Detection and Tracking the Vanishing Point on the Horizon

Young-Woo Seo

CMU-RI-TR-14-07

May 2014

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

In advanced driver assistance systems and autonomous driving vehicles, many computer vision applications rely on knowing the location of the vanishing point on a horizon. The horizontal vanishing point's location provides important information about driving environments, such as the instantaneous driving direction of roadway, sampling regions of the drivable regions' image features, and the search direction of moving objects. To detect the vanishing point, many existing methods work frame-by-frame. Their outputs may look optimal in that frame. Over a series of frames, however, the detected locations are inconsistent, yielding unreliable information about roadway structure. This paper presents a novel algorithm that, using line segments, detects vanishing points in urban scenes and, using Extended Kalman Filter (EKF), tracks them over frames to smooth out the trajectory of the horizontal vanishing point. The study demonstrates both the practicality of the detection method and the effectiveness of our tracking method, through experiments carried out using thousands of urban scene images.

Contents

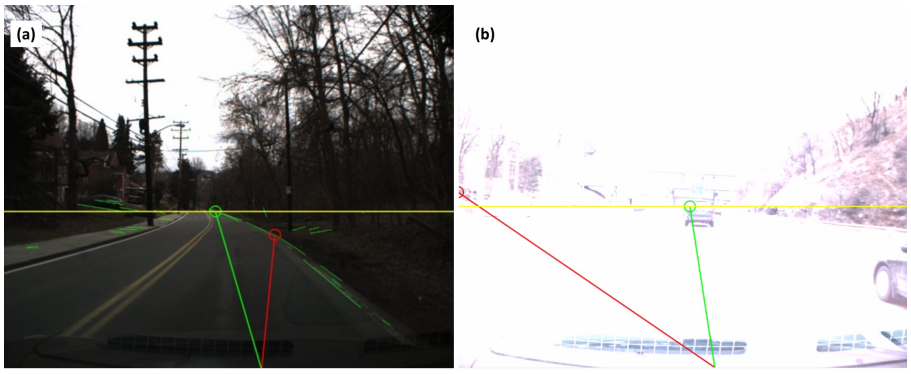

Figure 1: Sample images show the necessity of a vanishing point tracking for real-world, automotive applications. The red circle represents the vanishing points detected from the input images and the green circle represents the vanishing points tracked over frames. The yellow line represents the estimated horizon line. The existing, frame-by-frame vanishing point detection methods would fail when relevant image features are not present at the input images.

1 Introduction

This paper presents a simple, but effective method for detecting and tracking the vanishing point on a horizon appearing in a stream of urban scene images. In urban street scenes, such detecting and tracking would enable the obtaining of geometric cues of 3-dimensional structures. Given the image coordinates of the horizontal vanishing point, one could obtain, in particular, the information about the instantaneous driving direction of a roadway [5, 11, 14, 16, 17, 18, 20, 21], the information about the image regions for sampling the features of the drivable image regions [13, 15], the search direction of moving objects [12], and computational metrology through homography [19]. Advanced driving assistance systems or self-driving cars can exploit such information to detect neighboring moving objects and decide where to drive. Such information about roadway geometry can be obtained using active sensors (e.g., lidars with multi-horizontal planes or 3-dimensional lidar [23]), but, as an alternative, many researchers have studied the use of vision sensors, due to lower costs and flexible usages [1, 2, 5, 12].

A great deal of excellent work has been done in detecting vanishing points on perspective images of man-made environments; their performances are demonstrated on collections of images [3, 4, 10, 22]. Most of these methods, in voting on potential locations of vanishing points, use low-level image features such as spatial filter responses (e.g., Garbor filters) [15, 11, 18, 25] and geometric primitives (e.g., lines) [10, 19, 21, 22]. To find an optimal vote result, the methods use an iterative algorithm such as Expectation and Maximization (EM).

However, these frame-by-frame vanishing point detection methods may be impractical for real-time, automotive applications primarily because 1) they require intensive computation per frame and 2) they expect a presence of low-level image features.

In particular, it may take longer than a second simply to apply spatial filters to large parts of or the entire input image. Meanwhile, a vehicle drives a number of meters with no information about road geometry. Furthermore, these frame-by-frame methods would fail to detect the vanishing point appearing on over- and under-exposed images. Such images are acquired when a host-vehicle is emerging from tunnels or overpasses. Figure 1 (b) shows a sample image acquired when our vehicle emerges from a tunnel. When this happens, these methods would fail to continuously provide information about the vanishing point's location. Because of such a practical issue, some researchers developed Bayes filters to track the vanishing point's trajectory [15, 21]. In addition to the two aforementioned concerns, we have one of our own. In an earlier work [19], we demonstrated the ability to acquire, using a monocular camera sensor, the information of a vehicle's lateral motions as well as metrological information of the ground plane. To correctly compute metric information such as lateral distances of a vehicle to both boundaries of the host road-lane, it is critical to accurately estimate the angle between the road plane and the camera plane. To do this, we detect the vanishing point on the horizon to estimate the angle between two planes. But, because, image features relevant to detecting the vanishing point are missing in certain frames, our vanishing point detection fails to correctly locate the vanishing point, resulting in incorrect angle measurements and distance computations.

