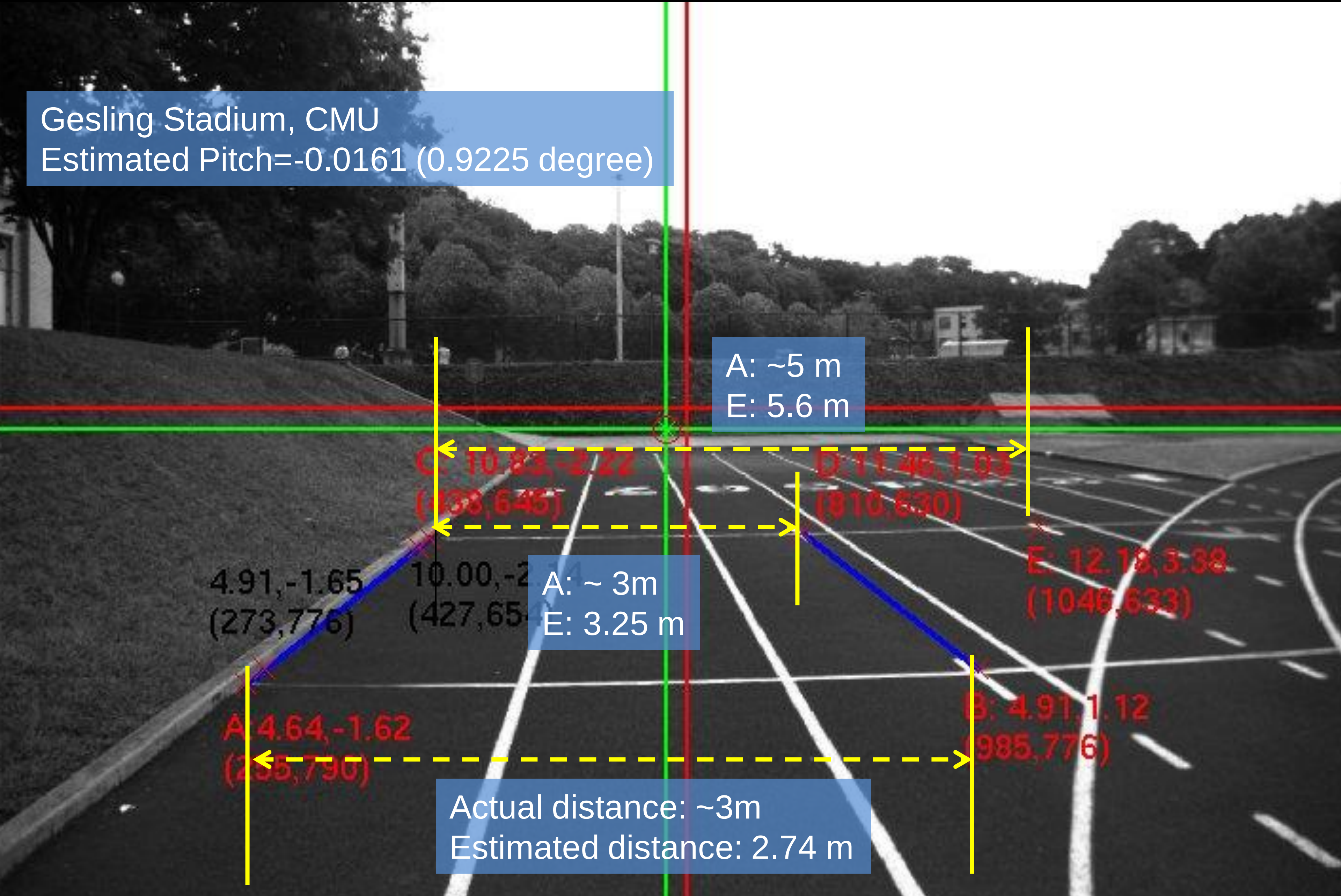
Camera center

A: ~10m
E: 10.16m

Actual distance (A): ~15m
Estimated distance (E): 14.88m

A: ~5m
E: 5.35m

