2018 IEEE Intelligent Vehicles Symposium (IV)
Changshu, Suzhou, China, June 26-30, 2018

Toward Autonomous Driving in Highway and Urban Environment: HQ3 and IVFC 2017

Lilin Qian[1], Hao Fu[1], Xiaohui Li[1], Bang Cheng[1], Tingbo Hu[1], Zengping Sun[1], Tao Wu[1], Bin Dai[1], Xin Xu[1] and Jin Tang[2]

***Abstract*— The 2017 Intelligent Vehicle Future Challenge of China (IVFC) was held in Changshu between 24th November and 26th November, 2017. As the ninth series of this event, last year's competition has introduced many new features and has attracted 21 teams to join this competition. The HQ3 autonomous vehicle, jointly developed by National University of Defense Technology, Jilin University and Central South University, took part in this competition. This paper mainly describes the key modules of HQ3, including GPS-free localization, environment perception and behavior planning. All of these modules together enable HQ3 to perform well during the competition.**

I. INTRODUCTION

The passed three decades have witnessed tremendous progress of autonomous driving. Many competitions have been held, which greatly promote technological progress and have also attracted a lot of attention from the whole society.

The Defense Advanced Research Projects Agency (DARPA) organized two Grand Challenge for autonomous vehicle in 2004 and 2005. Both competitions were held in the desert region [1]. The third DARPA challenge, also known as the Urban Challenge, took place in 2007. This competition was held in city area, which required the autonomous vehicle to obey traffic rules and interacting with other traffic participants [2].

To accelerate the developments in cooperative driving in realistic traffic scenario, the Grand Cooperative Driving Challenge (GCDC) [3] was hold in Helmond, the Netherlands in 2016. By introducing the vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2X) communication, the autonomous vehicle could behave more safely and handle more complicated scenarios.

Sponsored by National Natural Science Foundation of China, the Intelligent Vehicle Future Challenge (IVFC), has been held for nine times since 2009. The 2017 IVFC was recently held in Changshu, Jiangsu. Compared to its previous series, last year's competition has introduced many new tests, including driving in real traffic flow, driving across a rough country road, navigating in the tunnel, passing through the toll station, etc. All of these newly introduced features pose significant challenges to the autonomous vehicle,making the 2017 IVFC the hardest one in its series.

*This work was supported by National Natural Science Foundation of China under Grant 61503400, 61375050 and 91220301

[1]Lilin Qian, Hao Fu, Tingbo Hu, Tao Wu, Bin Dai, Xin Xu are with College of Intelligence Science and Technology, NUDT, Changsha, China qianlilin1989@gmail.com

[2]Jin Tang is with School of Infromation Science and Engineering, Central South University, Changsha, China.

The jointly developed HQ3 autonomous vehicle has taken part part in the IVFC 2017 competition. As shown in Fig.1, the HQ3 autonomous vehicle is equipped with different kinds of sensors, including lidar, radar, camera, wheel encoders, inertial measurement unit (IMU), GPS receiver, etc. In the remainder of this paper, we will review the key module design of HQ3.

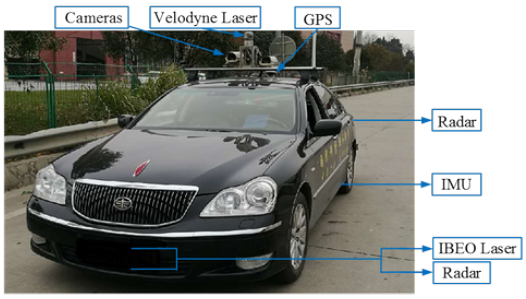


Fig. 1. The HQ3 autonomous vehicle and sensors installed in IVFC 2017.

II. HIGH-PRECISION MAP BASED GPS-FREE NAVIGATION MODULE

It has been a common practice to utilize High-Precision Map (HPM) to enhance the environment perception and navigation capability for the autonomous vehicle. HPM can aid the autonomous vehicle for at least three aspects [4]: Firstly, by matching the current sensor information to the HPM, the autonomous vehicle could be accurately localized on the map, regardless of the possible unreliability of GPS signals. Secondly, by manually annotating the objects(eg. traffic lights, traffic signs, lane markings, road curbs, etc) on the map, an accurate static prior of interest could be obtained. Thirdly, the shape of the road obtained from the map could also help to track other dynamic traffic participants and predict their behaviors.

