Proceedings of 2018 IEEE 8th Annual International Conference
on CYBER Technology in Automation, Control, and Intelligent Systems
July 19-23, 2018, Tianjin, China

Outdoor Navigation of a Mobile Robot by Following GPS Waypoints and Local Pedestrian Lane*

Jiatong Bao, Xiaomei Yao, Hongru Tang and Aiguo Song

Abstract— **Combining global GPS-based navigation and local pedestrian lane following is an promising way to autonomous navigation of mobile robots in unknown outdoor environments. By taking advantage of open SDK online satellite maps, the navigation route which contains a sequence of intersection waypoints could be autonomously planned. Through moving toward the GPS coordinates of the waypoints sequentially, the robot could be piloted to the destination in a global and coarse-grained navigation manner. Meanwhile, a simple vision-based lane following method is proposed to control the robot to move safely on the pedestrian lane in a local and fine-grained navigation manner. The two types of controllers are implemented as two behaviors which are managed by a Finite State Automaton (FSA) and coordinated with an obstacle avoiding behavior. The overall navigation system is implemented on the HunterBot platform with a cloud based human-robot interaction system. Elementary blind test results on the campus environments have shown that the proposed method is potentially effective to address the autonomous navigation problem in unknown outdoor environments.**

I. Introduction

Autonomous navigation in large scale outdoor environments is a key capability that many intelligent mobile platforms should have. It plays an important role in the technologies of self-driving car [12], autonomous driving wheelchair [13], navigation assistance for the blind [8], etc.

Researches have paid more attention to the high-speed automobiles that drive on motorized vehicle lanes [3] than the low-speed vehicles that travel on pedestrian lanes [2]. Mapping and localization are two fundamental components in the autonomous navigation systems. The accuracies of mapping and localization are the most critical for the high-speed vehicles, without which it would be very dangerous to others. Thus different kinds of sensors such as laser scanners [9] and stereo cameras [5] are used to build the accurate maps of outdoor environments. Many challenges still exist in the mapping and localization fields. As for the low-speed vehicles such as wheelchairs, there is no need to equip itself with high-cost sensors and spend too much efforts to build accurate maps. Most importantly, either the high-speed or low-speed vehicles are always required to navigate in unknown areas. When there is no built maps in advance, vehicle localization and route planning in those areas would be very difficult.

With the availability of many open SDK online satellite maps, navigating a mobile vehicle in large scale unknown areas by using online maps and consumer-level GPS receiver becomes a feasible and economical way. However, the positioning errors between maps and GPS receivers could only provide the vehicle with coarse-grained navigation service. To ensure that the vehicles could travel safely on the desired lanes, robust local navigation methods should be investigated and integrated. Lane segmentation [7] and lane border detection [6] are two main categories of methods commonly used for lane detection [2]. How to effectively combine global GPS based navigation with local lane following becomes a very interesting topic.

This paper studies the autonomous outdoor navigation problem on a low-speed mobile robot platform, aiming at combing global GPS-based navigation and local pedestrian lane following together in order to provide a simple and effective way to address the problem. By taking advantage of open SDK online satellite maps, the navigation route which contains a sequence of intersection waypoints could be autonomously planned. Through moving toward the GPS coordinates of the waypoints sequentially, the robot could be piloted to the destination in a global and coarse-grained navigation manner. Considering the low-speed vehicles should travel in pedestrian environments, a simple pedestrian lane following method based on Vanishing Point (VP) detection and robot position offset calculation is then proposed in order to control the robot to move safely on the pedestrian lane in a local and fine-grained navigation manner.

With the proposed method, we do not need to spend too much efforts on building maps in advance and solving many challenging problems in mapping and localization. The elementary blind test results on the campus environments have shown that the proposed method is potentially effective to address the autonomous navigation problem in unknown outdoor environments. With the advantages of low cost and easy-to-implement, it will be very meaningful to many applications such as autonomous wheelchair navigation, unknown space exploration, etc.

