Detection and Tracking of Boundary of Unmarked Roads

Young-Woo Seo
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213 USA
Email: young-woo.seo@ri.cmu.edu

Ragunathan (Raj) Rajkumar
Dept of Electrical Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213 USA
Email: raj@ece.cmu.edu

***Abstract*— This paper presents a new method of detecting and tracking the boundaries of drivable regions in road without road-markings. As unmarked roads connect residential places to public roads, the capability of autonomously driving on such a roadway is important to truly realize self-driving cars in daily driving scenarios. To detect the left and right boundaries of drivable regions, our method first examines the image region at the front of ego-vehicle and then uses the appearance information of that region to identify the boundary of the drivable region from input images. Due to variation in the image acquisition condition, the image features necessary for boundary detection may not be present. When this happens, a boundary detection algorithm working frame-by-frame basis would fail to successfully detect the boundaries. To effectively handle these cases, our method tracks, using a Bayes filter, the detected boundaries over frames. Experiments using real-world videos show promising results.**

I. Introduction

This study aims at developing a computer vision algorithm that detects and tracks the boundaries of drivable regions appearing on input images. The roads we are particularly interested in are paved roads, but no road-markings (e.g., lane-markings, stop-lines, etc.) are painted, like ones connecting public roads to residential areas. Figure 1 shows four examples of such roads. To make self-driving cars truly useful, they should be capable of driving on such roads to demonstrate autonomous driving maneuvers from a house, through public roads, to a destination. This work reports the progress of our effort that develops the perception part of such a driving capability – detecting and tracking of drivable regions' boundaries.

We assume that, in a given image, the appearance (e.g., color and texture) of drivable regions would be, with a minor variation, homogeneous. To understand the characteristics of the drivable regions' appearance, we analyze the image sub-region at the front of ego-vehicle. The blue rectangles in Figure 1 show examples of such a sampling area. The idea of analyzing the image region around ego-vehicle to understand the characteristics of drivable regions has been exploited in many outdoor navigation work [1], [2], [3], [4], [5], [6]. However, the actual methods of identifying road-drivable regions are different from one another, e.g.) utilizing lidar scans to define close-range drivable regions [2], using a stereo camera to define road plane [3], and using the results of vanishing point detection to guide the sampling region [4]. They are all similar to each other in that they analyzed the color in multi-channels (e.g., Hue-Saturation-Intensity) of the sampling area to extract color patterns of drivable regions. By contrast, our method does not learn the color model of drivable regions because 1) analyzing images in multiple color channels requires more resources (e.g., computational time and memory space) and 2) learning the color model is often unreliable in terms of the performance of a color classification. Instead our method transforms an input image into a bird-eye view image and uses an intensity thresholding to identify the boundary of drivable regions. Alon et al. also used an inverse perspective image to identify the left and right boundaries of off-roads [7].

Fig. 1: Sample images of road without road-markings and the outputs of the proposed algorithm. A self-driving car should be capable of driving on roads like the ones appearing on these images. To do so, it is necessary to identify the boundaries of drivable regions. Red lines represent the outputs of our boundary detection algorithm and green lines depict the outputs of our boundary tracking algorithm.

The performance of image-based algorithms is sensitive to the variation of the relevant image features' presence. For example, if the values of image features such as color or intensity, are drastically changed due to the changes in an image acquisition process, frame-by-frame detection algorithms would fail to provide the outputs consistent over frames. To avoid such inconsistent outputs, we develop a Bayes filter to track the detected boundaries. To this end, we use Dickmanns and Mysliwetz's seminal work, the clothoid model of road shape [8] to model the detected boundary and develop an unscented Kalman filter (UKF) [9] to implement to track the detected boundaries.

Most work of the drivable region boundary tracking begins with a boundary detection. For the boundary tracking, they used the results of boundary detection as measurements. The majority of boundary tracking work employed the particle filter [10], [11], [12], [13], [14]. To the best of our knowledge, the clothoid (or Euler spiral) is the most widely used model for delineating road geometric shape. Some works including ours used the clothoid to model road shape and approximated actual shapes using cubic polynomials. Using this clothoid model, some works aimed at tracking the boundaries of rural roads [10] and unpaved roads [11], but most of them targeted to track the boundaries of paved roads with road-markings [12], [13], [14].