The HPM we used is generated offline with graph SLAM (Simultaneous Localization And Mapping) algorithm. The graph SLAM algorithm tries to optimize a pose graph, which contains a lot of factors. Each factor either poses constraint on a single pose or several poses when they share observations. The pose graph is then optimized using the incremental Smoothing And Mapping (iSAM) approach proposed in [5]. Based on the optimized poses, the lidar scans are assembled to generate the map.

There exist different forms of HPM, such as the obstacle map, the 2D height map, the 2D reflectance intensity map

and the 3D point cloud map. As the autonomous vehicle mostly runs on flat road, it is not always necessary to use the 3D point cloud map, which is both memory consuming and computatio demanding. Therefore, the 2D height map and the 2D reflectance intensity map can be a better choice. In using the 2D map, as shown in Fig.2, the autonomous vehicle is assumed to run on a planar surface, and its pose is simplified to 3 DoF (Degree of Freedom): x, y and $azimuth$. However, when the vehicle runs on rough terrain or there exist large fluctuations in its pitch angle, we prefer the 3D point cloud map and the its pose extends to full 6 DoF (x,y,z,$azimuth$,$pitch$ and $roll$) domain. An illustrative example is shown in Fig .3.

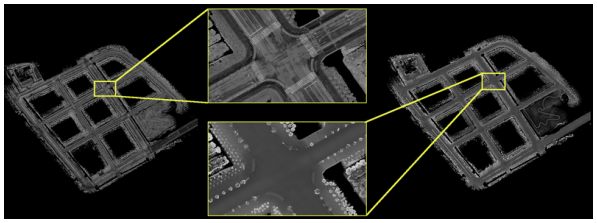


Fig. 2. The 2D reflectance intensity map and the 2D height map.

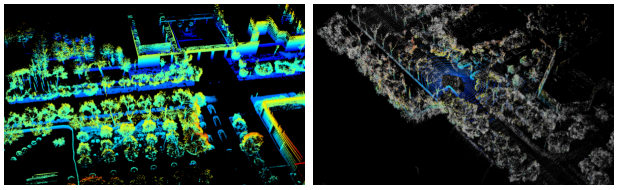


Fig. 3. The left figure is the 3D point cloud map generated offline using Graph SLAM algorithm. The right figure illustrates how the current scan (represented as colored dots) is registered to the 3D map (the gray dots).

For the localization module, we adopt a recursive Bayesian estimation framework. The output of the wheel encoder models the incremental motion between two vehicle poses as a prior term. The likelihood term is calculated by comparing the current scan with the HPM. We could obtain the Maximum A-Posterior (MAP) estimate of the current pose according to Bayes theorem. To handle the non-linearity of the system and avoid the linearisation error of the Extended Kalman Filter (EKF), we use Particle Filter to track the poses.

Two challenging scenarios that occurred on the second day of IVFC 2017 are shown in Fig.4. In one scene, as shown in the top-left of Fig.4, the road is covered with a roof. The roof height is 4 meters, allowing the autonomous vehicle to pass through it, but the roof itself will block the GPS signal. The aim of setting up this scene is to test the autonomous vehicle's capability of GPS-free navigation. In another scene, as shown in the bottom-left of Fig.4, the autonomous vehicle is required to pass through a curvy country road. The country road is built in the open area, with few man-made structures or distinctive features around. The GPS signal in this area might also get disturbed. This featureless and structureless environment poses strong challenges for HPM based localization approaches, which are usually based on distinctive structures or features on the road or the roadside.

In both of these two scenarios, our 2D HPM based localization module performs well, as shown in the top-right and bottom-right of Fig.4. The gray-scale part in these two sub-figures is the prebuilt 2D height map, and the colored dots are the currently-observed scan. The small green circle in the center of these two sub-figures represents the estimated vehicle position in the localization module, whilst the GPS position (obtained from a high-end Novatel Integrated Navigation System which couples the GPS signal with inertial measurement unit) is shown as the red circle. Note that in the top-right figure, the GPS position deviates from the localization output, as the GPS signal is blocked by the roof, making the output of the Integrated Navigation System (INS) deviates from the true position.

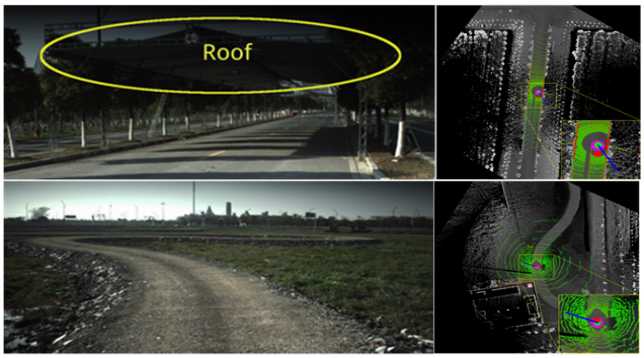


Fig. 4. Two challenging scenarios for perception occurred in IVFC 2017. The top row shows a road covered. In both scenarios, the 2D HPM based localization module performed well irrespective of the GPS signal blockage or the flat featureless environment.

