Lane Detection & Localization For UGV in Urban Environment*

Teawon Han, Yanghyun Kim, and Kisung Kim *Research Engineer, Samsung Techwin*

***Abstract*— Generally, the main components of autonomous driving system consists of perception (geometry recognition, localization, and objects detection & tracking) and navigation processes (global & local path planning, and controller). In this paper, we focus on finding an accurate position for Unmanned Ground Vehicle (UGV) in urban environments. A GPS sensor is fundamentally used to get a current global position, but its accuracy is susceptible to satellite geometry or receiver's conditions; even though high performance guaranteed GPS such as differential-GPS (DGPS) always return precise position(position error is less than 0.4m) because of problems such as disconnection between base station and vehicle. The position-errors, even few meters, could engender serious accidents when UGV is driving under urban environments. In this paper, we suggest the *localization algorithm* and *novel lane detection algorithm*. The detected lane information is implemented to overcome GPS sensor's position-errors. First, our novel lane detection algorithm is described, and then the localization algorithm is discussed. This paper also provides experimental results of the lane-detection and the localization in urban environment by using UGV.**

I. Introduction

An Unmanned Ground Vehicle (UGV) has been researched and developed for decades. Especially, the DARPA (Defense Advanced Research Projects Agency) Grand Challenge had inspired researches related to UGV. Among various kinds of modules (e.g., object detection & tracking, traffic signal recognition, localization, path planning, etc.,) for autonomous driving, we are focusing on localization in this paper.

The basic idea of global localization is searching a current position on a map regardless of map is given. For UGV, a GPS-sensor is generally used to get the global position, but other sensors are also implemented to increase accuracy of position [2] [3] A depth camera (e.g., the Microsoft Kinect sensor) [1], stereo camera [2], and LIDAR (e.g., Velodyne HD-LIDAR 64-beam scanner) [3] have implemented with GPS sensor to obtain more precise position. However, these additional sensors are required more processing time and computations. For example, additional localization method that is using 3D scan-matching with LIDAR sensor needs much more computations than using only a GPS sensor, while its result is better. In addition, there are external supporting methods for increasing accuracy of positions such as DGPS. However, external supporting methods are depending on the status of communication.

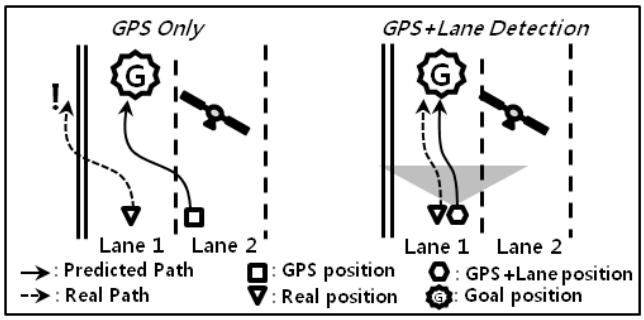

Figure 1. Scenario of autonomous driving under urban environments with only-GPS sensor and the practical localization with lane detection.

Therefore, we propose the practical localization algorithm to overcome GPS sensor's errors without external supports. For the algorithm, lane information is implemented, with single camera and GPS sensor in this paper. There are related researches that applied lane information for different purposes. Sayanan Sivaraman and Mohan Manubhai Trivedi [4] implemented lane information to detect and trace other vehicles on road. They mentioned that lane is helpful to prevent false alarms in detecting vehicles assuming the vehicles are driving on lane. Z. Tao et al., [5] implemented lane for localization. The purpose of applying lane information is similar to our research, but the algorithm is totally different. In [5], they are using lane patterns as a set of features to construct the map called mobile-map. The mobile map consists of synchronized sequential data that includes global positions, orientations, and lanes. Through the mobile-map, they can access a position by searching a matched lane pattern in the map. On the other hand, our map called the road information file (*RIF*) is including the center positions and orientations of lanes. In comparing two different localization methods, *RIF* is easier and simpler to construct than a mobile-map. It doesn't need lots of memory spaces even if map is expending continuously while the mobile-map needs much more memory spaces. In addition, we don't use matching algorithms that need additional computations and time while the Z. Tao et al., apply the *point-to-curve map-matching process* [6].

This paper is structured as follows: In section II, we introduce SAMSUNG TECHWIN's Unmanned Ground Vehicle and *RIF*. In section III, an novel lane detection algorithm is described. Section IV shows how to calculate and compensate GPS sensor's position errors by using lane information and RIF. Lastly, experimental results are shown in section V and discussion and future work are in section VI.

*Resrach supported by Samsung Techwin.

Teawon Han is a research engineer in Robot Technology group, Advanced Technology Institute, Samsung Techwin R&D Center, Seongnam-si, Gyeonggi-do, 463-400 South Korea (phone: +82-70-7147-7737; e-mail: teawon.han@samsung.com).

Yanghyun Kim and Kisung Kim are research engineers in Robot Technology group, Advanced Technology Institute, Samsung Techwin R&D Center, Seongnam-si, Gyeonggi-do, 463-400 South Korea (e-mail: yan.kim@samsung.com, ks.kim@samsung.com).