In what follows, we detail how our method detects the boundaries of roads paved, but no road-markings painted in Section II-A. And then we describe how our method, using an unscented Kalman filter (UKF), tracks the detected boundaries over frames in Section II-B. Section III discusses the finding of the experimental results and Section IV summarizes the findings.

II. Identification of Drivable Region Boundaries of Roads without Road-Painting

This section details how our method 1) detects the left and right boundaries of roads with no-road-paintings and 2) tracks the detected boundaries over frames.

To detect the boundaries of drivable regions, we first convert an input image into a bird-eye view image (or an inverse perspective image). We do this to reduce the computational cost for image processing[1] and to remove any distortion caused by perspective image acquisition. From an inverse perspective image, our method analyzes the image region at the front of our vehicle to understand the characteristics of drivable regions' appearance. In particular, we determine two intensity threshold values to identify the drivable region. Given the results of such a simple, but effective image analysis, we identify multiple of road-boundary points from the left and right side of ego-vehicle. The detected boundary points are then used as measurements for the boundary tracker. We use the clothoid model to model the boundary of drivable regions and develop an UKF to track the detected boundaries.

A. Detection of Drivable Region Boundaries

To detect the boundary of road, we first transform an input image into an inverse perspective image [15] and then analyze a rectangular image region at the front of ego-vehicle to understand the characteristics of drivable regions' appearance. The red rectangle at Figure 2 (b) shows an example of such a sample image region. This red rectangle corresponds to the blue rectangle at Figure 2 (a). We do such a sampling because we assume that drivable regions appearing on an input image have similar appearance (e.g., intensity). Figure 2 (b) shows an example of an inverse perspective image that converts the input image shown at Figure 2 (a). This happens through the transformation, $P_G = T_I^G P_I$, where P_G is a point of an inverse perspective image, T_I^G is the inverse perspective transformation matrix, P_I is a point on the input image.

[1]The dimension of an inverse perspective image (i.e., 627×711) is smaller than that of the original image (i.e., 2448×2048).

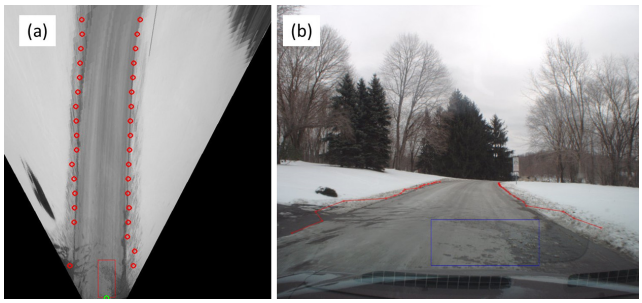

Fig. 3: Examples of the boundary points. (a) The red circles represents the detected boundary points on an inverse perspective image. (b) The boundary points detected from an inverse perspective image are projected onto an input, perspective image.

Using pixels in the sample region, we analyze the intensity of drivable regions and compute the statistics (i.e., mean, μ_I and standard deviation, σ_I) of intensity to choose two intensity thresholds: $\mu_I - 3\sigma_I$ and $\mu_I + 3\sigma_I$. We assume that the intensity of drivable regions follows a Gaussian distribution, $\mathbf{x} \sim N(\mu, \sigma)$ and choose all the pixels within 3σ (intensity values similar to 99% of the sampling area). We apply such an intensity thresholding to the input image and perform a connected-component analysis to the results of the thresholding, to identify a drivable region. Figure 2 (c) shows the results of these steps. As the thresholding is not a fine-tuned technique, the results of the thresholding is not optimal in that this method does not return a convex polygon of drivable region. However, as shown in Figure 2 (c), executing grouping based on pixels' connectivity results in most of the pixels belonging to the drivable region into the same pixel group. This technique is fairly simple, but works effectively in that the pixel group with the largest pixels always corresponds to the drivable region of an input image. Figure 2 (c), for instance, shows that our thresholding method returned a pixel group in light blue color at the right front of our vehicle. Although this pixel group has a couple of elliptical holes in it, the edges of the pixel blob correspond to the boundary of drivable regions. Once we find a pixel group corresponding to the drivable region (i.e., the pixel group with largest number of pixels), we scan through the pixel group, investigate the n number of rows, and pick two end of points of each investigating row as the left and the right boundary points. This results in $2n$ boundary points, $\mathbf{B} = \left\{\mathbf{b}_j^L, \mathbf{b}_j^R\right\}_{j=1,\ldots,n}$, where $\mathbf{b}_j^{L,R} \in R^2$ is a two-dimensional vector containing the pixel coordinates, $\mathbf{b}_j^L\left(\mathbf{b}_j^R\right) = [x_j^L, y_j^L]^T\left([x_j^R, y_j^R]^T\right)$. Figure 3 (a) shows the examples of these detected boundary points in an inverse perspective image and its perspective image (b).