II. Problem Statement

A four-wheel differential driving tracked mobile robot, named HunterBot, is used in this work. As shown in Figure 1, the HunterBot is equipped with different types of sensors such as RGB-D camera, GPS receiver, compass, Inertial Measurement Unit (IMU), etc. The coordinate frames of the

*This research work is partially supported by the Postdoctoral Science Foundation of Jiangsu Province (No. 1601006C) and National Natural Science Foundation of China (No. 61325018 and 61272379).

Jiatong Bao and Aiguo Song are with the School of Instrument Science and Engineering, Southeast University, Nanjing 210096, China (E-mails: jtbao@yzu.edu.cn, a.g.song@seu.edu.cn)

Xiaomei Yao and Hongru Tang are with the School of Hydraulic, Energy and Power Engineering, Yangzhou University, Yangzhou 225000, China (E-mails: 284079480@qq.com, hrtang@yzu.edu.cn)

978-1-5386-7057-6/18/$31.00 © 2018 IEEE

camera, the robot base and the global reference system (e.g. the world or the map) are defined as shown in the Figure 2. The kinematic model of the robot is

$$\begin{aligned} \dot{x} &= v cos\alpha \\ \dot{y} &= v sin\alpha \end{aligned} \tag{1}$$

where $(\dot{x}, \dot{y})$ is the velocity in the global reference frame, v is the linear velocity of the robot and α is the robot heading angle with respect to the x-axis of the global reference frame. In addition, the angular velocity, linear velocity acceleration, angular velocity acceleration, maximal linear velocity and maximal angular velocity of the robot are denoted by ω, $\dot{v}$, $\dot{\omega}$, v_{max} and ω_{max}, respectively.

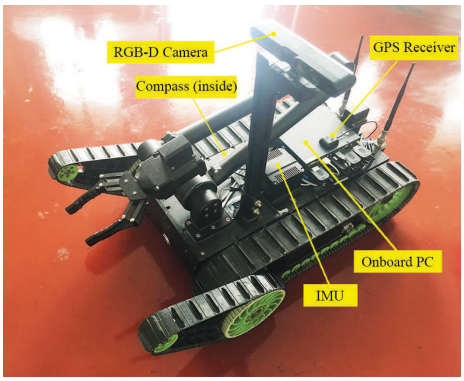


Fig. 1. The HunterBot mobile robot platform.

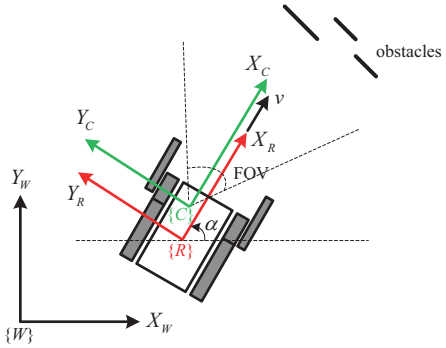


Fig. 2. The coordinate frames defined in the mobile robot system. $\{C\}$, $\{R\}$ and $\{W\}$ are the coordinate frames of the camera, the robot base and the global reference system, respectively.

As shown in Figure 3, the navigation problem is how to pilot the HunterBot to achieve a specific destination autonomously and safely in outdoor pedestrian lane environments. To achieve a specific outdoor destination S_N, the HunterBot is navigated globally from the start position S_0 along a sequence of intersection waypoints $S = \{S_0, S_1, \cdots, S_N\}, S_i = (S_i^{lon}, S_i^{lat}), i \in [1, N]$, where S_i^{lon} and S_i^{lat} are the longitude and latitude of the waypoint S_i defined in the WGS84 coordinate frame. The sequence of waypoints could be manually defined or autonomously generated by online maps such as Google maps, Amap [1], etc. The coordinate systems used by the online maps should be transformed to the WGS84 system accordingly. The current robot position could be directly collected from the GPS receiver where the longitude and latitude are always represented in the WGS84 coordinate frame. To get robust robot position, extended Kalman filters are used to estimate the robot position by fusing data from GPS receiver, odometry and IMU [10]. All locations are finally represented in the Universal Transverse Mercator (UTM) coordinate frame which is a planar coordinate frame.

