Lane Tracking in Hough Space Using Kalman filter

Kyuhyoung Choi[1], Kyungwon Min[2], Sungchul Lee[2], Wonki Park[2], Yongduek Seo[1] and Yousik Hong[3]
[1]Graduate school of media at Sognag Uinv. Korea
{kyu, yndk}@sogang.ac.kr
[2]SoC Research Center, Korea Electronics Technology Institute
{minkw, leesc, wkpark74}@keti.re.kr
[3]Dept. computer eng. Sangji Univ. Korea
yshong@sangji.ac.kr

Abstract

This paper deals with vision-based lane detection and tracking which is a necessary part of a lane departure warning system(LDW). Hough transform is used for both detection and tracking. Lane is detected in consequence of vanishing point detection. Then the detected lane which is a pair of line is tracked in Hough space, that is the space of line parameters in a polar coordinates. For frames where observation is poor the proposed system uses the information obtained from the frames where observation was pretty good. Experimental results show the robustness of the proposed algorithm.

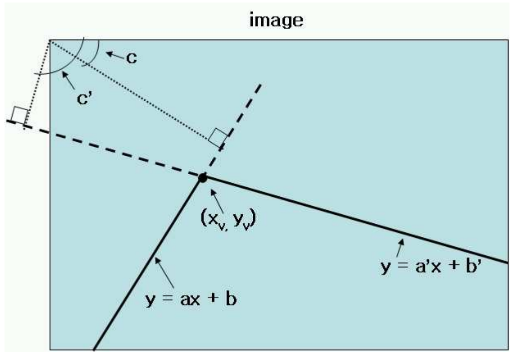


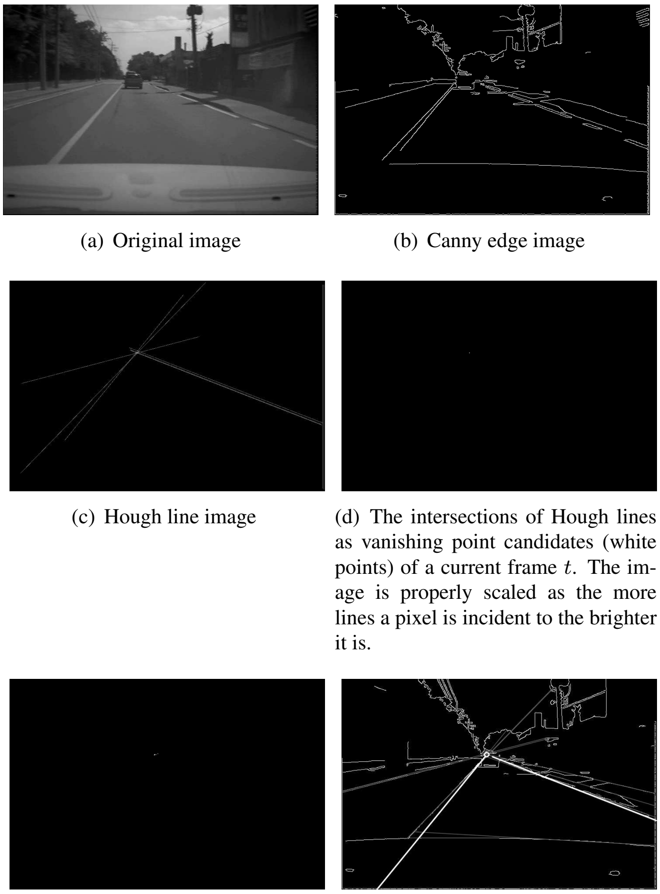
Figure 1. Illustration of road lane model and parameters.

1 Introduction

Intelligent vehicle has been an interesting topic in a couple of decades. Even it is yet far from a smart car that most people expect [8], major auto companies have launched cars with primitive driver assistance system(DAS) one of whose features is lane departure warning system(LDW) [2]. Most of LDW are based on vision, that is, processing the images captured by on board camera(s) [10, 5, 3, 6, 9, 1]. In previous studies, many geometric curves (including line) have been suggested for modelling a lane, some of which are parametric [3, 10, 1] while others are not [9, 5]. Also they can be classified according to the need of camera calibration, that is, whether the tracked parameters are those of a real 3D road [1, 3, 6] or 2D image [9, 5, 3, 10]. For the rest of this paper, the camera is considered to be monocular, looking forward and mounted around rear-view mirror. The result of image processing on the image sequence is detection and tracking of lane boundaries. The difficulties arising in lane detection and tracking are as following.

- The road surface is not planar. Due to many reasons, some parts of a road surface may be convex while other parts are concave. Furthermore, very often the slope of near view surface is different from that of far view. This makes the imaged lines are not straight.
- The width of lane, that is, the distance between lane boundaries not constant while a car is moving along the road. Literally, this poses another variable to estimate.
- The lighting condition varies along the road surface and shadows casted by trees and other vehicles clutter the road.
- The other cars in front occlude lines and distract the tracking system unless there is already a module for detecting and tracking them.