Just as we mentioned, another useful property of the HPM is to aid the environment perception module. Fig.5 is an illustrative example, where we manually annotate all the topologically connected lane markings with different attributes(e.g. white or yellow and dotted or solid, etc) on the HPM. Besides the lane markings, we can also annotate traffic light, traffic sign, zebra crossing, etc. All these manually labeled information is stored in a database. When the autonomous vehicle runs online, its position is used as a query to the database, and the database will return all the labeled information nearby. Those information could be then utilized by the environment perception module.

III. Environment Perception Module

The environment perception module of the autonomous vehicle is in charge of processing all the sensor information, and outputing fused perception result to the planning module. It mainly consists of the following tasks:

A. Road Understanding

The task of road understanding is not merely to detect the road area, but to detect the drivable path which satisfies a lot of constraints, including the vehicle dynamics constraint, safety constraint, traffic rule constraint, etc. This task could be divided into the following sub-tasks:

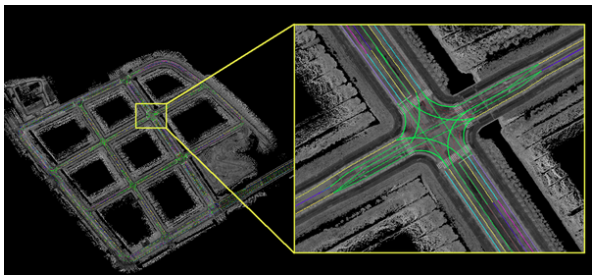


Fig. 5. The left is the 2D HPM built for the IVFC'2017, with all the annotated lane markings overlaid on it. A zoomed in view is shown on the right.

(a). Road Detection. The aim of this task is to detect the road area either based on the geometric property obtained from the lidar or the texture information from the image. An illustrative example [6] is shown in Fig.6 top, where the lidar points is classified into three semantic classes, road (colored in yellow), obstacle (colored in red) and hanging objects (colored in green). Fig.7 right shows the road detection result using image sensor, where the road area is colored in red. We have also developed a principled method [7] by fusing the image with the lidar in a Conditional Random Field (CRF) framework, which obtains a better road detection result.

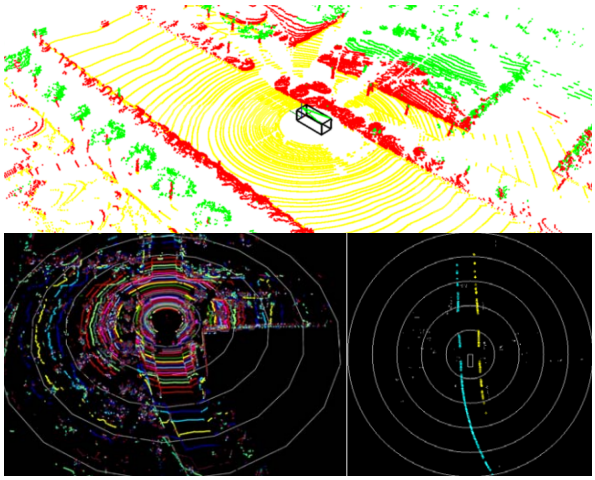


Fig. 6. The top figure shows the road detection result of a lidar scan [6]. The road area is colored in yellow, the obstacles and hanging objects are colored as red and green respectively. The bottom left figure is a lidar scan, whose scan lines are colored differently. By independently processing each scan line, we can detect the road curb [9] as shown in the bottom right figure.

(b). Road Boundary Detection. Although the road detection algorithm will produce the byproduct of road boundary, it is sometimes necessary to consider the road boundary detection as an independent problem. Road boundary could either be well defined as a road curb or implicitly defined by a change of slope or image texture as in the country road shown in Fig.4 bottom. In both cases, the road boundary is a weak signal that should be took particular care of. As shown in Fig.6 bottom right [9] and Fig.7 left [8], we have developed road boundary detection methods that could detect the road curb up to 50 meters away using lidar or image sensor in real-time.

Fig. 7. From left to right: the detection result of the road boundary, lane markings and road area.