Metric Measurement: Model Verification



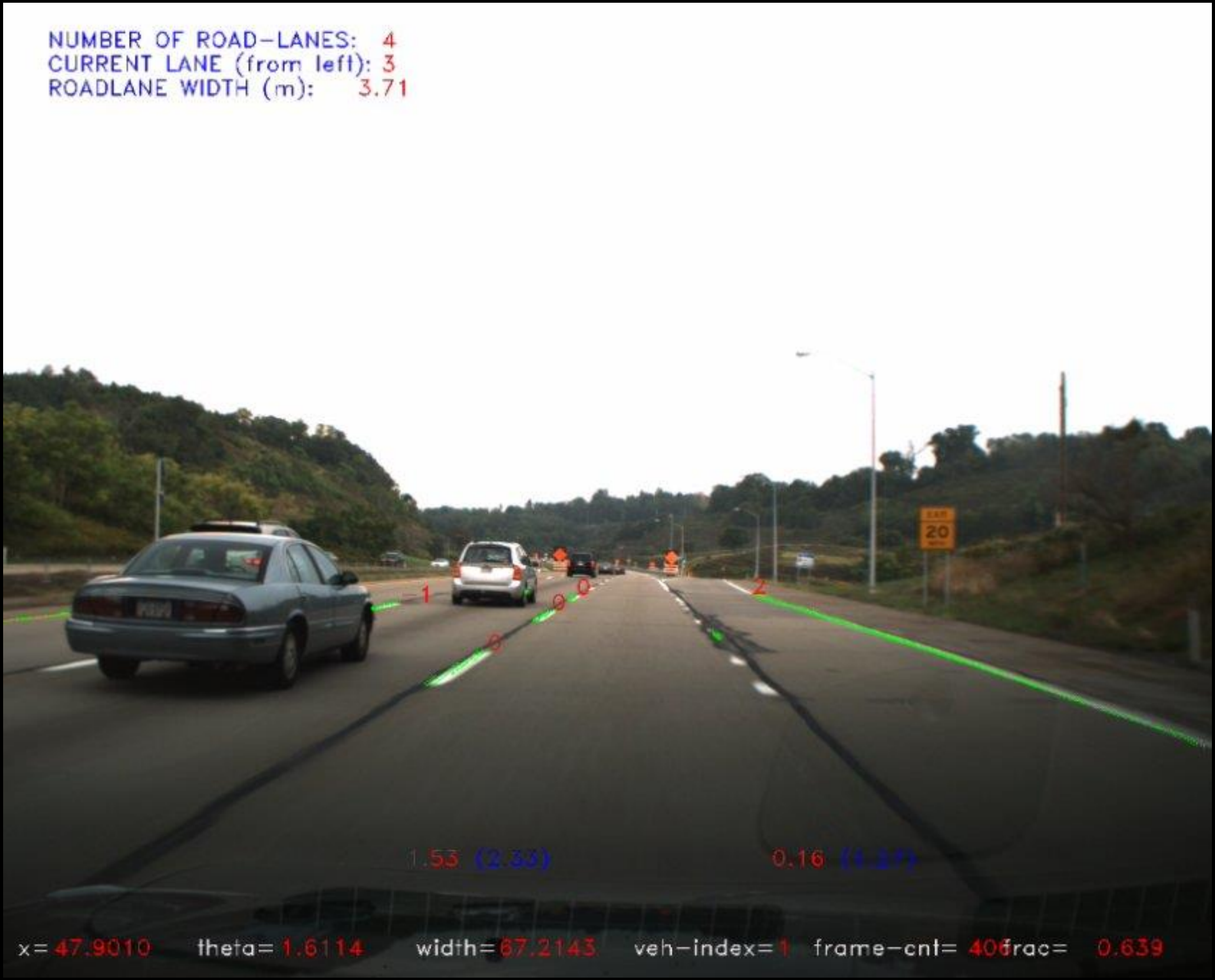
Gesling Stadium, CMU
Estimated Pitch=-0.0161 (0.9225 degree)