II. RELATED WORK

A. Samsung Techwin (STW) Unmanned Ground Vehicle

STW recreated a commercial vehicle (model is QM5 made by Renault Samsung), to UGV. STW has been researching and developing autonomous vehicles and robots for various purposes. Actually, STW-UGV was designed not just for an autonomous driving vehicle, but also one of surveillance solutions to secure broad sites like airports or power plants. December 2011, STW successfully demonstrated an intelligent mobile surveillance system at Incheon International Airport. At that time, a pen-tilt-zoom (PTZ) camera was installed on the top of UGV's roof to detect and warn unauthorized people. STW-UGV basically embeds three laser scanners, one GPS sensor, and a camera. The sensors are assigned depending on the UGV's applications. It is able to drive 90km/h (max-speed) fully autonomously on urban environments [figure 2].

Figure 2. The samsung techwin unmanned ground vehicle & autonomous driving system.

B. Road Information File (RIF)

For UGV under urban environments, we need not only a geographic map, but also traffic information such as speed limitation and structures of transactions. *RIF* includes traffic and road information. Road is consisted by lanes, and each lane is represented by a set of global coordinates and orientations. Different types of zones such as intersection zone, parking zone, and school zone are also described. It is usually used in path planning, but perception modules implement it to reduce a processing time and avoid false alarms. For example, it is not efficient that the traffic signal detector, one of perception modules, is always trying to detect signals when the vehicle is driving the road where is no signal. Therefore, *RIF* can be a reference for each module to decide when and where it should be executed. In this paper, *RIF* is assisting to save process time and avoid false alarms in detecting the lane. Also it is used to calculate and compensate GPS sensor's position errors as well.

III. AN NOVEL LANE DETECTION

There are different kinds of vision-based methods in lane detection. In [8], Shengyan Zhou et. al., categorized the methods into three classes: feature-based, region-based, and model-based. Here, we are using a feature-based method to detect a lane. As we described in section I, the purpose of lane detection is to overcome GPS sensor's position errors and compensate it to get more accurate position. Specifically, our algorithm uses the location of robot in lane to measure GPS sensor errors. And, the vehicle location in lane is represented as a distance between left (or right) line and a center of vehicle. Before discussing specific algorithms, we define terms. A lane is consisted of two lines: *left-line* and *right-line*. Each line is reconstructed by finding and connecting two control points: start-point and end-point denoting *SP* and *EP*. SP and EP are selected from each pool of *Candidate-SPs* and *Candidate-EPs* respectively. Fundamentally, candidate points are collected from control points (CPs) and CPs are obtained by detecting lane features from camera images.

In this section, we are going to explain sequential processes of the *novel lane detection* following steps: (1) detecting lane features and collecting CPs, (2) generating Candidate-SPs and searching Candidate-EPs, (3) selecting two SPs for left-line and right-line, and (4) filtering Candidate-EPs and determining lane. Figure 3 is a diagram showing the overview of the lane detection process.

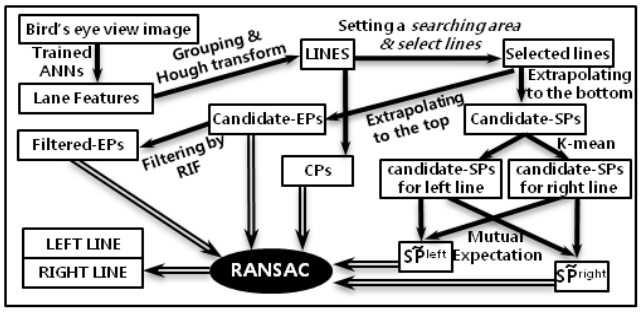

Figure 3. The overveiw of an novel lane detection.

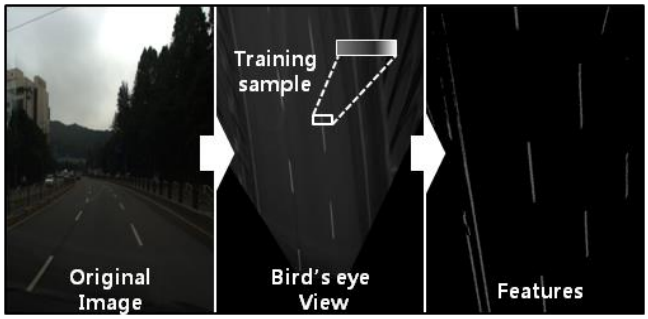

Figure 4. The sequence of receiving lane features from camera-image.

A. Detecting Lane Features and Collecting Control Points

Vision based lane detection is not only susceptible to a status of the painted lane on road (e.g. dirt, deletion, or etc.), but also the brightness of environment. That is why we choose feature-based method for lane detection; a model trained by various lane-samples would find lane-features on camera image. The artificial neural networks (ANNs) is implemented to detect lane features from images accepting ZuWhan Kim's analysis in [7]. We collect 21402 training samples and train an ANNs-model whose structure is four layers (input, output, and two hidden layers). The size of training samples is 9x3 pixels, and the samples are collected from bird's eye view images as shown in Figure 4 and [7]. Among detected features, we find CPs by following steps: a) grouping features by dividing them through grids as shown in Figure 5-Stage 1, b) finding a line-pattern in each grouped features by *Hough Transform*, and c) storing start-points and end-points of found line-patterns as CPs; CPs will be referred when selecting

Candidate-EPs and scoring lines of the lane during *RANdom SAmple Consensus* (RANSAC).