To address these practical concerns, we have developed a novel method of detecting and tracking the vanishing point on the horizon. In what follows, Section 2.1 details how we extract line segments from an input image and how we detect, using extracted line segments, the vanishing point on the horizon. Section 2.2 describes our implementation of Extended Kalman Filter (EKF) for tracking the detected vanishing point. Section 3 explains experiments conducted to demonstrate the effectiveness of the proposed algorithms and discusses the findings. Finally Section 4 lays out our conclusions and future work.

The contributions of this paper include 1) a method, based on line segments, for fast detection of vanishing points, 2) a novel vanishing point tracking algorithm based on a Bayes filter, and 3) empirical validations of the proposed work.

2 A Bayes Filter for Tracking a Vanishing Point on the Horizon

This section details our approach to the problem of detecting and tracking a vanishing point on a horizon, in particular, on perspective images of urban streets. A vanishing point on a perspective image is the intersection point of two parallel lines. In urban street scenes, as long as the image is under normal exposure, one can obtain plenty of parallel line pairs, pairs such as longitudinal lane-markings and building contour lines. Section 2.1 describes how we extract lines and, with them, detect vanishing points. The image coordinates of the vanishing points detected from individual frames may be temporally inconsistent because lines relevant to and important for vanishing point detection may not have been extracted. To smooth out the location of vanishing points over time, we develop an extended Kalman filter to track vanishing points. Section 2.2

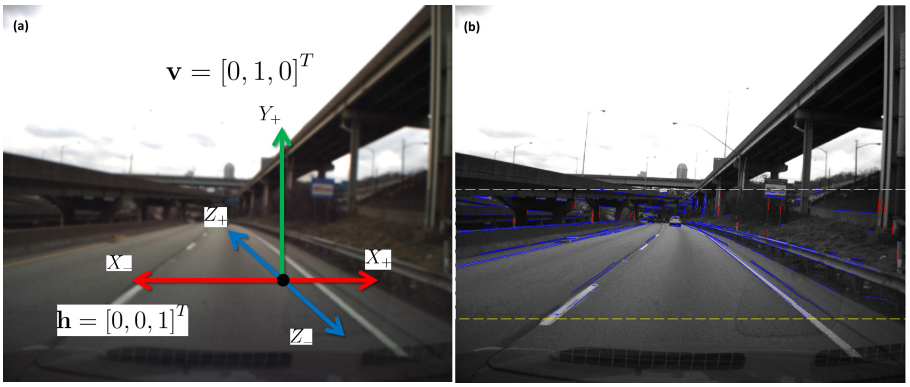

Figure 2: (a) A prior for the line classification. (b) An example of line detection and classification. The red (blue) lines are categorized into the vertical (horizontal) line group. The yellow, dashed rectangle represents a ROI for line extraction.

details the procedure and measurement model of our EKF implementation.

2.1 Vanishing Point Detection

Our algorithm detects, by using line segments, vanishing points appearing on a perspective image. In an urban scene image, one can extract numerous lines from urban structures, like man-made structures (e.g., buildings, bridges, overpasses, etc.) and traffic devices (e.g., Jersey barriers, lane-markings, curbs, etc.) To obtain these lines, we tried three line-extraction methods: Kahn's [9, 10], the probabilistic, and the standard Hough transform [6]. We found Kahn's method to work best in terms of the number of resulting lines and their geometric properties, such as lengths or representation fidelity to the patterns of low-level features. To implement Kahn's method, we first obtain Canny edges and run the connected component-grouping algorithm to produce a list of pixel blobs. For each pixel blob, we compute the eigenvalues and eigenvectors of the pixel coordinates' dispersion matrix. The eigenvector, $\mathbf{e}_1$, associated with the largest eigenvalue is used to represent the orientation of a line segment and its length, $\mathbf{l}_j = (\theta_j, \rho_j) = (atan2(\mathbf{e}_{1,2}, \mathbf{e}_{1,1}), \bar{x}\cos\theta + \bar{y}\sin\theta)$, where $\bar{x} = \frac{1}{n}\Sigma_k x_k, \bar{y} = \frac{1}{n}\Sigma_k y_k$. The two parameters, θ_j and ρ_j, are used to determine two end points, $\mathbf{p}_j^1 = \left[x_j^1, y_j^1\right]$ and $\mathbf{p}_j^2 = \left[x_j^2, y_j^2\right]$, of the line segment $\mathbf{l}_j$. Figure 2 (b) shows an example of line detection result.