A nice part of lane tracking is that most of the time, the moving car is in the center of the lane and parallel to the boundaries as long as the driver is sane. This implies that



(a) Original image

(b) Canny edge image

(c) Hough line image

(d) The intersections of Hough lines as vanishing point candidates (white points) of a current frame t. The image is properly scaled as the more lines a pixel is incident to the brighter it is.

(e) Vanishing point probability as an accumulated histogram on frames from $t - \Delta t$ to t. The histogram is properly scaled for visualization

(f) Detected lane boundaries and the vanishing point marked as white thick lines and circle respectively.

Figure 2. Intermediate images during lane detection.

Input: Sequential Canny edge images $\{E_t\}$ (See Figure 2(b)) and accumulated 2D histogram o (See Figure 2(e))
Output: Left(q_L) and right(q_R) lines of imaged lane boundaries and vanishing point v
Set count C zero ($C = 0$)
Set all the bin values of o zero ($o(m) = 0, \forall m$)
foreach *time* t **do**
  Hough transform E_t to detect a set of lines, K_t (See Figure 2(c))
  foreach *line* $i \in K_t$ **do**
    Compute the orientation n_i of the line.
  end
  foreach *pair of lines* i *and* j $(i, j \in K_t$ *and* $i \neq j)$ **do**
    Find their intersection $m_{i,j}$
    Increase histogram value $h_t(m_{i,j})$ by 1 (See Figure 2(d))
  end
  if $\max_m (h_t(m)) > T_h$ **then**
    Add the current histogram to the accumulated histogram ($o = o + h_t$)
    Increase the count C by 1
    if $C > T_C$ **then**
      Find the vanishing point if exists as well as left and right boundaries. (Algorithm 2)
    end
  end
  else
    $C = 0$
    $o(m) = 0, \forall m$
  end
end

Algorithm 1: The lane detection algorithm by histogram of line intersections

detection of failure and recovery(re-detection) is as important as tracking desperately.

In this sense, in this paper, the imaged road lane is modelled as a pair of straight lines whose intersection is the vanishing point. The case of traveling very curvy road is not covered in this paper.

2 Lane Model

Since a real road lane consists of two boundaries(lines), the imaged lane is modelled as a pair of lines. So the dimension of state space is four: the slope and y-intercept of each line(a, b, a' and b' in Figure 1). Equivalently, instead of slope and y-intercept, we use the vanishing point coordinates and orientation of each line(x_v, y_v, c and c' in Figure 1) as elements of the state vector **v** which shows more linear evolution [7].

3 Lane Detection

The lane detection method used in this paper is based on the Hough-based vanishing point detection method which is most popular in this purpose. The main idea is to wait until the vanishing point is detected at the same position for some consecutive frames. The pseudo code of the detection module is described in Algorithm 1 where T_h, T_C, T_o and T_D are thresholds whose values are properly set. The addition and subtraction between histograms h_1 and h_2 of the same size is defined as following.

$$h_1 \pm h_2 := h_1(m) \pm h_2(m) \tag{1}$$

where m is the index of a bin.

Input: Set of detected lines K_t at the current frame t and histogram of line intersections o
Output: vanishing point v, left(q_L) and right(q_R) lane boundaries if exists. Updated histogram o

$m^* = \arg\max_{m} (o(m))$
if $o(m^*) > T_o$ **then**
  Set vanishing point v as m^*
  foreach (i, j) *such that* $i, j \in K_t, n_i > 0, n_j < 0$ **do**
    Get distance D from $m_{i,j}$ to v
    if $D < T_D$ **then**
      Add the pair (i, j) to the set R
    end
  end
  $(i^*, j^*) = \arg\min_{(i,j)\in R} n_i - n_j$
  Set q_L and q_R as i^* and j^* respectively (See Figure 2(f)).
  Break
end
Subtract the histogram of $t - T_C$ from the accumulated histogram ($o = o - h_{t-T_C}$)

Algorithm 2: Estimation of the vanishing point as well as both lane boundaries.