B. Generating candidate-SPs and Searching Candidate-EPs

A lane is represented by left-line and right-line, and a set of SP and EP is necessary to draw each line. Therefore, we should find the most appropriate four points (two SPs and two EPs) to detect a lane. Two SPs will be chosen from generated Candidate-SPs , and two EPs will be selected from CPs that were collected previously. To generate Candidate-SPs, we are using line-patterns that were taken by *Hough Transform* in *III-A*. First, we set a *searching area* (rectangular sized 2 meters width and 3 meters height) in front of the vehicle because the line-patterns where are far from the vehicle are not eligible to generate them accurately; a dotted-rectangular in Figure 5 is representing the *searching area*. Second, we generate Candidate-SPs by extrapolating selected line-patterns to the front of the vehicle as shown in Figure 5- Stage 3. Next, searching Candidate-EPs is following steps: a) extrapolating the selected line-patterns to the top of the image, b) setting rectangular boundaries (width 3 pixels) of which the extrapolated lines are centered as shown in Figure 5- Stage 4, and c) choosing a CP where is at uppermost in each boundary. The chosen CPs will be Candidate-EPs.

C. Selecting two SPs for left-line and right-line

We divide Candidate-SPs into two classes by using *K-mean* to recognize which SP could become a part of left-line or right-line; one class (C^{left}) is for finding a SP of left-line (SP^{left}), and another (C^{right}) is for detecting a SP of right-line (SP^{right}). For *K-mean*, initial mean-values of C^{left} and C^{right} are set left-most and right-most points respectively among Candidate-SPs as shown in Figure 6; the classes are defined as eq. (1) and (2). Next, we implement the *Mutual-Expectation* to choose two points, SP^{left} and SP^{right}, in the two classes as shown as in *Table I*. This is because that the *Mutual-Expectation (ME)* [9] is efficient to find the best answer given noisy data-sets if and only if each data-set represents different information, but are related each other. . To be specific, we have two different data-sets, C^{left} and C^{right}, and they have a relation that left-line's location is always distanced from right-line constantly. To choose the best SP in each class, we calculate likelihoods of every Candidate-SPs in both classes by using Gaussian distribution as shown in Figure 6-(a) and (b). Following the concept of *ME*, the likelihoods of SPs in C^{left} are obtained through SPs in C^{right} by setting the *mean*-values of Gaussian distributions like eq. (5) and (6). SP^{left} and SP^{right} will be chosen as one of Candidate-SPs whose likelihood is the biggest in each class.

$$C^{left} = \{sp_1^{left}, sp_2^{left}, \ldots, sp_n^{left}\} \; where \; sp_n^{left} = (x_n, y_n) \quad (1)$$

$$C^{right} = \{sp_1^{right}, sp_2^{right}, \ldots, sp_m^{right}\} where \; sp_m^{right} = (x_m, y_m) \quad (2)$$

$$D_{ppm} = ([SP_{t-1}^{right}].x - [SP_{t-1}^{left}].x) / L_{meter} \quad (3)$$

where [*Pos*].***x*** refers to *x*-coordinate of *Pos* (position).

$$L_{pixel} = L_{meter} \times D_{ppm} \quad (4)$$

$$\mu_i^{left} = sp_i^{right} - L_{pixel}, \quad i = \{1,2, \ldots, m\} \quad (5)$$

$$\mu_i^{right} = sp_i^{left} + L_{pixel}, \quad i = \{1,2, \ldots, n\} \quad (6)$$

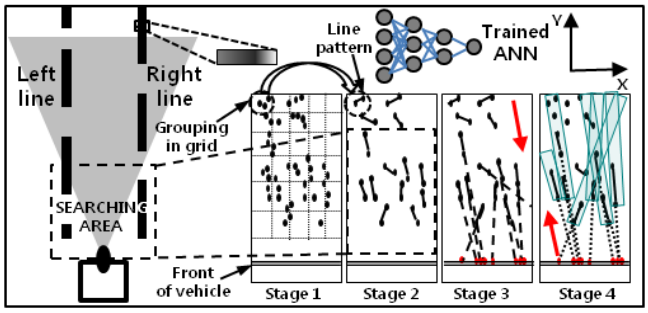

Figure 5. Steps in collecting CPs, generating SP-candidates, and searching EP-candidates. In stage 1, dots are features of lane obtained by ANNs. In stage 2~4, dots are CPs. Stage 3 and 4 show extrapolations of lines to generate candidate-SPs and search candidate-EPs respectively.

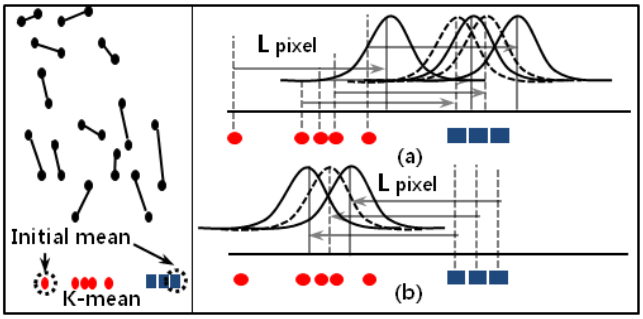

Figure 6. The process of obtaining likelihoods of Candidate-SPs in both classes. Rectangular and circular points are Candidate-SPs in C^{right} and C^{left} respectively. (a) and (b) are representing likelihoods for Candidate-SPs in C^{right} and C^{left}. K-mean is started with two initial mean-values for two caterories. *L pixel* is a given width of lane.