B. Tracking of Drivable Region Boundaries

The previous section describes how our boundary detection method works. Our boundary detector analyzes a rectangular image sub-region at the front of our car on an in-

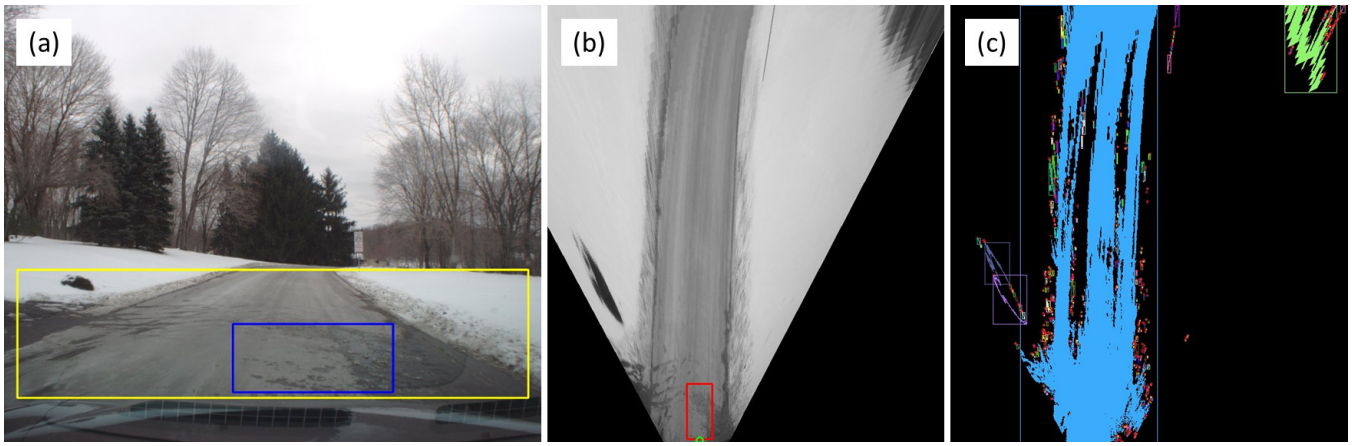

Fig. 2: (a) A part of an input image (i.e., the yellow rectangle area) is transformed into a bird-eye view image (or an inverse perspective image) (b) to remove perspective effects and to facilitate image processing. (c) Our algorithm analyzes the image region at the front of our vehicle and detects drivable-region by applying intensity-thresholding.

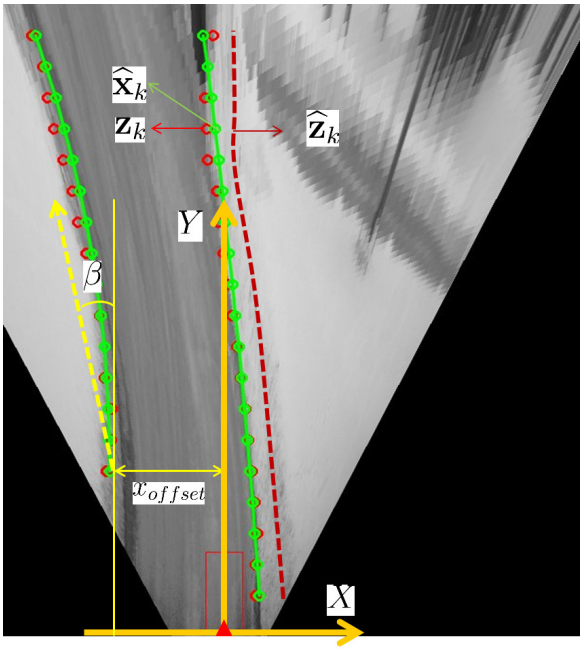

Fig. 4: The road-shape model and the measurement model.