A: ~5 m
E: 5.6 m

A: ~ 3m
E: 3.25 m

Actual distance: ~3m
Estimated distance: 2.74 m

Metric Measurement: Example

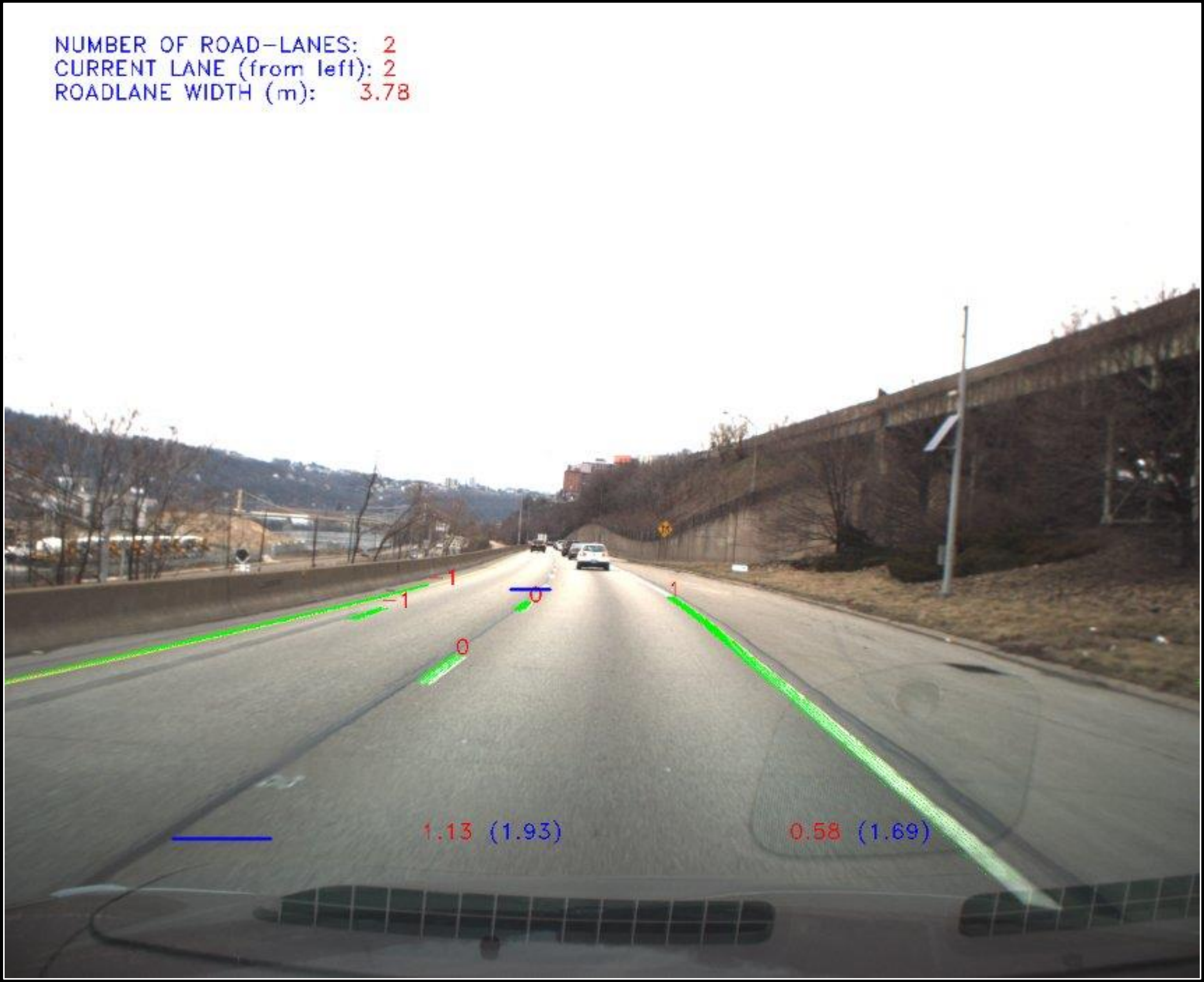


```
NUMBER OF ROAD-LANES:   4
CURRENT LANE (from left): 3
ROADLANE WIDTH (m):    3.71
```

```
1.53 (2.33)                0.16 (4.2[illegible])
```

```
x= 47.9010    theta= 1.6114    width=67.2143   veh-index=1  frame-cnt= 406frac=   0.639
```