TABLE I. PSEUDO CODE FOR CHOOSING SP^{LEFT} AND SP^{RIGHT} BY *ME*

```
/* Assuming that width (meters) of lane is given, we need to
   convert width's unit from meters to pixels */
/* D_ppm (pixel per meter) is updated at time t by using detected
   SP^left and SP^right at time t-1*/

L_meter = 3.3;                          /* Lane's width is 3.3m */
D_ppm = ([SP^right_t-1].x - [SP^left_t-1].x ) / L_meter

/* SP^right_t-1 and SP^left_t-1 are detected start-point of
   right-line and left-line at t-1, Eq (3) and (4) */
L_pixel = L_meter × D_ppm               /* Pixels of lane's width */

/* Converting and updating End */
/* Classifying Candidate-SPs */
[C_left, C_right] = k-mean(initM^left, initM^right, Candidate_SPs)

/* set mean-values and standard deviation for Gaussian
   distribution */
/* standard deviation is half of lane's width */
std = L_pixel / 2
For i from 0 to size of C^right
    μ_i^left = sp_i^right - L_pixel                    /* Eq (5) */
    For j from 0 to size of C^left
        P_tmp^left = f_g(sp_j^left, μ_i^left, std )    /* Eq (7) */
        P_j^left += P_tmp^left                         /* Eq (9) */
    End for
End for
```

```
For i from 0 to size of C^left
    μ_i^right = sp_i^left + L_pixel                    /* Eq (6) */
    For j from 0 to size of C^right
        P_tmp^right = f_g(sp_j^right, μ_i^right, std)  /* Eq (7) */
        P_j^right += P_tmp^right                       /* Eq (8) */
    End for
End for

/* choose a point that has biggest likelihood in each class as a SP*/
/* choose best SP of right-line*/
t_right = P_0^right
t_left = P_0^left
For i from 1 to size of C^left
    If (t_left < P_i^left) then
        t_left = P_i^left                              /* Eq (10) */
        SP_left = sp_i^left   /* choose a point whose likelihood is the biggest in the class*/
    end if
End for

/* choose best SP of left-line*/
For i from 1 to size of C^right
    If (t_right < P_i^right) then
        t_right = P_i^right                            /* Eq (11) */
        SP_right = sp_i^right  /* choose a point whose likelihood is the biggest in the class*/
    end if
End for
```

To obtain vehicle's position through detected lane, the *pixels per a meter* (D_{ppm}) should be defined to convert from pixels to meters because the obtained distance between the vehicle and lane on image is represented by pixels. Initially, D_{ppm} is set by using calibration borad on flat ground, but it is autonomously updated over time. The updating D_{ppm} is necessary because it is depending on camera's pitch, and the pitch is changing when vehicle is driving through bumps, uphill, or downhill. As shown in Eq. (3) and (4), it is updated by using detected lane at previous moment, and implemented to convert given width of lane (L_{meter}) to L_{pixel}.

$$f_g(x, \mu_i^{right\ or\ left}, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu_i^{right\ or\ left})^2}{2\pi^2}} \quad (7)$$

$$where\ \sigma = L_{pixel} / 2$$

$$P_j^{right} = \sum_{i=1}^{n} f_g(sp_j^{right}, \mu_i^{right}, \sigma) \quad (8)$$

$$where\ i = \{1, 2, ..., n\} \text{ and } j = \{1, 2, \ldots, m\}$$

$$P_j^{left} = \sum_{i=1}^{m} f_g(sp_j^{left}, \mu_i^{left}, \sigma) \quad (9)$$

$$where\ i = \{1, 2, ..., m\} \text{ and } j = \{1, 2, \ldots, n\}$$

$$t\_right = \arg\max_i(P_i^{right}), \quad i = \{1, 2, ..., m\} \quad (10)$$

$$t\_left = \arg\max_j(P_j^{left}), \quad i = \{1, 2, ..., n\} \quad (11)$$

D. Filtering candidate-EPs and Determining Lane

We use the *RIF* to filter some of candidate-EPs. The unnecessary candidate-EPs are filtered by choosing points that satisfy following *conditions*: 1) EP's *x*-coordinate is smaller than SP_{left}'s *x* coordinate on right-curve or 2) EP's x coordinate is larger than SP_{right}'s *x*-coordinate on left curve. Because *RIF* includes lane information as a set of positions and orientations, we can recognize whether current lane's shape is straight, left-curve, or right-curve by calculating curvature of forward lane's positions. For example, there are SP^{left}, SP^{right}, and Candidate-EPs on the left-curve in Figure 7. Through *RIF* information, we can expect that the lane's shape would be left-curve before detecting the lane completely, and then remove unnecessary Candidate-EPs (EP_3 and EP_4) following the above *condition*-2. This helps to avoid false alarms and reduce computations.