Fig. 3. Illustration of the navigation problem.

The navigation task could then be decomposed into a sequence of sub-tasks, each of which is to navigate the robot from the current position S_{robot} to the next position S_i by controlling the linear and angular velocities of the robot in the UTM coordinate frame. Once the robot reaches the last waypoint S_N, it means that the last sub-task is accomplished and thus the overall task is completed. The overall task is decomposed based on our previous work [11] and is shown in Figure 4.

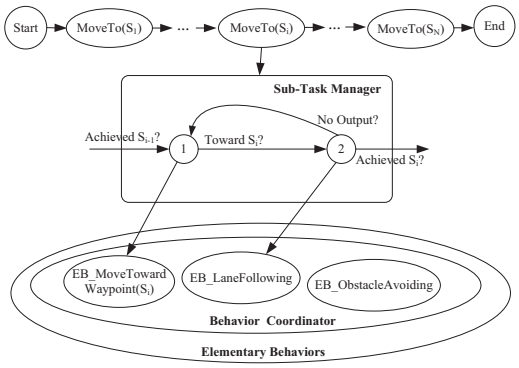


Fig. 4. Task decomposition of the outdoor navigation problem.

The behavior based robot control architecture is used to implement the sub-tasks. A sub-task will execute three kinds of elementary behaviors in parallel, i.e. $EB\_MoveTowardWaypoint$, $EB\_LaneFollowing$ and $EB\_ObstacleAvoiding$. The first behavior provides a global and coarse guidance for the robot moving toward the next waypoint. However, the robot's safety can not be guaranteed only by using this behavior, because the robot localization

accuracy, which highly depends on the online maps and the GPS receiver, typically has errors in several meters. The path between two waypoints is also not always straight (see the path between S_3 and S_4 in Figure 3). The second behavior ensures that the robot should move along the pedestrian lane and do not move off the lane where may be unsafe for the robot. The third behavior is to help the robot move away from pedestrians, cars or other types of obstacles that are also located on the way. Since all three behaviors want to take control of the robot, their outputs will be arranged by the sub-task manager in a specific sequence and coordinated by a behavior coordinator.

III. Method Description

A. Moving toward a GPS Waypoint

To control the robot moving toward a specific GPS waypoint, the control structure shown in Figure 5 is designed based on the following principles.

- If the heading direction of the robot with angle α deviates from the route direction with angle β to a large extent, the drift angle δ should be reduced as the main goal of the controller.
- If the robot is far away from the next waypoint with distance d, the controller should make it reach the position with minimal time.
- The robot should slow down as it is approaching at the waypoint and then stop when it achieves the position with a tolerable distance.

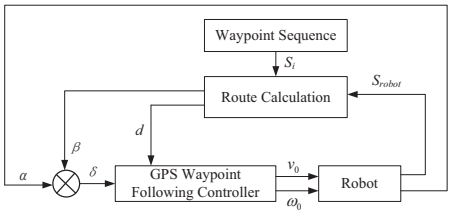


Fig. 5. The control structure for navigating the robot from the current position to the next waypoint.

As shown in Figure 6, suppose that a heading angle of 0 means the robot is facing east, and that the heading angle increases counter-clockwise while it decreases clockwise. Therefore, the robot heading angle α and the route direction angle β both vary in $(-\pi, +\pi]$. The drift angle of the robot δ could then be calculated as

$$\delta(\alpha,\beta)=\begin{cases} \beta-\alpha, & |\alpha-\beta|\le\pi \\ \frac{\alpha-\beta}{|\alpha-\beta|}\left(2\pi-|\alpha-\beta|\right), & |\alpha-\beta|>\pi \end{cases} \quad (2)$$

In addition, α comes from the compass of the robot, and β is calculated by the route calculation module according to the coordinates of S_{robot} and S_i.