Given a set of the extracted lines, $L = \{\mathbf{l}_j\}_{j=1,\ldots,|L|}$, we first categorize them into one of two groups: vertical L_V or horizontal L_H, $L = L_V \cup L_H$. We do this to use only a relevant subset of the extracted lines for detecting a particular (vertical or horizontal) vanishing point. For example, if vertical lines were used to find a horizontal vanishing point, the coordinates of the resulting vanishing point would be far from optimal. To set the criteria for this line categorization, we define two planes: $\mathbf{h} = [0, 0, 1]^T$ for a horizontal plane and $\mathbf{v} = [0, 1, 0]^T$ for a vertical plane in the camera coordinate. We do this because we assume that the horizontal (or vertical) vanishing points lie

at a horizontal (or vertical) plane at the front of our vehicle. Figure 2 (a) illustrates our assumption about these priors. We transform, the coordinates of the extracted line segments' two points into those of the camera coordinates, $\mathbf{p}_{cam} = \mathbf{K}^{-1}\mathbf{p}_{im}$, where $\mathbf{p}_{cam}$ is a point in the camera coordinates, $\mathbf{K}$ is the camera calibration matrix of intrinsic parameters, and $\mathbf{p}_{im}$ is a point in the image coordinates. We then compute the distance of a line segment, $\mathbf{l}_j = [a_j, b_j, c_j]^T$,[1] to the horizontal, $\mathbf{h}$, and the vertical plane, $\mathbf{v}$. We assign a line to either of two line groups based on the following:

$$\begin{aligned} L_V &\leftarrow \mathbf{l}_j, \quad \text{if } \mathbf{l}_j^T \cdot \mathbf{v} \leq \mathbf{l}_j^T \cdot \mathbf{h}, \\ L_H &\leftarrow \mathbf{l}_j, \quad \text{Otherwise} \end{aligned} \tag{1}$$

where $\mathbf{l}_j^T \cdot \mathbf{v} = \frac{[a_j,b_j,c_j]^T[0,1,0]}{\sqrt{a_j^2+b_j^2+c_j^2}}$. Figure 2 (b) shows an example of line classification result; vertical lines are depicted in red, horizontal lines in blue. Such line categorization results help us use a subgroup of the extracted lines relevant to computing the vertical or horizontal vanishing point. Our approach of using line segments to detect vanishing point is similar to some found in earlier work [10, 21, 22]. All uses line segments (or edges) to detect vanishing points. Our distinguishes itself in terms of line classification. Suttorp and Bucher's method relied on a heuristic, to cluster lines into left or right sets for vanishing point detection [21]; Tardif [22] used a J-linkage algorithm to group edges into the same clusters. In contrast, our method distinguishes horizontal line segments from vertical ones by setting priors about the ideal locations of vanishing points.

Given two sets of line groups (vertical and horizontal), we run RANSAC [6] to find the best estimation of a vanishing point. For each line pair randomly selected from the horizontal and vertical line groups, we first compute the cross-product of two lines, $\mathbf{vp}_{ij} = \mathbf{l}_i \times \mathbf{l}_j$, to find an intersection point. The intersection point found thus is used as a vanishing point candidate. We then claim the vanishing point candidate with the smallest number of outliers as the vanishing point for that line group. A line pair is regarded as an outlier if the angle between a vanishing point candidate and the vanishing point obtained from the line pair is greater than a pre-defined threshold (e.g., 5 degrees). We repeat this procedure until a vertical vanishing point is found and more than one horizontal vanishing point is obtained. The horizontal vanishing point with the smallest number of outliers is selected as the vanishing point on the horizon. Figure 3 shows sample results of vanishing point detection.

2.2 Vanishing Point Tracking

The previous section detailed how we detect vanishing points using line segments extracted from urban structures. Such frame-by-frame detection results, however, may be inconsistent over frames. This is because some image features (i.e., line segments) relevant to detecting vanishing points on the previous frame may not be available in the current frame. When this happens, any frame-by-frame, vanishing point detection

[1] Using two end-points of a line segment, we can represent a line segment in an implicit line equation, where, $a_j = y_j^1 - y_j^2, b_j = x_j^2 - x_j^1, c = x_j^1 y_j^2 - x_j^2 y_j^1$.