Finally, left-line and right-line will be detected by using *RANdom SAmple Consensus* (RANSAC) with two SPs and Candidate-EPs. Iteratively, randomly chosen EP and SP^{left} or SP^{right} will produce hypotheses for lane such as Case 2 in Figure 7, and left-line and right-line are chosen among them by scoring. The score of each hypothesis is the number of CPs where are inside of each boundary. The rectangular boundaries of which each produced line is centered are set as shown Case 1 in Figure 7. The two lines that obtained the highest score in each group (C^{left} and C^{right}) become left-line and right-line. Lastly, we check the consistency of detected lines over time. Specifically, comparing between detected lines at *t* and *t-1*, if the difference is over the threshold, the lines at *t* are considered as false alarms based on an assumption that lane's location cannot be jumped to far away at once. After detection of current lane, vehicle's location can be recognized. As we mentioned before, vehicle's location on the lane is represented by using a distance (*meters*) from a center of vehicle to left (or right) line as shown in Eq. (12) and (13). The vehicle's center-position (Pos_{vc}) is given as a coordinate on a camera image.

$$D_{left} = \text{Abs}([Pos_{vc}].x - [SP^{left}].x) / D_{ppm} \quad (12)$$

$$D_{right} = \text{Abs}([Pos_{vc}].x - [SP^{right}].x) / D_{ppm} \quad (13)$$

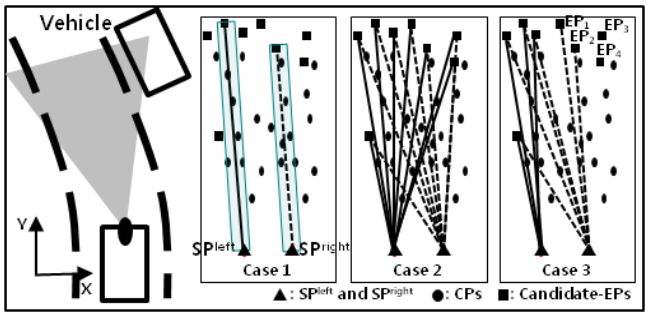

Figure 7. The scoring process (Case 1) and two different methods for producing hypotheses for lane. The result of only-RANSAC is Case 2 and one of RANSAC after filtering Candidate-EPs is Case 3. In cases, rectangular dots are refering to Candidate-EPs, and the triangular dots on bottom are SPs. In case 2 and 3, lines and dotted lines are produced hypotheses for left-line and right-line.

IV. Localization By GPS, Lane, And RIF

In this paper, GPS sensor's errors are measured and compensated by using detected lane information with *RIF*. Applying the taken position from lane detection into *RIF*,

vehicle's rough *global position* is predicted. And, the predicted global position is implemented to calculate GPS sensor's errors by measuring difference between the predicted position and GPS sensor's position.

A. Converting Vehicle's location on lane to global position

D_{left} or D_{right} taken from lane detection is provided as a vehicle's location. However, both are not global coordinate, but distance (*meters*) from left-line or right-line. To calculate GPS sensor's errors, we need to convert the obtained distance from left-line or right-line to global position. *RIF* and GPS sensor's position are implemented. Initially, the lane's number is given, and *RIF* includes lane information as s set of global positions (lane's center positions) and orientations like shown Eq.(14). Among lane positions in *RIF*, the closest position ($P_{closest}$) to position taken from GPS sensor (P_{gps}) is chosen to convert the distance on lane to global position. The converted position ($P_{predicted}$) is taken through Eq. (15) and (16) as shown in *Table II* and Figure 8.

$$P^{i}_{lane\_c} = \{ (p^{i}_{1},\ \theta^{i}_{1}), (p^{i}_{2},\ \theta^{i}_{2}), \ldots, (p^{i}_{n},\ \theta^{i}_{n}) \} \quad (14)$$

where $p^{i}_{n} = (x_n, y_n)$, and θ^{i}_{n} is a heading (front-direction of vehicle), and i is a lane's number.

$$\mathrm{Diff}_{exp} = D_{left} - (L_{meter} / 2) \quad (15)$$

$$P_{predicted} = ([P_{closest}].x + \mathrm{Diff}_{exp}, [P_{closest}].y) \quad (16)$$

TABLE II. PSEUDO CODE FOR CONVERTING VEHICLE'S LOCATION TO GLOBAL POSITION BY USING RIF

```
/* D_left is obtained distance from lane detection*/
/* P_gps is current position taken from GPS sensor.*/
lane_num = 1                    /*  Initial lane's number is given  */

/* Searching the closest a global position to P_gps in RIF */
P_closest = f(P^lane_num_lane_c , P_gps )

/* Calculate distance from vehicle to lane's center position */
Diff_exp = D_left – ( L_meter / 2 )                    /*  Eq (15)  */

/* Predicted position is taken */
P_predicted = ( [P_closest].x + Diff_exp, [P_closest].y )    /*  Eq (16)  */
```

Meanwhile, the predicted position has uncertainty on y-axis as shown in Figure 8 (shaded ellipses are representing uncertainty of each position) when vehicle's *local x-y coordinate plane* is defined that the vehicle's front direction is y-axis and its orthogonal line is x-axis. This is because that lane information can be used to correct the position only on x-axis of the *local coordinate plane*. Moreover, the uncertainty of predicted position will affect in measuring GPS sensor's errors and obtaining accurate position. However, this is not critical because the lane is not parallel permanently, thus vehicle's *local x-y coordinate plane* is rotated following lane's shape. In other words, the uncertainty will not be accrued permanently because the vehicle's coordinate plane is rotating depending on lane's shape as shown in Figure 8; vehicle's *local x-y coordinate planes* at different positions are $x1$-$y1$, $x2$-$y2$, and $x3$-$y3$.