The P-controller is used to provide as command of angular velocity ω_0 to the robot that is proportional to the drift angle δ, as formulated in

$$\omega_0 = K_\omega \delta, K_\omega > 0 \quad (3)$$

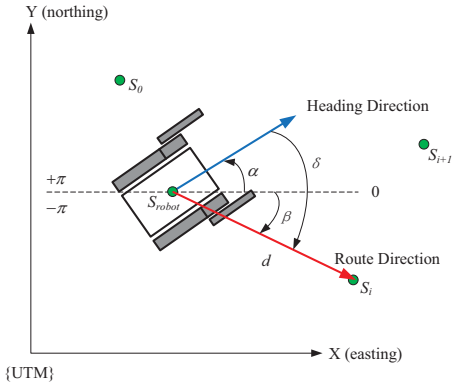


Fig. 6. Illustration of the robot status when moving toward the next waypoint.

where K_ω is a suitable constant gain factor. ω_0 is then limited by the maximal angular velocity ω_{max}.

When the robot has large drift angle, the command of linear velocity should be limited and determined by

$$v_0 = e^{-\lambda_{v0}|\delta|} v_d \quad (4)$$

where λ_{v0} is a tuning parameter, v_d is the desired linear velocity of the robot according to its distance from the target, which is calculated by

$$v_d = \begin{cases} v_{max}\frac{d}{d_{warn}} & d \le d_{warn} \\ v_{max} & d > d_{warn} \end{cases} \quad (5)$$

where d is the distance between S_{robot} and S_i, and d_{warn} is the distance threshold that warns the robot it is approaching to the target.

B. Local Pedestrian Lane Following

The task of following a local pedestrian lane is to help the robot move along the lane by taking advantage of the vision information extracted from the local area. Figure 7 shows the control structure for the lane following task. The vision system iteratively takes a $W_I \times H_I$ RGB image which is fed into the local path line calculation module. This calculation module comprises two main steps: (i) lane line detection, and (ii) calculation of local path line to follow.

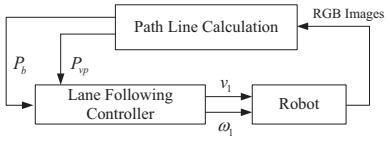


Fig. 7. The control structure for the lane following task.

In the first step, the module first selects as a Region of Interest (ROI) the RGB image part that is below a reference line. The reference line L_{ref} is above the manually calibrated horizon line $L_{horizon}$ with a distance d_t in pixels. $L_{horizon}$ can be corrected online according to the pitch angle of the camera system. Canny edge detection is then performed on the ROI to create an edge map. Progressive probabilistic

hough transform is applied on the edge map to detect lines. The horizontal and vertical lines are further filtered out. All detected lines are used to vote for the VP candidates on the horizon line and to extract the lane lines that contribute to the corresponding VP. In the second step, the local path line is calculated based on the already detected lane lines and represented in the image coordinate frame. Without calibration of extrinsic and intrinsic camera parameters, the position and orientation of the path line would relatively reflect the robot heading and position with respect to the lane center well, and thus can be taken advantage of for robot control. Concretely, the path line is extracted by considering two cases:

In case of the VP candidate set is not null. The VP candidate with the highest votes is selected as the true VP denoted by P_{vp}. The M supporting lane lines are denoted by $L_{lane} = \left\{L^i_{lane}\right\}, i = 1, \cdots, M$, where the equation of L^i_{lane} is $a_i u + b_i v + c_i = 0$ defined in the image coordinate frame. Then, the lane lines are divided into two sets

$$L_{lane} = L_{left} \cup L_{right} \tag{6}$$

where any line $L^i_{lane} \in L_{left}$ satisfies $k_i = -\frac{a_i}{b_i} < 0$ and any line $L^j_{lane} \in L_{right}$ satisfies $k_j = -\frac{a_j}{b_j} > 0$.