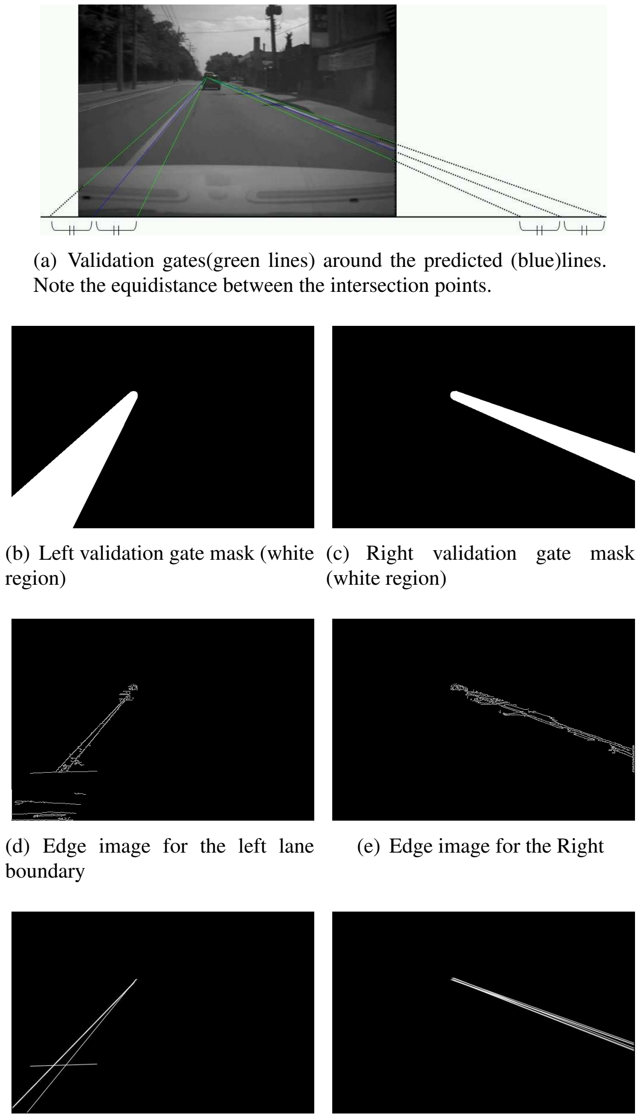
The intermediate result images are shown in Figure 2.

After detecting the vanishing point, the left lane boundary is chosen as the most vertical among the lines whose distance to the vanishing point are small enough and slope are positive, and the right boundary vice versa.

4 Lane Tracking

The proposed tracking has three cases according to measurements of lines on validation gates (See Figure 3):

- When both lines are available, find a most valid pair of detected lines as the measurement of a Kalman filter.
- When either of them is not available, the proposed algorithm makes a dummy measurement for the unavailable then passes the measurement pair to the Kalman filter. For example, if there is no lines detected from the left validation gate, the system make some expected left lines whose angles are different from the detected right lines by the moving average w of orientation angle difference. In some sense, this is similar to the expectation step of EM(Expectation Maximization). For this, the system computes and saves w, the moving average of angle difference from the right to left lines by weighted sum of the previous value of w and the estimated angle difference when the current frame is of the first case.
- When none of both are detected for long enough consecutive frames, the system takes it as tracking failure then switches to lane (re)detection mode



(a) Validation gates(green lines) around the predicted (blue)lines. Note the equidistance between the intersection points.

(b) Left validation gate mask (white region)

(c) Right validation gate mask (white region)

(d) Edge image for the left lane boundary

(e) Edge image for the Right

(f) Detected lines for the Left

(g) Detected lines for the Right

Figure 3. Example images of lane tracking steps.

Even though there are at least one side where lines are detected, all the detected lines might be from noise or clutters. In that case, it is rather to take it as there are no detected lines from neither of sides. For this, a Mahalanobis distance d of the hypotheized state from the Kalman-predicted state is computed as :

$$d_{i,j} = (\mathbf{v}_{i,j} - \tilde{\mathbf{v}})^T Q^{-1} (\mathbf{v}_{i,j} - \tilde{\mathbf{v}}) \tag{2}$$

Input: Sequential sets of Hough lines $\{K_t\}$ and an integer $\#Failure$ counting the consecutive frames where the tracking is failed.
Output: Vanishing point coordinates $(\hat{x}_v, \hat{y}_v)$ and orientations of both lane boundaries, $\hat{n}_L$ and $\hat{n}_R$ respectively estimated by a Kalman filter M and a flag showing whether it should switch to detection mode or not.

```
Set the count #Failure zero
foreach time t do
    Kalman-Predict vanishing point ((x̃_v, ỹ_v)) and
    lines of both sides (ñ_L, ñ_R):
      ṽ = (x̃, ỹ, ñ_L, ñ_R)
    foreach side s ∈ {L, R} do
        Mark validation gate V_s (Figure 3(a) to 3(c))
        Find a line set B_s from K whose clipped
        length by V_s is long enough (Figure 3(f), 3(g)).
    end
    if At least, one of line sets is not empty then
        Enumerate and evaluate all the possible
        combinations of line pairs. (Algorithm 4)
        Find the best line pair of minimum distance:
          i*, j* = min_{i,j} d_{i,j}
        if d_{i*,j*} < T_m then
            Get the estimation of the current state v̂ by
            updating the Kalman filter M
            Set #Failure zero
            if None of line sets was empty then
                Update the moving average w:
                  w = f (n̂_R − n̂_L) + (1 − f) w
            end
        end
        else
            Increase #Failure by 1.
        end
    end
    else
        Increase #Failure by 1.
    end
    if #Failure > T_f then
        Break
    end
end
```