B. Measuring GPS sensor's errors & Obtaining New Position

To calculate GPS sensor's errors, $P_{predicted}$ and P_{gps} should be on same *x-y coordinate plane* by fitting their heading because headings of $P_{predicted}$ and P_{gps} are based on *RIF* and *current vehicle's orientation* respectively. We set 90° as a common heading (*y-axis*) and rotate both points through rotation function f_r as shown Eq. (17), (18), and (19). After GPS sensor's error is obtained by Eq. (20), m numbers of errors are stored and implemented to get a revised position ($P_{revised}$) as shown in *Table III*.

$$f_r(P, \theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} P.x \\ P.y \end{bmatrix} \quad (17)$$

$$P^{rot}_{predicted} = f_r(P_{predicted}, (90 - \theta_{predicted}) \quad (18)$$

$$P^{rot}_{gps} = f_r(P_{gps}, (90 - \theta_{gps}) \quad (19)$$

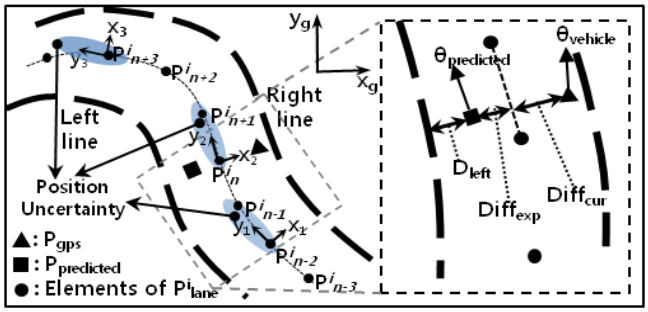

Figure 8. Analysis of GPS sensor's errors through lane information with RIF. $\mathbf{P}_{gps}$ and $\mathbf{P}_{predicted}$ are representing the vehicle's positions received from GPS sensor and lane detection respectively. $\mathbf{P}^{i}_{lane}$ represents ith lane's positions $\boldsymbol{\theta}_{predicted}$ and $\boldsymbol{\theta}_{vehicle}$.are headings of $\mathbf{P}_{predicted}$ and $\mathbf{P}_{gps}$.The shaded ellipses is representing uncertainty of each position.

TABLE III. PSEUDO CODE FOR OBTAINING REVISED POISITION

```
vector vE_gps                   /*  Stores m numbers of E_gps  */
If size of vE_gps equal to m then
    /*  After m numbers of E_gps are collected, oldest value is
        removed and new is stored at every time                  */
    remove first value in vE_gps
    push_back an obtained new E_gps into vE_gps

    /*  m numbers of E_gps are collected and get average value of
    them to obtain a revised position by Eq. (21)                */
    double avgE_gps = sum( vE_gps ) / m

    /* compensating GPS sensor's errors by Eq. (22) */
    P^rot_revised = P^rot_gps + avgE_gps
Else
    push_back an obtained new E_gps into vE_gps
End if
```

$$E^{t}_{gps} = [P^{rot}_{predicted}].x - [P^{rot}_{gps}].x \quad \textit{where } t \text{ is time} \quad (20)$$

$$\mathrm{avgE}_{gps} = (E^{t-m+1}_{gps} + \cdots + E^{t-1}_{gps} + E^{t}_{gps}) / m \quad (21)$$

$$P^{rot}_{revised} = ([P^{rot}_{gps}].x + \mathrm{avgE}_{gps}, [P^{rot}_{gps}].y) \quad (22)$$

$$P^{new}_{gps} = f_r^{-1}(P^{rot}_{revised}, (90 - \theta_{gps}) \quad (23)$$

Because the revised position is based on rotated orientation, final new position (P^{new}_{gps}) will be obtained by rotating reverse the revised position as shown in Eq. (23).

V. EXPERIMENTAL RESULTS

As shown in Figure 2, we installed a camera at the top-center of the vehicle and it takes 800×600 pixels sized image with 15 fps speed. We tested lane detection and localization algorithm at Midan City (South Korea). The driving path for experiment is 2.5km-long and shape is shown in figure 9. The *STW-UGV* is equipped with a NovAtel's SPAN-CPT (Single Point L1 mode: accuracy is 1.5 meters). In addition, *RIF* is built by using DGPS (accuracy is 0.4 meters). ProPak-v3 GPS receiver is implemented for base-station, and it communicates with rover through Pacific Crest's ADL Vantage Pro.

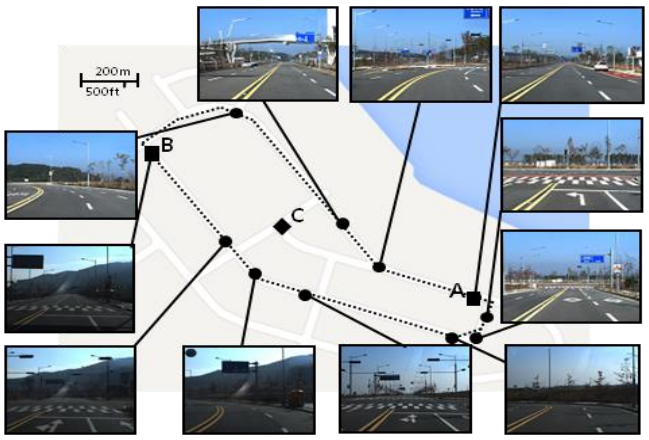

Figure 9. Midan City Map. Dotted line is the path for evaluating lane detection algorithm. Localization is tested on the path (from waypoint A to B). DGPS is installed at C.