When $L_{left} \neq \varnothing \wedge L_{right} \neq \varnothing$, the two nearest lines, i.e. L_l and L_r, to the left and right side of the robot are selected from L_{left} and L_{right} respectively by

$$\begin{aligned} L_l &= \text{argmin}_{L^i_{lane} \in L_{left}} k_i \\ L_r &= \text{argmax}_{L^j_{lane} \in L_{right}} k_j \end{aligned} \tag{7}$$

As shown in Figure 8 (a), the local path line L_p is then calculated by

$$\begin{aligned} v &= ku + b \\ k &= -\frac{2}{k_l + k_r} \\ b &= \frac{2u_{vp}}{k_l + k_r} + v_{vp} \end{aligned} \tag{8}$$

where k_l and k_r are the slopes of L_l and L_r respectively, and (u_{vp}, v_{vp}) is the image coordinate of P_{vp}. Suppose the intersection of the path line and the image bottom border line is denoted by P_b, its image coordinate can be calculated by

$$(u_b, v_b) = \begin{cases} \left(\frac{H_I - b}{k}, H_I\right) & k_l + k_r \neq 0 \\ \left(\frac{W_I}{2}, H_I\right) & k_l + k_r = 0 \end{cases} \tag{9}$$

In other words, the local path line is determined by two image points P_{vp} and P_b. The horizontal deviation of P_{vp} from the image center reflects the robot heading deviation from the lane heading, while the horizontal deviation of P_b from the image center reflects the robot position deviation from the lane center.

When $L_{left} = \varnothing \wedge L_{right} \neq \varnothing$ or $L_{left} \neq \varnothing \wedge L_{right} = \varnothing$, P_{vp} is generated by one side supporting lane lines, as shown in Figure 8 (b). We then assume $P_b = P_c = (\frac{W_I}{2}, H_I)$ which means the robot position deviation is zero.

In the cases where L_p is detected, the control goal is to keep P_{vp} stay in the horizontal image center and the robot position deviation be zero. Since the robot deviation from the lane center is a combination of its heading deviation and position deviation, the control law of the robot angular velocity is define by

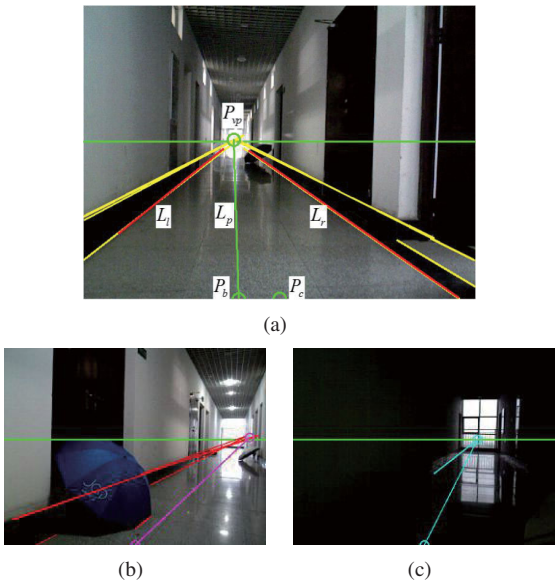
$$\omega_1 = \frac{(W_I - u_b - u_{vp})}{W_I} \omega_{max} \tag{10}$$

The linear velocity for control is then determined empirically by

$$v_1 = e^{-\lambda_{v1}|\omega_1|} v_{max} \tag{11}$$

based on the basic consideration that the robot should not move fast when it is turning. In addition, the exponential function curve in Eq. (11) could be tuned by λ_{v1} according to the maximal velocities of the real robot.

In case of the VP candidate set is null. This case happens when there is no intersection of any two lane lines lies on the horizon line. As shown in Figure 8 (c), if lane lines exist, we empirically randomly select one lane line denoted by L_s, and generate P_{vp} as the intersection of L_s and $L_{horizon}$, and set $P_b = P_c$. Therefore, the angular and linear velocity for robot control can be calculated by using Eq. (10) and (11). In other cases, the controller will generate no output and the robot will be taken over by other controllers.



(a)

(b) (c)

Fig. 8. Typical situations for local path line calculation. (a) and (b) show the path line can be determined by two side and only one side lane lines that supports the VP, respectively. (c) shows there is no intersection of any two lines lies on the horizon line and the path line is calculated by using one randomly selected lane line.