Figure 3: Some examples of vanishing point detection results. For most of testing images, our vanishing point detection worked well as good as tracking method. But it often failed to correctly identify the location of the horizontal vanishing point. For the last two images, our detection method found the locally optimal vanishing points (red circles) based on the lines extracted from those images. By contrast, our tracking method were able to find the globally optimal locations (green circles) of the vanishing points.

algorithm, including ours, fails to find an optimal location of the horizontal vanishing point. This results in incorrect information about roadway geometry [19].

To address such potential inconsistency, we develop a tracker to smooth out the trajectory of the vanishing point of interest. Our idea for tracking the vanishing point is to use some of the extracted line segments as measurements, thus enabling us to trace the trajectory of the vanishing point. To implement our idea, we developed an Extended Kalman Filter (EKF). Algorithm 1 describes the procedure of our vanishing point tracking method.

For our EKF model, we define the state as, $\mathbf{x}_k = [x_k, y_k]^T$, where x_k, y_k is the k step's camera coordinates of the vanishing point on the horizon. We initialize the state, $\mathbf{x}$ and its covariance matrix, $\mathbf{P}$ as:

$$\begin{aligned} \mathbf{x}_0 &= [\mathrm{IM}_{\mathrm{width}}/2/\mathrm{f_x}, \mathrm{IM}_{\mathrm{height}}/2/\mathrm{f_y}]^T, \\ \mathbf{P}_0 &= \begin{bmatrix} \left(\frac{x_{im}}{f_x}\right)^2 & 0 \\ 0 & \left(\frac{y_{im}}{f_y}\right)^2 \end{bmatrix} \end{aligned}$$

where x_{im} and y_{im} are our initial guesses about the uncertainty of the state in pixels, along the x- and y-axises, and f_x and f_y are focal lengths of the vision sensor we use. The initial values need to be scaled by focal lengths are required because the state is represented in the normalized coordinates.

Given an input image, our algorithm predicts the location of the vanishing points, $\hat{\mathbf{x}}_k^- = \mathbf{I}_2\hat{\mathbf{x}}_{k-1} + \mathbf{w}_{k-1}$, where $\mathbf{I}_2$ is 2×2 identity matrix and $\mathbf{w}_{k-1}$ is a 2×1 vector of process model's noise, normally distributed, $\mathbf{w}_k \sim N(0, \mathbf{Q})$.[2] While doing so, we neither define a motion model nor incorporate any information about ego-motion. We set the process noise as a constant, $\mathbf{Q}_{2\times2} = \mathrm{diag}(\sigma_Q^2)$, where $\sigma = \frac{x_{im}}{f_x}$.

[2]The $\hat{\mathbf{x}}$ represents an estimate and the superscript, $\hat{\mathbf{x}}^-$, indicates that it is a predicted value.

Algorithm 1 EKF for tracking the vanishing point.

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

Ensure: $\hat{\mathbf{x}}_k = [x_k, y_k]^T$, 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 $vp^h_x \leq \mathrm{IM}_{\mathrm{width}}$ **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: $\mathbf{P}_k = \mathbf{F}_{k-1}\mathbf{P}_{k-1}\mathbf{F}^T_{k-1} + \mathbf{Q}_{k-1}$
6: EKF: **Measurement Update**
7: **for all** $\mathbf{l}_j \in L$ **do**
8: $\quad \tilde{y}_j = z_j - h(\hat{\mathbf{x}}^-_k)$
9: $\quad \mathbf{S}_j = \mathbf{H}_j\mathbf{P}_j\mathbf{H}^T_j + \mathbf{R}_j$
10: $\quad \mathbf{K}_j = \mathbf{P}_j\mathbf{H}^T_j\mathbf{S}^{-1}_j$
11: $\quad$ Update the state estimate if $\tilde{y}_j \leq \tau$
12: $\quad \hat{\mathbf{x}}_k = \hat{\mathbf{x}}^-_k + \mathbf{K}_j\tilde{y}_j$
13: $\quad \mathbf{P}_j = (\mathbf{I}_2 - \mathbf{K}_j\mathbf{H}_j)\mathbf{P}_j$
14: **end for**

For the measurement update, we first change the representation of an extracted line segment, $\mathbf{l}_j$, as a pair of image coordinates of its mid-point and orientation, $\mathbf{l}_j = [\mathbf{m}_j, \theta_j]^T$, where $\theta_j \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$, $\mathbf{m}_j = [m_{j,x}, m_{j,y}]^T$. Note that the line segments we use as measurements for EKF are the same ones that used for detecting the vanishing point. Our approach is similar to Suttorp and Bucher's method [21], but both employ different measurement models. We then compute the residual, $\tilde{y}_j$, the difference between our expectation on an observation, $h(\hat{\mathbf{x}}^-_k)$ and an actual observation $z_j = \theta_j$.