First, we evaluate the *novel lane detection algorithm*. The algorithm is performed under various conditions such as day, night, and rainy because lane detection implements images coming from a camera, and the camera is usually susceptible to brightness and weathers. According to the experimental result, the lane-detection module detects 12 fps speed. Figure 8 is showing experimental results of the detections under different conditions. (a) and (e) are at daytime, (c) and (g) are at nighttime, and (b) and (f) are in rainy day. Moreover, (d) and (h) are showing the results of lane detection when another vehicle occludes lane partially; (a), (b), and (c) are on left-curve while (e), (f), and (g) are on straight-road. In each scene, there are three lines; the center-line refers to vehicle's center position, and two side lines are left-line and right-line. Occasionally, inaccurate result is observed on curves such as Figure 10-(a) because the algorithm is approximating detected lane-features to a straight line. However, it is ignorable because the pure GPS sensor's position is updated by using avgE_{gps}, not E_{gps}^{t}.

In addition, the *filtering Candidate-EPs before RANSAC* is more efficient in detecting lane as shown in Figure 11. (b) is using filtering and (c) is not. The results are that (b) and (c) generate 16 and 23 possible hypotheses respectively.

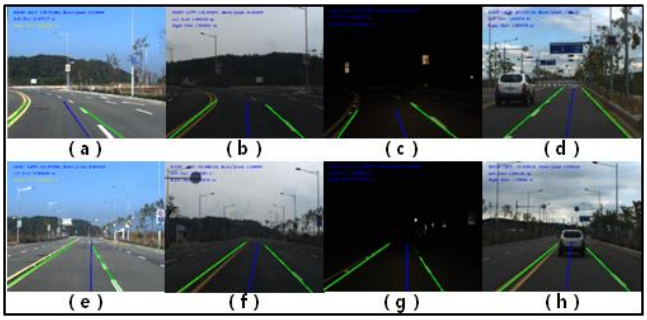

Figure 10. Results of lane-detection under various conditions.

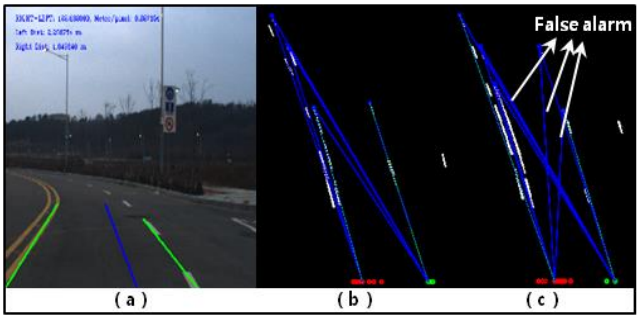

Figure 11. The comparison of two methods in generating hypotheses of lane during RANSAC. (a) is camera view of scene. (b) is an obtained result of RANSAC after filtering Candidate-EPs, and (c) is one of RANSAC-Only.

Therefore, it is obvious that (b) has less computation than (c) in generating and evaluating (or scoring) the hypotheses for lane. Moreover, (b) has less number of possible hypotheses that could result in false alarms than (c). The full experimental result of lane detection can be accessed at *http://www.hantw. com/itsc2014.html.*

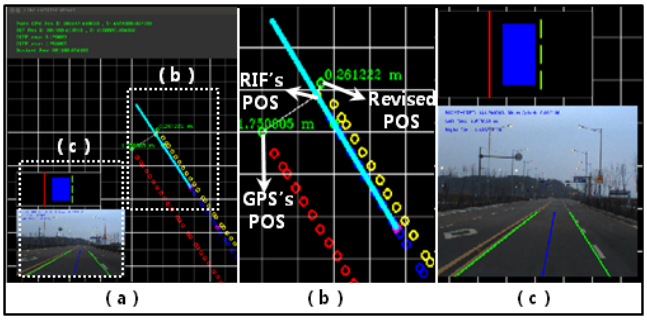

Figure 12. The comparision between GPS sensor's positions and revised positions. (a) is a scene of our visualizer. In (b), revised positions and GPS's positions are represented. In (c), upper image is vehicle's position on lane and bottom one is a screenshot of lane detection view.

Second, we analyze performance of the *localization algorithm* by comparing the ground truth with revised position and pure GPS sensor's position. Figure 12 is our analysis-tool that can record different information (lane detection, pure GPS position, revised position, and ground truth) simultaneously and analyze them. In the figure, *RIF's Pos* is referring ground-truth obtained from DGPS. The *Revised Pos* and *GPS's Pos* are the positions received from the *localization algorithm* and a GPS sensor respectively. Defining that the *position error* is the distance from *RIF's*

Pos to another position, we took results as shown Figure 13. The *x* and *y*-axis in the graph are representing *time* and *position errors* while driving the given path (from A to B in Figure 9). Specifically, the maximum and minimum position errors of GPS sensor are 1.75786 and 0.000352506 meters while 0.75786 and 0.000628652 meters are the errors of revised positions.