C. Depth Camera based Obstacle Avoidance

When navigating in pedestrian lane environments, the robot should avoid the obstacles such as pedestrians and cars as well. The obstacles are detected by using the fake laser scans that converted from the depth image which comes from the RGB-D camera. As shown in Figure 2, the locations of the detected obstacles are first represented in the camera coordinate frame $\{C\}$ and then transformed to the robot base coordinate frame $\{R\}$ on which the local 2D map is built. On

the map, the positions where the obstacles locate are marked as not traversable.

Based on the local map, the dynamic window approach [4] is used to determine the possible linear and angular velocities for control according to the current velocities of the robot. The dynamic windows is defined as

$$\begin{aligned} V_d &= \{(v_c, \omega_c)\} \\ v_c &\in [\max\{0, v - \dot{v}\Delta t\}, \min\{v_{max}, v + \dot{v}\Delta t\}] \\ \omega_c &\in [\max\{0, \omega - \dot{\omega}\Delta t\}, \min\{\omega_{max}, \omega + \dot{\omega}\Delta t\}] \end{aligned} \tag{12}$$

where Δt is the time interval during which the accelerations are applied. Each velocity (v_c, ω_c) is then evaluated whether it is viable for controlling the robot without obstacle collision. By applying each velocity on the robot kinematic model (see Eq. 1), the trajectory which starts from the robot's current position is predicted with a specific time period. If the minimal distance between the trajectory and the obstacles is less than half of the robot base width, the trajectory would be not applicable and the corresponding velocity be not viable. In this work, the obstacle avoidance behavior does not directly take control of the robot, whereas outputs a list of viable velocities for control.

D. Behavior Coordination

As shown in Figure 4, $EB\_MoveTowardWaypoint$ and $EB\_LaneFollowing$ are coordinated by the sub-task manager. The FSA is used to select the corresponding behavior output for taking control of the robot in a event driven manner. Concretely, if the robot reaches the waypoint S_{i-1}, $EB\_MoveTowardWaypoint$ is triggered to guide the robot. After the robot moves toward the next waypoint S_i in the right direction, $EB\_LaneFollowing$ will take effect until it generates no output or the robot achieves S_i. Either $EB\_MoveTowardWaypoint$ or $EB\_LaneFollowing$ is further coordinated with $EB\_ObstacleAvoiding$, providing only one pair of linear and angular velocities for control. The basic idea is to select from the viable velocity set the final velocity for control which satisfies the control demand of $EB\_MoveTowardWaypoint$ or $EB\_LaneFollowing$ to a large extent. Therefore, the output of the coordinator is calculated by

$$(v_2, \omega_2) = \underset{(v_c, \omega_c) \in V_d}{\operatorname{argmin}} \left\{ \lambda \cdot \frac{|v_c - v_{01}|}{\sum |v_c - v_{01}|} + \rho \cdot \frac{|\omega_c - \omega_{01}|}{\sum |\omega_c - \omega_{01}|} \right\} \tag{13}$$

where (v_c, ω_c) comes from the obstacle avoidance behavior, (v_{01}, ω_{01}) is the output of $EB\_MoveTowardWaypoint$ or $EB\_LaneFollowing$, λ and ρ are the parameters for balance of the two types of velocity bias.

IV. Experimental Results

We implemented all above robot behaviors and behavior coordinators within the ROS framework running on the on-board PC of the HunterBot. The parameters are set as $v_{max} = 1m/s$, $\omega_{max} = 20deg/s$, $K_\omega = 0.5$, $\lambda_{v0} = 2$, $\lambda_{v1} = 5$, $\lambda = 0.5$ and $\rho = 0.5$. A cloud based human-robot interaction system was also developed. In the system, the robot was continuously sending the collected sensor data (i.e. GPS coordinates, velocities, heading direction, etc.) to the cloud server, while waiting for the navigation command with planned waypoints. The human-robot interaction GUI embedded with the online map was loaded from a web browser. The user could monitor the robot status and select the goal position. After clicking on the map and confirming the point to be the destination, the navigation route was then autonomously planned. The navigation command with waypoint coordinates was then generated and sent to the cloud server. The cloud server then routed the command to the robot immediately.