We presume that if a selected line segment, $\mathbf{l}_j$, is aligning with the vanishing point of interest, $\hat{\mathbf{x}}_k = [x_k, y_k]^T$, the angle between the vanishing point and the orientation of the line should be zero (or very close to zero). Figure 5 illustrates the underlying idea of our measurement model that investigates the geometric relation between an extracted line and a vanishing point of interest. Based on this idea, we design a model of what we expect to observe, our observation model, as

$$h(\mathbf{x}_k) \quad = \quad \tan^{-1}\left(\frac{y_k - m_{j,y}}{x_k - m_{j,x}}\right) \tag{2}$$

To linearize this non-linear observation model, we take the first-order, partial derivative of $h(\mathbf{x}_k)$, with respect to the state, $\mathbf{x}_k$, to derive the Jacobian of the measurement model, $\mathbf{H}$.

$$\frac{\partial h(\mathbf{x}_k)}{\partial \mathbf{x}} = \left[\frac{-(y_k - m_{j,y})}{d^2}, \frac{(x_x - m_{j,x})}{d^2}\right] = \mathbf{H} \tag{3}$$

where $d^2 = \sqrt{(x_k - m_{j,x})^2 + (y_k - m_{j,y})^2}$. We set the measurement noise, $\mathbf{v}_k \sim$

Figure 4: A comparison of vanishing point locations by the frame-by-frame detection and by the EKF tracking.

$N(\mathbf{0}, \mathbf{R})$ and $\mathbf{R}_{1\times 1} = \sigma_R^2$, where $\sigma = 0.1$ radian. We then compute the innovation $\mathbf{S}_j$ and the Kalman gain $\mathbf{K}_j$ for the measurement update.

Before actually updating the state using these measurements, we treat individual line segments differently based on their lengths. This is because the shorter the length the higher the chance of the line being a noise measurement.[3] To implement this idea, we compute a weight of the line based on its length and heading difference, to update the measurement noise.

$$\mathbf{R} = R_{max} + \left(\frac{R_{min} - R_{max}}{l_{max} - l_{min}} \right) |\mathbf{l}_j| \tag{4}$$

where R_{max} (e.g., 10 degrees) and R_{min} (e.g., 1 degree) define the maximum and the minimum of heading difference in degree, and l_{max} (e.g., 500) and l_{min} (e.g., 20) define the maximum and minimum of observable line lengths in pixels, $|\mathbf{l}_j|$ is the length of the line. This equation ensures that we treats the longer line more importantly when updating the state and we only use lines of which heading differences are smaller than the threshold, τ.

In summary, the task of our EKF is to analyze line measurements to estimate the location of the vanishing point on the horizon. Figure 4 shows some example results that one can see the difference of the locations between the detected and the tracked vanishing points.[4]

We use the tracked vanishing point to compute the (pitch) angle between the camera plane and the ground plane. The underlying assumption is that the vanishing point along the horizon line is exactly mapped to the camera center if the road plane is flat and perpendicular to an image plane. From this assumption, we can derive the location of the vanishing point on the horizon line as [8]:

$$\mathbf{vp}_h^*(\phi, \theta, \psi) = \left[\frac{c\phi s\psi - s\phi s\theta c\psi}{c\theta c\psi}, \frac{-s\phi s\psi - c\phi s\theta c\psi}{c\theta c\psi} \right]^T \tag{5}$$

[3]Recall that we extract line segments from Canny's edge image where short edges may originate from artificial patterns, not from actual objects' contours.

[4]Some of the vanishing point tracking videos are available from, `http://www.cs.cmu.edu/~youngwoo/research.html`

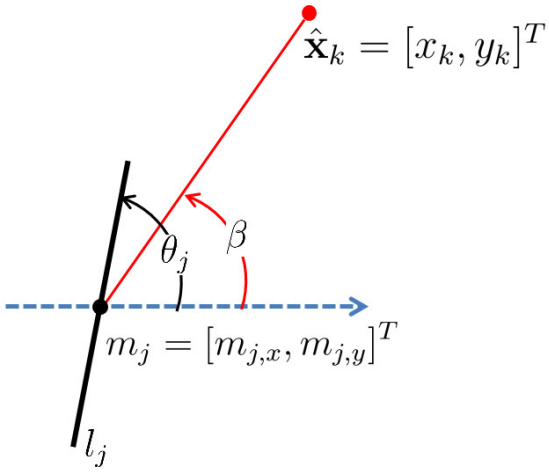

Figure 5: The line measurement model. The red circle represents the vanishing point, $\hat{\mathbf{x}}_k$, tracked until kth step. θ_j is the orientation of the jth line, l_j, and β is the orientation between the line's mid-point and the vanishing point. The orientation difference is the residual of our EKF model.

