Motivation

Knowledge of a **horizon line** and the **vanishing point** on the horizon line provides us with 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
- Line extraction
- Line classification: Vertical and Horizontal
- Vanishing Point Detection through RANSAC

Vanishing Point Tracking using EKF
- Motion model
- Observation model

Vanishing Point Detection and Tracking Applications

Experiments

Summary and Future Work
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**

\[ \mathbf{v} = [0, 1, 0]^T \]

\[ \mathbf{h} = [0, 0, 1]^T \]
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  

![Image of line segment and vanishing point detection]

\[
l_i = a_i x + b_i y + c_i = \begin{bmatrix} a_i \\ b_i \\ c_i \end{bmatrix}
\]

\[
l_i^T \cdot v_j^* = \frac{[a_i, b_i, c_i]^T [v_{j,1}, v_{j,2}, v_{j,3}]}{\sqrt{a_i^2 + b_i^2 + c_i^2}}
\]

![Diagram of line classification with vectors v and h]

\[
v = [0, 1, 0]^T
\]

\[
h = [0, 0, 1]^T
\]
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

A vanishing point on horizon line

Estimated Horizon line
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, \( \{l_j\}_{j=1,...,|L|} \in L \)

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

1. Detect a vanishing point, \( vp^h = \text{Detect}(IM, L) \)
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. \( \hat{x}_k^- = f(\hat{x}_{k-1}) + w_{k-1} \)
5. \( P_k = F_{k-1} P_{k-1} F_{k-1}^T + Q_{k-1} \)
6. EKF: State Estimation
7. **for all** \( l_j \in L \) **do**
8. \( \tilde{y}_j = z_j - h(\hat{x}_k^-) \)
9. \( S_j = H_j P_j H_j^T + R_j \)
10. \( K_j = P_j H_j^T S_j^{-1} \)
11. Update the state estimate if \( \tilde{y}_j \leq \tau \)
12. \( \hat{x}_k = \hat{x}_k^- + K_j \tilde{y}_j \)
13. \( P_j = (I_2 - K_j H_j) P_j \)
14. **end for**
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, $\{l_j\}_{j=1,...,|L|} \in L$

**Output:** $\hat{x}_k = [x_k, y_k]^T$, 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 $vp^h_x \leq IM_{\text{width}}$ and $vp^h_y \leq IM_{\text{height}}$. Otherwise exit.
3. **EKF: Prediction**
4. $\hat{x}_k^- = f(\hat{x}_{k-1}) + w_{k-1}$
5. $P_k = F_{k-1} P_{k-1} F_{k-1}^T + Q_{k-1}$
6. **EKF: State Estimation**
7. for all $l_j \in L$ do
8. $\tilde{y}_j = z_j - h(\hat{x}_k^-)$
9. $S_j = H_j P_j H_j^T + R_j$
10. $K_j = P_j H_j^T S_j^{-1}$
11. Update the state estimate if $\tilde{y}_j \leq \tau$
12. $\hat{x}_k = \hat{x}_k^- + K_j \tilde{y}_j$
13. $P_j = (I_2 - K_j H_j) P_j$
14. end for
Vanishing Point Tracking: **State Definition and Initialization**

\[ x_k = [x_k, y_k]^T, \quad P_k = \begin{bmatrix} \sigma_{x,k} & \sigma_{xy,k} \\ -1 & \sigma_{xy,k} & \sigma_{y,k} \end{bmatrix} \]

\[ x_0 = \left[ \frac{I_{width}}{2}, \frac{I_{height}}{2} \right] \]

\[ P_0 = \begin{bmatrix} \left( \frac{x_{img}}{f_x} \right)^2 & 0 \\ 0 & \left( \frac{y_{img}}{f_y} \right)^2 \end{bmatrix} \]

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**

\[
\hat{x}_{k}^- = f(\hat{x}_{k-1}) + Q_k, \text{ where, } Q_k \sim \mathcal{N}(0, \Sigma)
\]

\[
= I_2 \hat{x}_{k-1} + Q_k
\]

Predict the coordinates of the vanishing point at the next frame

No motion model (for now)
Vanishing Point Tracking: Measurement Model

\[ \hat{Z}_k = h(\hat{x}_k^-) + \mathbf{R}_k, \text{ where, } \mathbf{R}_k \sim \mathcal{N}(0, \Sigma) \]

Predict the expected line from the predicted state

\[ \hat{Z}_k = \frac{\alpha}{\beta} \]

\[ m_j = [m_{i,x}, m_{i,y}]^T \]

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

\[ \frac{\partial h(x_k)}{\partial x} = \begin{bmatrix} \frac{\partial h(x_k)}{\partial x_k} \frac{\partial h(x_k)}{\partial y_k} \end{bmatrix} \]

\[ = \begin{bmatrix} \frac{- (x_k - m_{i,y})}{d^2} \frac{(y_k - m_{i,x})}{d^2} \end{bmatrix} \]