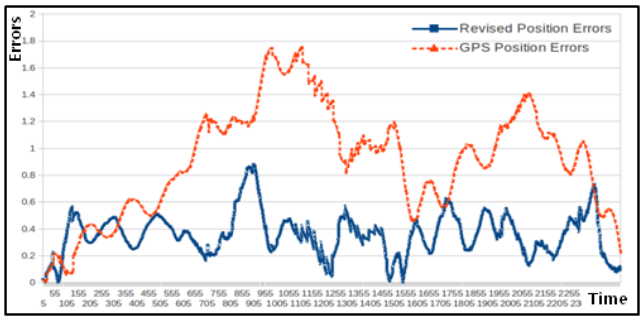

Figure 13. History of revised position errors and pure GPS position errors.

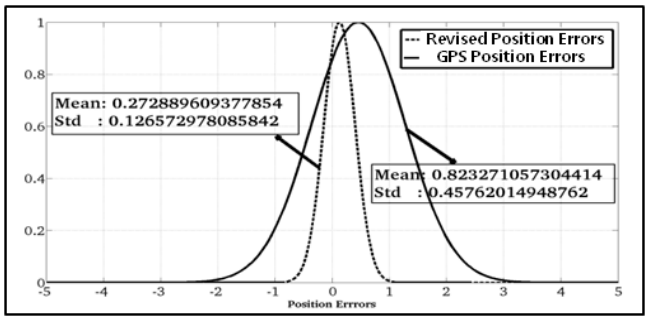

Figure 14. Gaussian distribution of position-errors.

Also, *Gaussian distributions* of the errors are representing that the revised position is more accurate (position errors: about 0.272±0.126 meters) than pure GPS sensor's (0.82 ±0.457 meters) as shown in Figure 14. The full experimental results are accessible at http://www.hantw.com/itsc2014.html. The movie clips are recorded while vehicle is driving autonomously. There are two results, one is driving with only GPS sensor's positions and another is driving with the revised positions. We can observe that the UGV invades another lane when driving with pure GPS sensor's positions, while it doesn't with the revised positions. Therefore, we conclude that the localization algorithm is appropriate to measure and overcome GPS sensor's position errors, and obtain more accurate positions. This is one of great localization solutions for UGV under urban environments.

VI. Conclusion & Future Work

This paper is suggesting novel lane detection and localization algorithm. As shown the experimental results, the lane detection algorithm performs well under various conditions, and the localization with lane information is efficient to overcome GPS sensor's position errors. Moreover, the method is easily applicable in industrial fields rather than using expensive sensors or heavy computations. However, the suggested algorithm has uncertainty of position on vehicle's *y*-axis (forward direction) as we described in section IV-(a). Therefore, we are trying to apply additional solutions such as stop-line or traffic-signal detection to reduce the uncertainty on *y*-axis and obtain more accurate position as a future work.

Acknowledgment

We would like to thank reviewers & robot technology group of SAMSUNG Techwin CO., LTD.

References

[1] Joydeep Biswas and Manuela Veloso, "Depth Camera Based Indoor Mobile Robot Localization and Navigation" in *2012 IEEE International Conference on Robotics and Automation*. RiverCentre, Saint Paul, Minnesota, USA, May 14-18, 2012.

[2] W.-K. Chen, "Visual topological SLAM and global localization*"* in *2009 IEEE International Conference on Robotics and Automation*. Kobe International Conference Center Kobe, Japan, May 12-17, 2009.

[3] H. Poor, "Robust Vehicle Localization in Urban Environments Using Probabilistic Maps" in *2010 IEEE International Conference on Robotics and Automation*. Anchorage Convention District, May 3-8, 2010, Anchorage, Alaska, USA.

[4] Sayanan Sivaraman and Mohan Manubhai Trivedi, "Integrated lane and vehicle detection, localization and tracking: a synergistic approach", in *IEEE Transactions on intelligent transportation system*, June 2013.

[5] Z. Tao et al., , "Mapping and localization using GPS, lane markings and proprioceptive sensors", in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.

[6] M. El Badaoui El Najjar and P. Bonnifait, "A road-matching method for precise vehicle localization using kalman filtering and belief theory", *Journal of Autonomous Robots, vol. Volume 19, no. Issue 2, pp. 173–191*, September 2005.

[7] Kim, ZuWhan, "Robust lane detection and tracking in challenging scenarios, *IEEE Transaction On Intelligent Transportation System, vol. 9,no. 1*, March 2008.

[8] Shengyan Zhou, Yanhua Jiang, Junqiang Xi, Jianwei Gong, Guangming Xiong, Huiyan Chen, "A Novel Lane Detection based on Geometrical Model and Gabor Filter", *2010 IEEE Intelligent Vehicles Symposium*, University of California, San Diego, CA, USA, June 21-24, 2010.

[9] Teawon Han, Nadeesha Ranasinghe, Luenin Barrios, and Wei-Min Shen. "An Online Gait Adaptation with SuperBot in Sloped Terrains", *IEEE International Conference on Robotics and Biomimetics (ROBIO 2012)*, Guangzhou, China, December 2012.