verse perspective image and applies an intensity thresholding to identify drivable image region. A connected-component grouping is applied to the result image of intensity thresholding and the biggest pixel group is chosen as the drivable image region. This detection method results in $2n$ boundary points, $\mathbf{B} = \left\{\mathbf{b}_j^L, \mathbf{b}_j^R\right\}_{j=1,\ldots,n}$. This section details how our boundary tracker models, using these boundary points, the left and the right boundaries and track, using a Bayes filter, these boundaries over time.

We use the clothoid to model roadway's geometric shape [8]. A clothoid defines a function of road curvature given road (longitudinal) length, $\kappa(l) = c_0 + c_1 l = \int_0^l \kappa(\tau) d\tau$, where c_0 is the initial curvature and c_1 is the curvature derivative. Suppose that the starting point of a clothoid is (x_0, y_0), where the curvilinear coordinate, $l = 0$, then each coordinate of the curvature function is in fact defined as

$$x(l) = x_0 + \int_0^l \sin\theta(\tau) d\tau \tag{1}$$

$$y(l) = y_0 + \int_0^l \cos\theta(\tau) d\tau \tag{2}$$

To numerically evaluate these functions, it is required to approximate them using the Taylor expansions of $\sin\theta$ and $\cos\theta$. The first-order approximation of the clothoid equations is as

$$x = \frac{1}{6} c_1 y^3 + \frac{1}{2} c_0 y^2 + \beta y + x_{offset} \tag{3}$$

where x_{offset} is the lateral offset between the boundary and the vehicle (or boundaries of vehicle body), β is the heading angle with respect to the vehicle's driving direction, c_0 is the curvature, and c_1 is the curvature rate. Figure 4 illustrates this model. We use this equation to model the boundary of drivable region appearing on an image and define the state of our boundary tracker as:

$$\mathbf{x} = [x_{offset}, \beta, c_0, c_1]^T \tag{4}$$

Algorithm 1 describes the procedure how our method tracks drivable region boundary. Our tracker takes, as input, ego-vehicle's speed and the outputs of the boundary detection methods described in Section II-A.

We initialize the state and its covariance matrix as

$$\mathbf{x}_0 = [100, 0, 0, 0]^T \tag{5}$$

$$\mathbf{P}_0 = diag([10^2, 0.05^2, 0.01^2, 0.001^2]^T) \tag{6}$$

Our initialization assumes that the boundary is initially straight and located from 100 pixels, laterally away from ego-vehicle. We empirically set those values for initializing $\mathbf{P}$ and found that they worked well.

An UKF uses an unscented transformation to model a non-linear dynamics [9]. This often works better than that of Extended Kalman filter, which uses a linearization technique to approximate a non-linear dynamics using Jacobian. Because

Algorithm 1 UKF for tracking the drivable regions' boundaries.

Input: v_k, ego-vehicle's speed and
$2n$ boundary points, $\mathbf{B} = \left\{\mathbf{b}_j^L, \mathbf{b}_j^R\right\}_{j=1,\ldots,n}$,

Output: $\hat{\mathbf{x}}_k = [v_{offset}, \beta, c_0, c_1]^T$, an estimate of the road-shape

For the left (and the right) boundary, do the following:

1: Given a previous estimate, $\hat{\mathbf{x}}_{k-1}$ and $\mathbf{P}_{k-1}$, compute $2m+1$ sigma points and their weights, $(\chi_j, w_j) \leftarrow (\hat{\mathbf{x}}_{k-1}, \mathbf{P}_{k-1}, \kappa)$
2: Predict a state and its error covariance, $(\hat{\mathbf{x}}_k^-, \mathbf{P}_k^-) = \mathrm{UnscentTransform}(f(\chi_i, w_i), \mathbf{Q})$
3: Predict a measurement and its error covariance, $(\hat{\mathbf{z}}_k, \mathbf{P}_z) = \mathrm{UnscentTransform}(h(\chi_i, w_i), \mathbf{R})$
4: Compute Kalman gain, $K_k = \mathbf{P}_{xz}\mathbf{P}_z^{-1}$, where, $\mathbf{P}_{xz} = \sum_{i=1}^{2m+1} w_i \left\{f(\chi_i) - \hat{\mathbf{x}}_k^-\right\}\left\{h(\chi_i) - \hat{\mathbf{z}}_k\right\}^T$
5: **for all** $\left\{\mathbf{b}_j^L\ (\mathbf{b}_j^R)\right\}_{j=1,\ldots,n} \in \mathbf{B}$ **do**
6: $\quad \hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + K_j(\mathbf{z}_j - \hat{\mathbf{z}}_j)$
7: $\quad \mathbf{P}_k = \mathbf{P}_k^- - K_j\mathbf{P}_j K_j^T$
8: **end for**

of Jacobian, it may cause a divergence problem [16]. At the step 1, the unscented transformation begins with generating $2m+1$ sigma points,

$$\chi_1 = \mathbf{x}_k, \chi_{i+1} = \mathbf{x}_k + \mathbf{u}_i, \chi_{i+m+1} = \mathbf{x}_k - \mathbf{u}_i, \quad (7)$$

$$w_1 = \frac{\kappa}{m+\kappa}, w_{i+1} = \frac{1}{2(m+\kappa)}, w_{i+m+1} = \frac{1}{2(m+\kappa)} \quad (8)$$

where m is the dimension of the state, for our case, $m = 4$, $\mathbf{u}_i$ is a row vector from $\mathbf{U}^T\mathbf{U} = (m+\kappa)\mathbf{P}_k$, $\kappa = 3 - m$ is a constant, and $\sum_{i=0}^{2m} w_i = 1$.

For the prediction steps 2 and 3, these sigma points sampled around the current state, $\hat{\mathbf{x}}_k$ are used to compute the mean and covariance of the process model, $f(\mathbf{x})$, to approximate the non-linear process model.

$$\hat{\mathbf{x}}_k^- = \sum_{i=1}^{2m+1} w_i f(\chi_i) \quad (9)$$

$$\mathbf{P}_k^- = \sum_{i=1}^{2m+1} w_i \left\{f(\chi_i) - \hat{\mathbf{x}}_k\right\}\left\{f(\chi_i) - \hat{\mathbf{x}}_k\right\}^T \quad (10)$$

where $\hat{\mathbf{x}}_k^-$ and $\mathbf{P}_k^-$ are the predicted state and its covariance. Using such an unscented transformation, our process model, $f(\mathbf{x}_k)$ predicts how the shape of a road is changed with an assumption that the acceleration and yaw rate of ego-vehicle are constant during each time step, Δt.

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k) + \mathbf{q}_k = f(\chi_i, w_i)_{i=1,\ldots,2m+1} + \mathbf{q}_k \quad (11)$$

$$= \begin{bmatrix} 1 & v\Delta t & \frac{1}{2}(v\Delta t)^2 & \frac{1}{6}(v\Delta t)^3 \\ 0 & 1 & v\Delta t & \frac{1}{2}(v\Delta t)^2 \\ 0 & 0 & 1 & v\Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_k + \mathbf{q}_k$$

where v is a speed of ego-vehicle and $\mathbf{q}_k$ is noise of the process model, $\mathbf{q}_k \sim N(\mathbf{0}, \mathbf{Q})$.

The measurement model, $h(\hat{\mathbf{x}}_k^-) = \hat{\mathbf{z}}_k$ also uses the unscented transformation to predict the expected measurements at a given state. The measurement model is defined as

$$\hat{\mathbf{z}}_k = h(\hat{\mathbf{x}}_k^{-1}; y) + \mathbf{r}_k = h(\chi_i, w_i)_{i=1,\ldots,2m+1} + \mathbf{r}_k$$

$$= \frac{1}{6}c_1 y^3 + \frac{1}{2}c_0 y^2 + \beta y + x_{offset} \quad (12)$$

where $\mathbf{r}_k$ is the noise of the measurement model, $\mathbf{r}_k \sim N(\mathbf{0}, \mathbf{R})$. Figure 4 illustrates the measurement model, where a crimson, dashed line at the right is a boundary line based on the expected measurements.