Algorithm 3: Lane tracking using Kalman filter

where $\mathbf{v}_{i,j} = (x(m_{i,j}), y(m_{i,j}), n_i, n_j)^T$ and Q is the covariance matrix of the Kalman filter. So unless there is a line pair whose d is smaller than a certain threshold T_m, observations(line detection) are invalidated. The pseudo code of the Kalman-based tracking is described in Algorithm 3 where T_f and T_m are thresholds whose values are properly set. The weight factor f is also set beforehand.

In the proposed system, the dynamic model of state is assumed to be zero velocity, that is, the transition matrix of the Kalman filter is identity. So the validation gates of the current frame as in Figure 3(a) are marked around the estimated lane boundaries at the previous frame.

Input: Left(B_L) and right(B_R) sets of valid lines.
Output: Set of all the possible combinations of line pairs U and set of corresponding cost $\{d_i\}_{i \in U}$

```
Set the count #Failure zero
if None of line sets is empty then
    Make a set of all possible combinations of line
    pairs U from the line sets:
      U = {(i, j) | i ∈ B_L, j ∈ B_R}
end
else
    if Left set is empty then
        Make U from B_L:
          U = {(i, j) | n_j = n_i + w, i ∈ B_L}
    end
    else
        Make U from B_R:
          U = {(i, j) | n_i = w − n_i, j ∈ B_R}
    end
end
foreach pair of lines (i, j) in U do
    Find their intersection m_{i,j}
    Compute n_i and n_j
    Compute the distance d_{i,j} as in Equation 2
end
```

Algorithm 4: Measurements and evaluations

5 Experimental Results

Figure 4 shows the images of the lane tracking result on a typical road at every 100th frame. The estimated angles in degree of both sides are displayed as well as the frame number. Note that, during the sequence, there are vehicles in front of and beside the host car as well as road humps. Moreover, in some frames, some parts of road lane markings are missing and/or heavy shadows are casted by roadside trees. Nevertheless the proposed system showed stable performance in detection and tracking. OpenCV library is used throughout the implementation [4]. For example, `cvHoughLines2` for line detection, `cvCanny` for edge detection and `CvKalman` is used for Kalman filtering.

6 Conclusion

The proposed lane detection and tracking system is based on a Kalman filter whose observation is line detection by Hough transform. Instead of tracking each side independently, our system estimates parameters of both lines together in Hough space. As a result, its performance is stable along ordinary but challenging roads.

Acknowledgements

This work was supported by System Integrated Semiconductor Technology Development Project(System IC 2010) fund of South Korea.

References

[1] M. Bertozzi, L. Bombini, A. Broggi, P. Zani, P. Cerri, P. Grisleri, and P. Medici. Gold: A framework for developing intelligent-vehicle vision applications. *IEEE Intelligent Systems*, 23(1):69–71, 2008.

[2] BMW. 6 series coupe', 2007.

[3] Q. Chen and H. Wang. A real-time lane detection algorithm based on a hyperbola-pair model. *Intelligent Vehicles Symposium, 2006 IEEE*, pages 510–515, June 2006.

[4] Intel. Open source computer vision library. http://www.intel.com/technology/computing/opencv/index.htm, October 2006.

[5] D. J. Kang, J. W. Choi, and I. S. Kweon. Finding and tracking road lanes using line-snakes. *Intelligent Vehicles Symposium, 1996., Proceedings of the 1996 IEEE*, pages 189–194, Sep 1996.

[6] J. McCall and M. Trivedi. An integrated, robust approach to lane marking detection and lane tracking. *Intelligent Vehicles Symposium, 2004 IEEE*, pages 533–537, June 2004.

[7] S. Mills, T. Pridmore, and M. Hills. Tracking in a hough space with the extended kalman filter. In *Proc. British Machine Vision Conference*, 2003.

[8] NBC. Knight rider, 1982 - 1986.

[9] Y. Wang, E. Teoh, and D. Shen. Lane detection and tracking using b-snake. *Image and Vision Computing*, 22(4):269–280, April 2004.

[10] B. Yu and A. Jain. Lane boundary detection using a multiresolution hough transform. pages II: 748–751, 1997.

(a) 100th frame (b) 200th frame (c) 300th frame (d) 400th frame (e) 500th frame (f) 600th frame (g) 700th frame (h) 800th frame (i) 900th frame (j) 1000th frame (k) 1100th frame (l) 1200th frame (m) 1300th frame (n) 1400th frame

Figure 4. Example images of lane tracking results at every 100th frame.