A. Experimental Scenario

As shown in Fig.9, the robot was originally placed on the lane to the west of a building with a heading to the east. The goal position is to the east of that building. The navigation path shown as the blue dashed lines was autonomously planned by the Amap [1], while two intermediate waypoints were generated. The path segment between the two waypoints is a curved one. Other two path segments are straight.

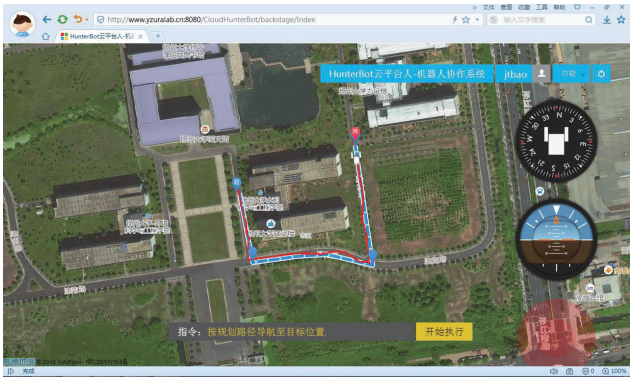
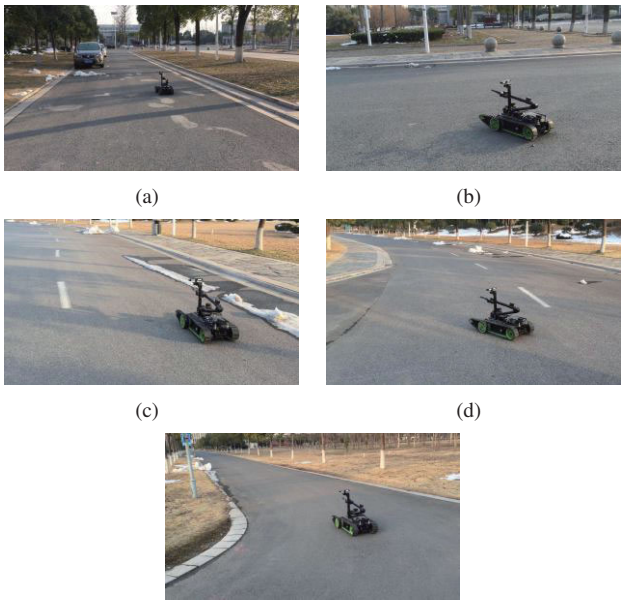


Fig. 9. Autonomous navigation results in an outdoor scenario.

B. Results and Discussion

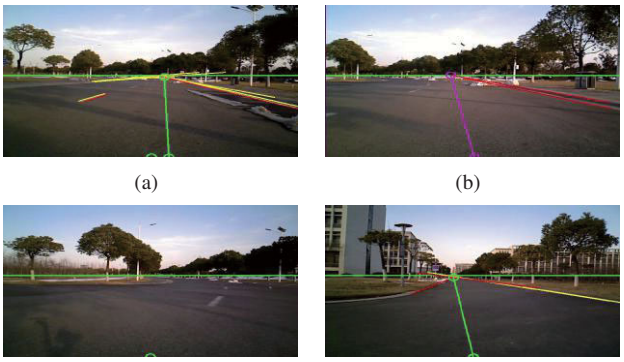
During the experiment, the coordinates of the robot were periodically recorded and displayed on the map, resulting in the overall navigation trajectory shown as the red solid lines in Fig.9. It shows that the proposed method could successfully navigate the robot from the starting position to the assigned destination. We also recorded the real navigation performance of the robot with a hand-held camera. Fig.10 shows some video frames verifying that the robot was doing the right actions as expected. Fig.11 shows typical cases when the robot was mainly navigated by the vision algorithm based lane following behavior.