where ϕ, θ, ψ are yaw, pitch, and roll angle of the camera plane with respect to the ground plane and c and s for $\cos$ and $\sin$. Since we are interested in estimating the pitch angle, let us suppose that there is no vertical tilt and rolling (i.e., the yaw and the roll angles are zero). Then the above equation yields:

$$\mathbf{vp}_h^*(\phi = 0, \theta, \psi = 0) = \left[\frac{0}{c\theta}, -\frac{s\theta}{c\theta}\right] \tag{6}$$

Because we assume that there is neither yaw nor roll, we can compute the pitch angle by computing the difference between the y-coordinate of the vanishing point and that of the principal point of the camera as

$$\theta = \tan^{-1}\left(|p_y - vp_y|\right) \tag{7}$$

where p_y is the y coordinate of the principal point. Figure 6 shows our setup to verify the accuracy of our pitch angle estimation. Because no precise angle measurement exists between the two planes, we instead measure the distances between the camera and markers on the ground to evaluate the accuracy of the pitch angle computation. We found that the distance measurements have, on average, a sub-meter accuracy (i.e., less than 30cm).

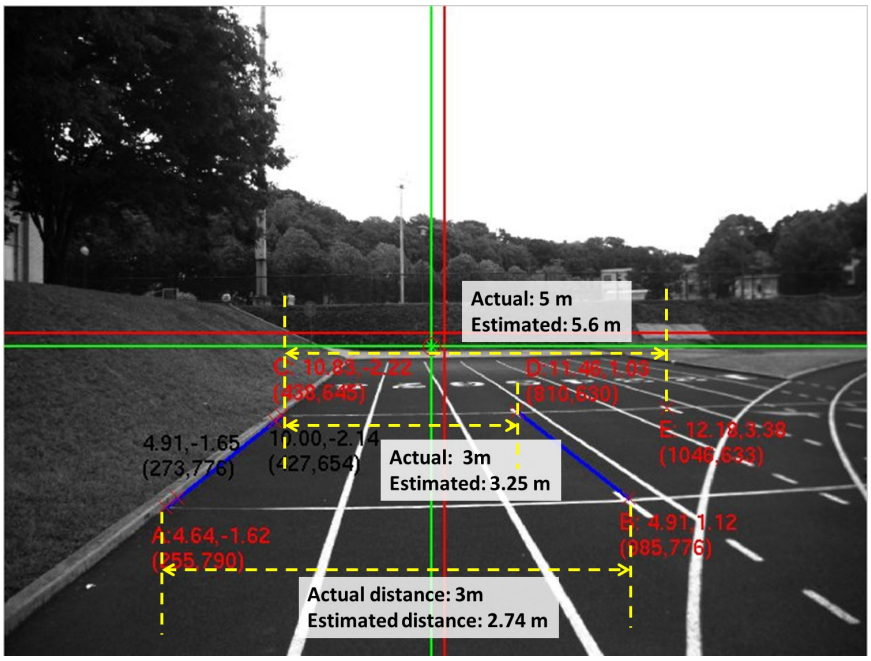

Figure 6: A setup for verifying the accuracy of our world-coordinate computation model. The intersection point of the two red lines represents the camera center and the intersection point of the two green lines represents a vanishing point computed from the two blue lines appearing on the ground.

3 Experiments

To evaluate the performance of our vanishing point detection and tracking algorithm, we drove our robotic car [24] on a route of inter-city highways, to collect some image data and the vehicle's ego-motion data. Our vehicle is equipped with a military-grade IMU which, in root-mean-square sense, the error of pitch angle estimation is 0.02 with GPS signals (with RTK corrections) or 0.06 degree with GPS outage, when driving more than one kilometer or for longer than one minute. The vision sensor installed on our vehicle is PointGrey's Flea3 Gigabit camera, which can acquire an image frame of 2448×2048, maximum resolution at 8Hz. While driving the route, we ran the proposed algorithms as well as the data (i.e., image and vehicle states) collector. We implemented the proposed methods in C++ and OpenCV that runs about 20Hz. The data collector automatically syncs the high-rate, ego-motion data (i.e., 100Hz) with the low-rate, image data (i.e., 8Hz). To estimate the camera's intrinsic parameters, we used a publicly, available toolbox for camera calibration[5] and define a rectangle for the line extraction ROI, $x_1 = 0$, $x_2 = \mathbf{I}_{width-1}$, $y_1 = 1300$ and $y_2 = 1800$. We empirically found that $R_{max} = 10$, $R_{min} = 1$, $l_{max} = 500$, and $l_{min} = 20$ worked best.