The next step, the step 4 is to compute a Kalman gain that is used to determine how importantly a measurement is used to estimate the state. When each boundary point is used to estimate the state, the last step updates the state as much as the Kalman gain using the innovation (or residual), $(\mathbf{z}_k - \hat{\mathbf{z}}_k)$, the difference between an actual measurement and a predicted measurement.

Figure 5 shows a sequence of detection and tracking outputs where the benefit of tracking is demonstrated. Some of the detection outputs incorrectly picked up the boundary locations whereas the tracking outputs correctly delineated the boundary of drivable regions on roads without road-paintings.

III. Experiments

This section details the settings and results of the experiments we carried out to evaluate performance of the proposed algorithm. The goal of this work is to develop computer vision algorithms that accurately detect and reliably track the boundaries of drivable regions of the roads with no road-markings. We first detail the experimental setup in Section III-A and then discuss the experimental results in Section III-B.

A. Experimental Settings

To collect video data, we drove our robotic vehicle [17] on multiple routes of sub-urban regions around Pittsburgh. These video data were obtained under various weather and illumination conditions. One video was recorded in wither with snow accumulation in the background, another were obtained in spring, under fairly gentle illumination conditions, and the last was recorded on an evening in summer.

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. For a faster, real-time processing, we rescaled the original resolution into half and used a predefined ROI, $x_1 = 0$, $x_2 = \mathbf{I}_{width-1}$, $y_1 = 1300$ and $y_2 = 1800$ for the inverse-perspective transformation.[2]

We implemented the proposed algorithms in C++ with OpenCV libraries. Our implementation of boundary detection and tracking ran about 10 Hz on a 2.7 GHz Pentium V. For the UKF, we used the Bayes++ package[3] to implement our boundary tracker and initialized the state and its error (or covariance) matrix, $\mathbf{x}_0 = [x_{offset}, \beta, c_0, c_1]^T = [100, 0, 0, 0]^T$ and $\mathbf{P}_0 = diag\left([10^2, 0.05^2, 0.01^2, 0.001^2]\right)$,

[2]These y-values are in the original resolution. If the image is scaled to a half of the original, these y-values are scaled as well.