Metric Measurement: Example



Experiments

Experimental Settings

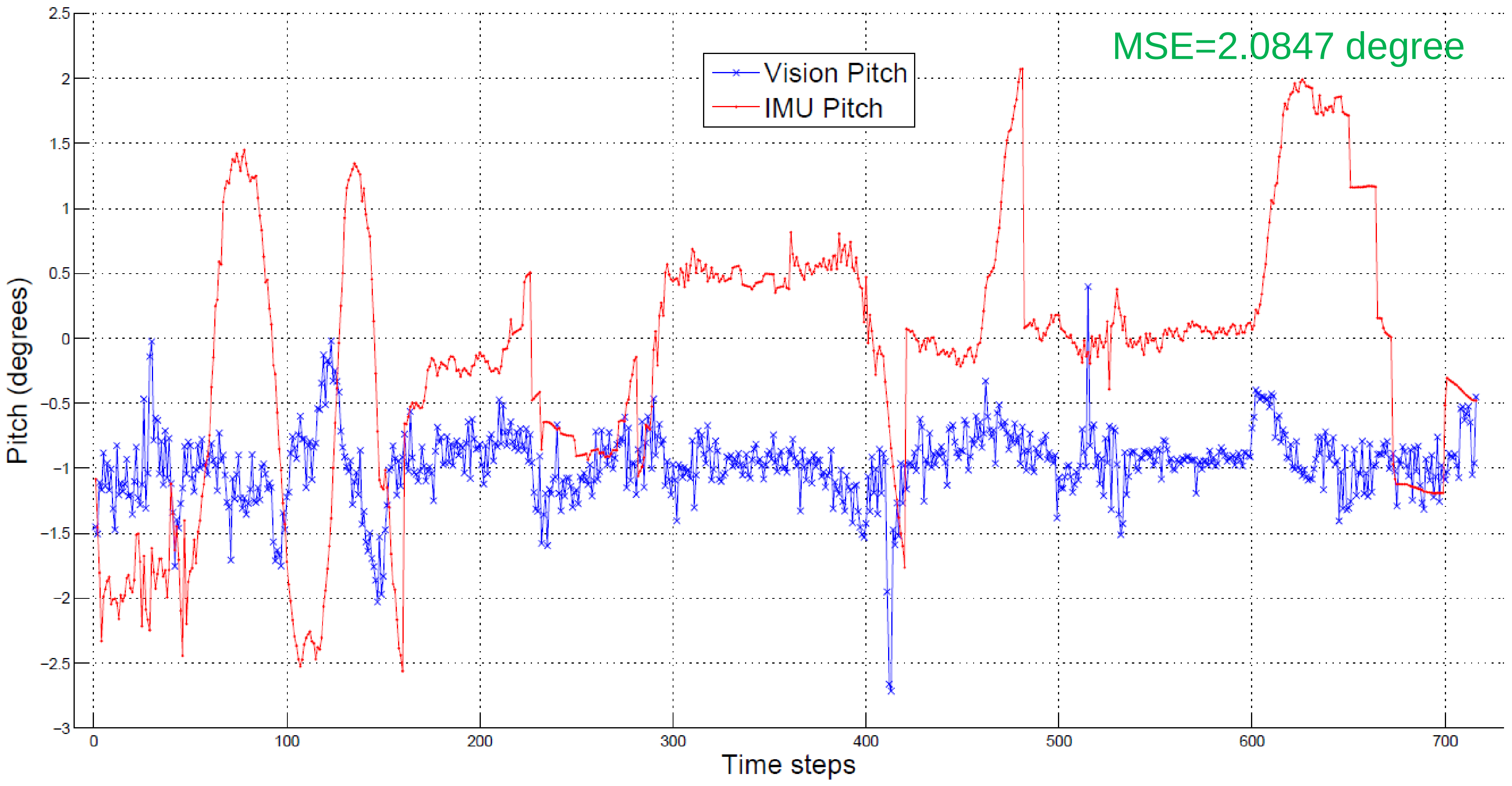
- The developed algorithms were implemented in C++ and OpenCV and ran on a self-driving car at 10Hz.

- Sensors and System:
    - Monocular vision sensor
        - Flea3 (FL3-GE-50S5C-C), CCD 2/3", 2448x2024 (1224x1024), 8fps
        - 8mm, HFOV=57.6, VFOV=44.8
        - Mounting height: 1.46m from the ground
    - Navigation solution
        - Applanix POS-LV w/ RTK corrections
        - RMS, 0.02 (0.06) degree pitch angle measurement with RTK corrections (GPS outage)

- Testing roads
    - Mostly inter-city highways, i.e., I-376, I-279, I-76
    - Some urban streets in Pittsburgh

Experimental Results: Pitch Angle Comparison

Compare the pitch angles measured by IMU with that measured by the developed algorithm



Vanishing Point Detection and Tracking: Video

Green circle is the vanishing point **tracked** over the frames.

Red circle is the one **detected** from each frame.

Yellow horizontal line is a detected horizon line.

Summary and Future Work

Developed a computer vision algorithm

- Detected vanishing points using the extracted lines
- Tracked, using EKF, the vanishing point on a horizon over frames

Through testing with inter-city highways videos, we demonstrated that the developed algorithms produced stable and reliable performance in tracking the vanishing point on a horizon line

Developed methods are used for 1) approximating road driving direction and 2) estimating the pitch angle between image and road plane

More field testing: To determine the limits of our algorithms, continue testing it against various driving environments.

Acknowledgements

I would like to thank Dr. Myung Hwangbo for fruitful discussion about 3D vision, Junsung Kim for data collection, and Prof. Raj Rajkumar for his support on this work.

Thank You

Questions or Comments?

young-woo.seo@ri.cmu.edu