[5] http://www.vision.caltech.edu/bouguetj/calib_doc/

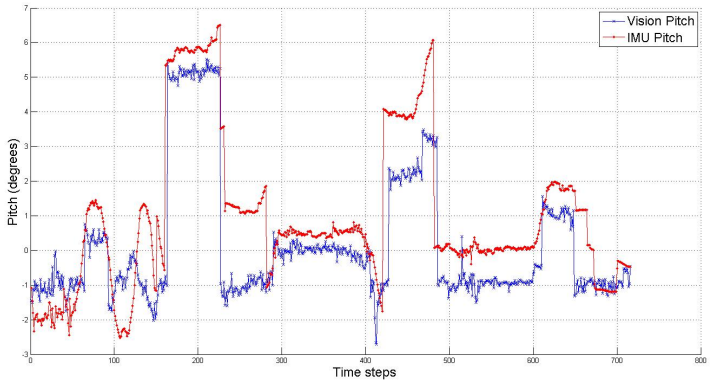

Figure 7: A comparison of the estimated pitch angles by an IMU and by the proposed method.

We evaluated quantitatively and qualitatively the performance of the presented vanishing point tracking method.

For the quantitative evaluation, we analyzed the accuracy of the pitch angles estimated from the vanishing point tracking. Figure 7 shows the comparison of the pitch angles measured by the IMU and estimated by a monocular vision sensor. Although the pitch angles estimated from our algorithm have some periods underestimate the true pitch angles, the two graphs have, at a macro-level, similar shapes where the blue curve follows the ups-and-downs of the red curve. The mean-square error is 2.0847 degrees.

For the qualitative evaluation, we analyzed how useful the output of the tracked vanishing point is in approximating the driving direction of a road way. Figure 8 shows some example results that, within a certain range, the driving directions of roads can be linearly (or instantaneously) approximated by linking the locations of the tracking vanishing point to the center of the image bottom (i.e., the image coordinates our camera is projected on).

4 Conclusions and Future Work

This paper has presented a novel method of detecting vanishing points and of tracking a vanishing point on the horizon. To detect vanishing points, we extracted lines and applied RANSAC to the locally optimal vanishing point from a given input image. Occasionally, however, our method failed to detect the vanishing point because relevant image features were unavailable. Our previous computer vision application for autonomous driving required metric computation to accurately measure the vehicle's lateral motion. To obtain this measurement, one needs an accurate measurement of the angle between the camera and the ground planes. To compute this angle, we used the detected vanishing point. Thus, when the vanishing point location was inac-

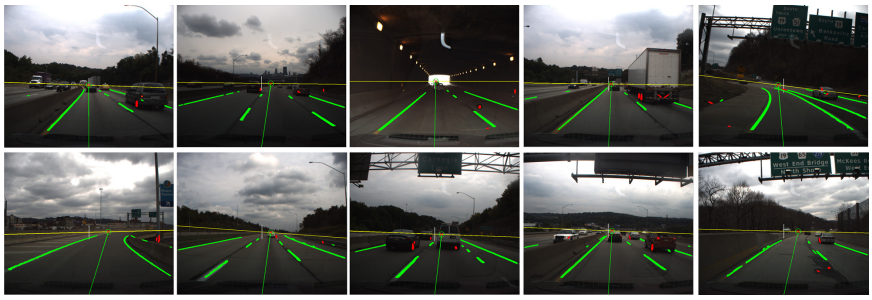

Figure 8: This figure shows the idea of using results of vanishing point detection to approximate the driving direction of a roadway. We used such approximated driving directions to remove false-positive lane-marking detections [20]. The green blobs are the final outputs of lane-marking detection and the red blobs are the false-positive lane-marking detections that are removed from the final results.

curately located, it led to an imprecise measurement of the vehicle's lateral motions. We addressed this jumpy trajectory of the vanishing point by tracking it using EKF.

As future work, we would like to determine the limits of our algorithms and so continue testing it against various driving environments. In addition, we would like to study the relation of ego-vehicle's motion between in the world coordinates and image coordinates and develop a motion model to enhance the performance of our tracking.

5 Acknowledgments

The author would like to thank the team members of the General Motors-Carnegie Mellon University Autonomous Driving Collaborative Research Laboratory (AD-CRL) for their effort.

References

[1] Nicholas Apostoloff and Alexander Zelinsky, Vision in and out of vehicles: integrated driver and road scene monitoring, *The International Journal of Robotics Research*, 23(4-5): 513-538, 2004.

[2] Massimo Bertozzi, Alberto Broggi, Alessandro Coati, and Rean Isabella Fedriga, A 13,000 kim intercontinental trip with driverless vehicles: the VIAC experiment, *IEEE Intelligent Transportation Sysstem Magazine*, 5(1): 28-41, 2013.