[3]`http://bayesclasses.sourceforge.net/Bayes++.html`

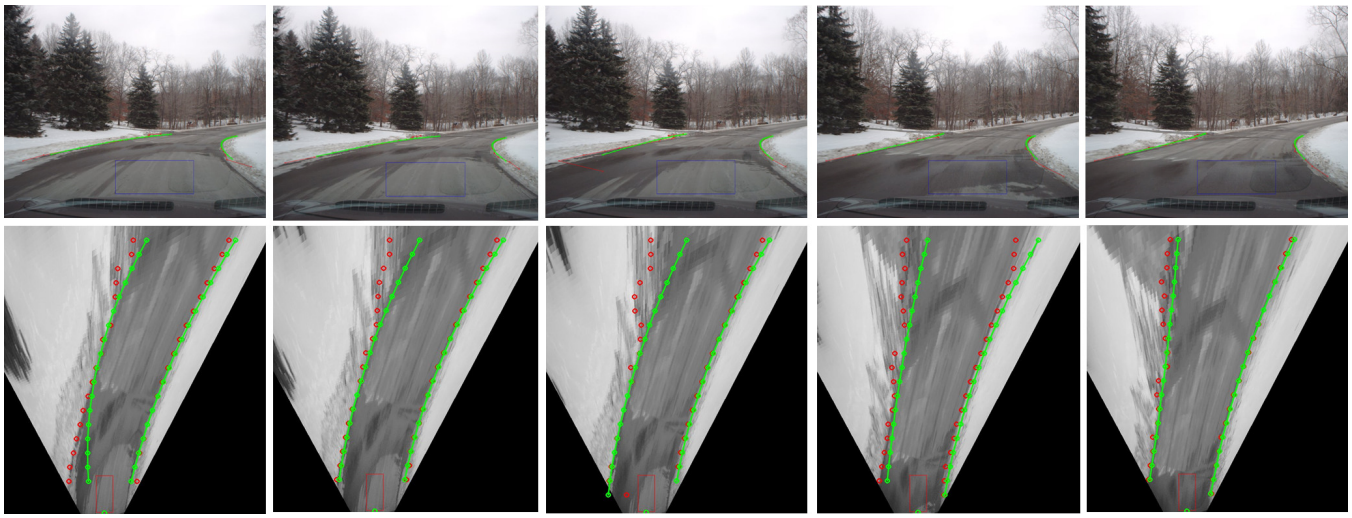

Fig. 5: An example sequence of drivable-regions' boundary detection and tracking.

where the values of x is in pixels, β is in radian, c_0, and c_1 are dimensionless curvature. In addition, the noises of the process model, $\mathbf{Q} = diag\left([10^2, 0.05^2, 0.01^2, 0.001^2]\right)$ and the measurement model, $\mathbf{R} = \left[3^2\right]$.

B. Experimental Results

We evaluate resulting boundary delineation in terms of accuracy of matching between output and ground truth pixels. To the best of our knowledge, no image data is available on identifying boundaries of unmarked road that we could use for comparison. Hence, we had to manually delineate the image data used for the evaluation.[4]

To evaluate our results at a pixel-to-pixel level, we utilized the method from evaluating performance of object boundary detection [18]. Similar to [18], we regard the extraction of boundaries as a classification problem of identifying boundary pixels and of applying the precision-recall metrics using manually labeled boundaries as ground truth. Precision is manifested in the fraction of outputs that are true positives; recall is the fraction of correct outputs over true positives. To evaluate the performance using these metrics, it is necessary to solve a correspondence problem that determines which output pixels are used to detect which true positive pixels. While resolving such a correspondence problem, we must carefully consider a localization error that accounts for the (Euclidean) distance between an output pixel and a ground truth pixel. Indeed, localization errors are present even in the ground truth images as well. For resolving the correspondence between output pixels and ground truth pixels, we utilized the Berkeley Segmentation Engine's [5] performance evaluation scripts. These scripts solve, using Goldberg's CSA package, the correspondence problem as a minimum cost bipartite assignment problem. Table I shows the performance difference between the detection and tracking of drivable regions' boundary, where the F-measure is computed by $f = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$. The performance of our detection algorithm showed a reasonable performance. As expected, the tracking indeed improved our system of identifying the boundaries of unmarked roads 11%.

	F-measure	Precision	Recall
Detection	0.70	0.63	0.78
Tracking	0.83	0.78	0.89

TABLE I: A precision-recall metric comparison.

Figure 6 shows some example outputs of our boundary detection and tracking algorithms. The first two rows show some of the correctly tracked boundary outputs and the last row show some example outputs that boundaries were incorrectly identified. Because our boundary detector works based on an intensity-thresholding, it occasionally failed to correctly pick up the boundary pixels. Despite of the detector's failures, the tracker was able to smooth out inconsistent outputs from the detector. However, our algorithms failed to track the boundaries 1) when our vehicle turned at intersections (e.g., the first image at the last row in Figure 6), 2) when the intensities of ground are highly uneven (e.g., the images between the second and the fourth at the last row), and 3) when the curvature of roads is high (e.g., the last image).

IV. Conclusions and Future Work

This paper presented our methods of detecting and tracking the boundaries of drivable regions of road with no road-markings. To detect the left and right boundaries of drivable regions, our method examines the image region at the front of ego-vehicle and uses the analyzed appearance information of that region to identify the boundary of the drivable region from input images. Due to variation in the image acquisition condition, the image features necessary for boundary detection might not be present. When this happens, a boundary detection algorithm working based on frame-by-frame would fail to successfully detect the boundaries. To prevent this, our method tracks, using an Unscented Kalman

[4]Some video clips about the results are available from `http://www.cs.cmu.edu/~youngwoo/research.html`

[5]The BSE and related information are available at `http://www.cs.berkeley.edu/~fowlkes/BSE/`

Fig. 6: Example outputs of boundary detection and tracking.

filter, the detected boundaries over frames. Experimental results using real-world videos show promising results.