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’s fidelity to the current vanishing point: The longer a line the lower chance it is an outlier

\[ R = R_{\text{max}} + \frac{R_{\text{min}} - R_{\text{max}}}{l_{\text{max}} - l_{\text{min}}} l \]
Vanishing Point Tracking: Summary

**Algorithm 1** EKF for tracking the vanishing point.

**Input:** $\mathbb{I}M$, an input image and $L$, a set of line segments extracted from the input image, $\{l_j\}_{j=1,\ldots,|L|} \in L$

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

1. Detect a vanishing point, $vp^h = Detect(\mathbb{I}M, L)$
2. Run EKF iff $vp^h_x \leq \mathbb{I}M_{\text{width}}$ and $vp^h_y \leq \mathbb{I}M_{\text{height}}$. Otherwise exit.
3. EKF: Prediction
4. $\hat{x}_k^- = f(\hat{x}_{k-1}) + w_{k-1}$
5. $P_k = F_{k-1} P_{k-1} F_{k-1}^T + Q_{k-1}$
6. EKF: State Estimation
7. for all $l_j \in L$ do
8. $\tilde{y}_j = z_j - h(\hat{x}_k^-)$
9. $S_j = H_j P_j H_j^T + R_j$
10. $K_j = P_j H_j^T S_j^{-1}$
11. Update the state estimate if $\tilde{y}_j \leq \tau$
12. $\hat{x}_k = \hat{x}_k^- + K_j \tilde{y}_j$
13. $P_j = (I_2 - K_j H_j) P_j$
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]

\[ l_0 = [0, 0, 0]^T \]

\[ l_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & \sin \theta \\ 0 & -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x_{cam} \\ y_{cam} \\ 1 \end{bmatrix} \]

\[ P_0 = [0, h_c, 0]^T \]

\[ P = l_0 + d(l_1 - l_0) \]
Metric Measurement: Homography

\[ P = [X, Y, Z]^T = [Y_R, h_c, X_R]^T \]

\[ P = l_0 + d(l_1 - l_0) = dl_1 + l_0 = dl_1 \]

\[
\begin{align*}
(P - P_0) \cdot n &= 0 \\
(dl_1 + l_0 - P_0) \cdot n &= 0 \\
d &= \frac{(P_0 - l_0) \cdot n}{l_1 \cdot n} = \frac{h_c}{y_{cam} \cos \theta + \sin \theta} 
\end{align*}
\]

(2)

\[ P = dl_1 \quad (1) \times (2) \]

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

\[
= \begin{bmatrix}
h_c \frac{x_{cam}}{y_{cam} \cos \theta + \sin \theta}, h_c, h_c \frac{-y_{cam} \sin \theta + \cos \theta}{y_{cam} \cos \theta + \sin \theta}
\end{bmatrix}^T
\]
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.

\[
v_{p_h}(\phi, \theta, \varphi) = \left[ \frac{\cos \phi \sin \varphi - \sin \phi \sin \theta \cos \varphi}{\cos \theta \cos \varphi}, \frac{-\sin \phi \sin \varphi - \cos \phi \sin \theta \cos \varphi}{\cos \theta \cos \varphi} \right]
\]

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

\[
v_{p_h}(0, \theta, 0) = \begin{bmatrix} 0 & -\sin \theta \\ \cos \theta & \cos \theta \end{bmatrix}
\]

\[
v_{p_h}(0, \theta, \varphi) = \begin{bmatrix} \sin \varphi - 0 & -\sin \theta \\ \cos \theta \cos \varphi & \cos \theta \end{bmatrix}
\]
Metric Measurement: Model Verification

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

A: ~5m
E: 5.35m

A: ~10m
E: 10.16m

Actual distance (A): ~15m
Estimated distance (E): 14.88m
Metrics Measurement: **Model Verification**

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

Actual distance: ~3m
Estimated distance: 2.74 m

A: ~5 m
E: 5.6 m

A: ~3m
E: 3.25 m

A: 4.64, -1.62
(295.990)

B: 4.91, 1.12
(985.776)

C: 10.00, -2
(427.65)

D: 11.46, 1.95
(910.630)

E: 12.18, 3.38
(1046.633)
Metric Measurement: **Example**

NUMBER OF ROAD-LANES: 4  
CURRENT LANE (from left): 3  
ROADLANE WIDTH (m):  3.71
Metric Measurement: **Example**

- **NUMBER OF ROAD-LANES:** 2
- **CURRENT LANE (from left):** 2
- **ROADLANE WIDTH (m):** 3.78
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.

MSE=2.0847 degree
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