[3] James M. Coughlan and A.L. Yuille, Manhattan world: compass direction from a single image by bayesian inference, In *Proceedings of IEEE International Conference on Computer Vision* (ICCV-99), pp. 941-947, 1999.

[4] A. Criminisi, I. Reid, and A. Zisserman, Single view metrology, *International Journal of Computer Vision*, 40(2): 123-148, 2000.

[5] Ernst D. Dickmanns, *Dynamic Vision for Perception and Control of Motion*, Springer, 2007.

[6] David A. Forsyth and Jean Ponce, *Computer Vision: A Modern Approach*, Prentice Hall, 2002.

[7] Richard Hartley and Andrew Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2003.

[8] Myung Hwangbo, Vision-based navigation for a small fixed-wing airplane in urban environment, *Tech Report CMU-RI-TR-12-11*, PhD Thesis, The Robotics Institute, Carnegie Mellon University, 2012.

[9] P. Kahn and L. Kitchen and E.M. Riseman, A fast line finder for vision-guided robot navigation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(11): 1098-1102, 1990.

[10] Jana Kosecka and Wei Zhang, Video compass, In *Proceedings of European Conference on Computer Vision* (ECCV-02), pp. 476-490, 2002.

[11] Hui Kong, Jean-Yves Audibert, and Jean Ponce, Vanishing point detection for road detection, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (CVPR-09), pp. 96-103, 2009.

[12] Joel C. McCall and Mohan M. Trivedi, Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation, *IEEE Transactions on Intelligent Transportation Systems*, 7(1): 20-37, 2006.

[13] Ondrej Miksik, Petr Petyovsky, Ludek Zalud and Pavel Jura, Robust detection of shady and highlighted roads for monocular camera based navigation of UGV, In *Proceedings of IEEE International Conference on Intelligent Robots and Systems* (IROS-11), pp. 64-71, 2011.

[14] Ondrej Miksik, Rapid vanishing point estimation for general road detection, In *Proceedings of IEEE International Conference on Robotics and Automation* (ICRA-12), pp. 4844-4849, 2012.

[15] Peyman Moghadam and Jun Feng Dong, Road direction detection based on vanishing-point tracking, In *Proceedings of IEEE International Conference on Intelligent Robots and Systems* (IROS-12), pp. 1553-1560, 2012.

[16] Macros Nieto, Luis Salgado, Fernando Jaureguizar, and Julian Cabrera, Stabilization of inverse perspective mapping images based on robust vanishing point estimation, In *Proceedings of IEEE Intelligent Vehicles Symposium* (IV-07), pp. 315-320, 2007.

[17] Marcos Nieto, Jon Arrospide Laborda, and Luis Salgado, Road environment modeling using robust perspective analysis and recursive Bayesian segmentation, *Machine Vision and Applications*, 22:927-945, 2011.

[18] Christopher Rasmussen, Grouping dominant orientations for ill-structured road following, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (CVPR-04), pp. 470-477, 2004.

[19] Young-Woo Seo and Raj Rajkumar, Use of a monocular camera to analyze a ground vehicle's lateral movements for reliable autonomous city driving, In *Proceedings of IEEE IROS Workshop on Planning, Perception and Navigation for Intelligent Vehicles* (PPNIV-2013), pp. 197-203, 2013.

[20] Young-Woo Seo and Raj Rajkumar, Utilizing instantaneous driving direction for enhancing lane-marking detection, In *Proceedings of the IEEE Intelligent Vehicles Symposium* (IV-2014), to appear, 2014.

[21] Thorsten Suttorp and Thomas Bucher, Robust vanishing point estimation for driver assistance, In *Proceedings of IEEE Intelligent Transportation Systems Conference* (ITSC-06), pp. 1550-1555, 2006.

[22] Jean-Philippe Tardif, Non-iterative approach for fast and accurate vanishing point detection, In *Proceedings of IEEE International Conference on Computer Vision* (ICCV-09), pp. 1250-1257, 2009.

[23] Chris Urmson, The self-driving car logs more miles on new wheels, `http://googleblog.blogspot.com/2012/08/the-self-driving-car-logs-more-miles-on.html`, 2012.

[24] Junqing Wei, Jarrod Snider, Junsung Kim, John Dolan, Raj Rajkumar, and Bakhtiar Litkouhi, Towards a viable autonomous driving research platform, In *Proceedings of IEEE Intelligent Vehicles Symposium* (IV-13), 2013.

[25] Qi Wu, Wende Zhang, and B.V.K Vijaya Kumar, Example-based clear path detection assisted by vanishing point estimation, In *Proceedings of IEEE International Conference on Robotics and Automation* (ICRA-11), pp. 1615-1620, 2011.