Our boundary tracking algorithm failed to track the boundary when ego-vehicle turned at intersection, drove on slanted road or road with high-curvature road. To improve the performance our algorithms on these cases, we would like to incorporate the vehicle motion information from an IMU and the information about the geometric shape of upcoming road from a map of road-network, if available.

ACKNOWLEDGMENT

The authors would like to thank the members of GM-CMU Autonomous Driving Collaborative Research Lab for their effort and dedication.

REFERENCES

[1] C.-K. Chang, C. Siagian, and L. Itti, "Mobile robot monocular vision navigation based on road region and boundary estimation," in *Proceedings of IEEE International Conference on Intelligent Robots and Systems*, 2012, pp. 1043–1050.

[2] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. Bradski, "Self-supervised monocular road detection in desert terrain," in *Proceedings of Robotics: Science and Systems*, 2006.

[3] D. si Tue-Cuong, G. Dong, Y. C. Hwang, and O. S. Heung, "Robust extraction of shady roads for vision-based ugv navigation," in *Proceedings of IEEE International Conference on Intelligent Robots and Systems*, 2008, pp. 3140–3145.

[4] O. Miksik, P. Petyovsky, L. Zalud, and P. 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*, 2011, pp. 64–71.

[5] I. Katramados, S. Crumpler, and T. P. Breckon, "Real-time traversable surface detection by colour space fusion and temporal analysis," in *Proceedings of International Conference on Computer Vision Systems*, 2009, pp. 265–274.

[6] S. Zhou, J. Gong, G. Xiong, H. Chen, and K. Iagnemma, "Road detection using support vector machine based on online learning and evaluation," in *Proceedings of IEEE Intelligent Vehicles Symposium*, 2010, pp. 256–261.

[7] Y. Alon, A. Ferencz, and A. Shashua, "Off-road following using region classification and geometric projection constraints," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[8] E. D. Dickmanns and B. D. Mysliwetz, "Recursive 3-d road and relative ego-state recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 199–213, 1992.

[9] S. J. Julier and J. K. Uhlmann, "A new extension of the kalman filter to nonlinear systems," in *Proceedings of SPIE Signal Processing, Sensor Fusion, and Target Recognition*, 1997.

[10] U. Franke, H. Loose, and C. Knoppel, "Lane recognition on country roads," in *Proceedings of IEEE Intelligent Vehicle Symposium*, 2007, pp. 99–104.

[11] K. Peterson, J. Ziglar, and P. E. Rybski, "Fast feature detection and stochastic parameter estimation of road shape using multiple lidar," in *Proceedings of IEEE Conference on Intelligent Robots and Systems*, 2008, pp. 612–619.

[12] S. Strygulec, D. Muller, M. Meuter, C. Nunn, S. Ghosh, and C. Wohler, "Road boundary detection and tracking using monochrome camera images," in *International Conference on Information Fusion*, 2013, pp. 864–870.

[13] P. Smuda, R. Schweiger, H. Neumann, and W. Ritter, "Multiple cue data fusion with particle filters for road course detection in vision systems," in *Proceedings of IEEE Intelligent Vehicles Symposium*, 2006, pp. 400–405.

[14] B. Southall and C.J.Taylor, "Stochastic road shape estimation," in *Proceedings of IEEE International Conference on Computer Vision*, 2001, pp. 205–212.

[15] H. A. Mallot, H. H. Blthoff, J. J. Little, and S. Bohrer, "Inverse perspective mapping simplifies optical flow computation and obstacle detection," *Biological Cybernetics*, vol. 64, no. 3, pp. 177–185, 1991.

[16] E. A. Wan and R. van der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proceedings of IEEE Symposium on Adaptive Systems for Signal Processing, Communications, and Control*, 2000, pp. 153–158.

[17] J. Wei, J. Snider, J. Kim, J. Dolan, R. Rajkumar, and B. Litkouhi, "Towards a viable autonomous driving research platform," in *Proceedings of IEEE Intelligent Vehicles Symposium*, 2013, pp. 763–770.

[18] D. R. Martin, C. C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 1, pp. 1–20, 2004.