However, the proposed method may suffer several potential shortcomings. Firstly, the current simple vision algorithm based lane following behavior may not be robust enough. As observed in the experiment shown in Fig.12 (a), the robot bumped against the curb especially when it was approaching the first waypoint. This happens when the lane following behavior generates unstable local lane path since it could take



(a) (b) (c) (d) (e)

Fig. 10. Some recorded video frames that show the HunterBot doing the right actions as expected during the experiment. The robot firstly traveled on the first straight lane guiding by the lane following behavior (a). When it achieved the first waypoint, its heading was being adjusted toward the next waypoint (b) controlled by the waypoint following behavior. Then the robot kept moving on the curved lane (c) until achieving the second waypoint and adjusting its heading (d). The robot moved along the second straight lane (e) and finally arrived at the goal position.



(a) (b) (c) (d)

Fig. 11. Typical local lane path generated by the lane following behavior in the experiment. (a) and (b) show the results when the robot moved along the curved lane segment. When the robot almost reached the road intersection (i.e. the second waypoint), the vision features were lost (c). Thus, the robot was taken controlled by the global waypoint following behavior until reaching the waypoint. After rotating its head to the next waypoint, a new lane was found (d) and the robot was controlled again by the lane following behavior.

(a) (b)

Fig. 12. Undesired situations the robot may come across. The robot was prone to bump against the curb (a) and rotate its heading to the next waypoint in advance (b).

many noise image features into account in the intersection. Secondly, there does exist positioning error between the robot and the planned waypoints. The robot may assert that it has arrived at the waypoint at an inappropriate time, as shown in Fig.12 (b).

V. CONCLUSIONS

This paper investigates the method of combining GPS based global waypoint following and vision algorithm based local pedestrian lane following for autonomous outdoor navigation. The overall navigation system is implemented on the HunterBot platform communicating with a cloud server that could provide web interface for the users. The effectiveness of the method is demonstrated by the experimental results. Generally, our work shows a novel and simple way to navigate the robot traveling in unknown outdoor environments with no need of building maps. Future work could investigate more robust vision algorithms for enabling the robot to understand well whether it is on the lanes or the intersections, thus the positioning errors would be ignored.

REFERENCES

[1] Amap.com. Amap open platform. http://lbs.amap.com/, 2017.

[2] C. Chang, C. Siagian, and L. Itti. Mobile robot monocular vision navigation based on road region and boundary estimation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1043–1050, 2012.

[3] Z. Chen and X. Huang. End-to-end learning for lane keeping of self-driving cars. In *IEEE Intelligent Vehicles Symposium*, pages 1856–1860, 2017.

[4] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics Automation Magazine*, 4(1):23–33, 2002.

[5] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[6] A. Y. Hata, F. S. Osorio, and D. F. Wolf. Robust curb detection and vehicle localization in urban environments. In *Intelligent Vehicles Symposium Proceedings*, pages 1257–1262, 2014.

[7] J. Kim and C. Park. End-to-end ego lane estimation based on sequential transfer learning for self-driving cars. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1194–1202, 2017.

[8] M. Martinez, A. Roitberg, D. Koester, R. Stiefelhagen, and B. Schauerte. Using technology developed for autonomous cars to help navigate blind people. In *IEEE International Conference on Computer Vision Workshops*, pages 1424–1432, 2017.

[9] F. Maurelli, D. Droeschel, T. Wisspeintner, S. May, and H. Surmann. A 3d laser scanner system for autonomous vehicle navigation. In *International Conference on Advanced Robotics*, pages 1–6, 2009.

[10] T. Moore and D. Stouch. A generalized extended kalman filter implementation for the robot operating system. *International Conference on Intelligent Autonomous Systems*, pages 335–348, 2016.

[11] H. Tang, A. Song, and X. Zhang. Hybrid behavior coordination mechanism for navigation of reconnaissance robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1773–1778, 2007.

[12] S. Thrun. Toward robotic cars. *Communications of the ACM*, 53(4):99–106, 2010.

[13] M. Yokozuka, Y. Suzuki, N. Hashimoto, and O. Matsumoto. Robotic wheelchair with autonomous traveling capability for transportation assistance in an urban environment. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2234–2241, 2012.