(c). Lane Marking Detection. Lane marking plays a vital role in the autonomous vehicle. We could either directly detect it from the image sensor, or we could use rich information of the HPM. Through localizing the autonomous vehicle on the HPM, the human annotated lane markings could be projected onto the current image. They could then be utilized to verify the detection result.

B. Static Object Detection

It is essential for the autonomous vehicle to be capable of detecting and recognizing traffic related objects, including traffic signs, traffic lights, ground signs, stop lines, zebra crossings, etc.

For the traffic sign detection, we have constructed the Chinese Traffic Sign Detection Benchmark (CTSDB) [10], which contains 12224 images with 17013 bounding boxes annotated. Based on this dataset, we have developed efficient algorithms that combine traditional channel features with deep features for the detection and recognition of the traffic signs.

To reliably detect the traffic light, we use a dedicated camera with fine-tuned parameters, including the field-of-view, camera focus, exposure time, etc, as shown in the left of Fig.8. High-precision map could also aid the detection of traffic light by providing a rough location estimate of it. The location is projected to the current image, generating a Region-Of-Interest (ROI). We then only need to run the detection algorithm within the ROI.

The ground sign, the stop line and the zebra crossing are all detected in the inverse perspective image, as shown in the middle and right of Fig.8.

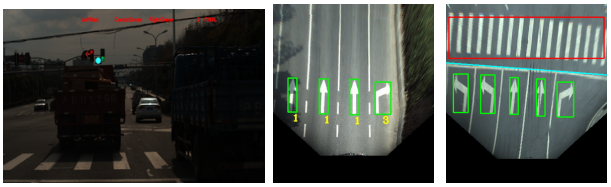


Fig. 8. The detetion of various traffic objects, including traffic light, ground sign, stop line, zebra crossing, etc.

C. Dynamic Object Detection and Tracking

The Detection And Tracking of Moving Objects (DATMO) is an essential capability for the autonomous vehicle. The autonomous vehicle always runs in a dynamic environment. Only if all other traffic participants, including pedestrians,

vehicles, etc, are reliably detected and tracked, the vehicle could navigate safely.

We have developed both lidar-based DATMO approaches and multi-sensor fusion based approaches. For lidar-based approaches, we firstly register neighboring lidar scans using any off-the-shelf scan matching algorithms. The lidar scans are also segmented into a number of potential objects. These objects are then linked based on the scan registration result. Each potential moving object is associated with a Kalman Filter to predict its position and velocity. To segment a lidar scan into a number of objects, it will inevitably produce over-segmentation or under-segmentation errors. In an extreme case [11], we try to bypass the segmentation module, and treat each grid cell as an 'object'. Particle filter is used to predict the velocity of each grid cell.

The HQ3 autonomous vehicle is also equipped with several millimeter-wave radars. By fusing the output of the radar with the lidar detection result, we can detect and track dynamic objects up to 150 meters away.

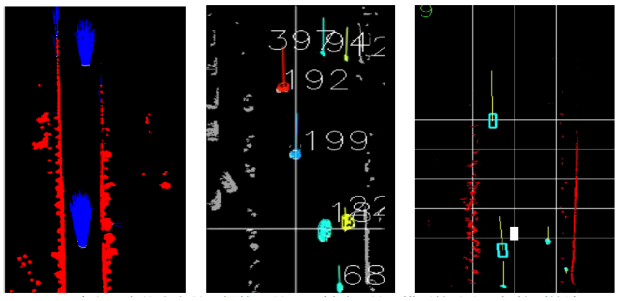


Fig. 9. The detection and tracking of moving objects. The left figure shows the estimated velocity of each grid cell [11]. The middle figure is the tracking result by choosing the segmented objects as the processing primitives. The right figure is the fused tracking result, which fuses the radar and lidar.

IV. BEHAVIOR PLANNING

The HQ3's behavior planner is responsible for the vehicle's behavior once the auto-switch of autonomous vehicle is on. The behavior planner makes decisions based on real-time driving contexts, which include a set of simplified traffic factors: global task, local environmental features and other traffic entities. Fig.10 shows a comprehensive planner architecture of HQ3.

A. Hierarchical Architecture

Our planner aims at being expandable, reusable and modular, so some tasks and behaviors can be added or removed when situation requires for new development. HQ3 could build additional modules quickly and these newly developed modules could be seamlessly embedded into the old system without the loss of original functions. To facilitate modularity and reuse, a hierarchical architecture is used to evaluate the feasible behavior in each scene.

1) Global Task: Fig.10 describes a hierarchical state stream of our behavior planner. At the highest level of the planner, different global driving tasks are designed according to the vehicle's state, which consist of highway task

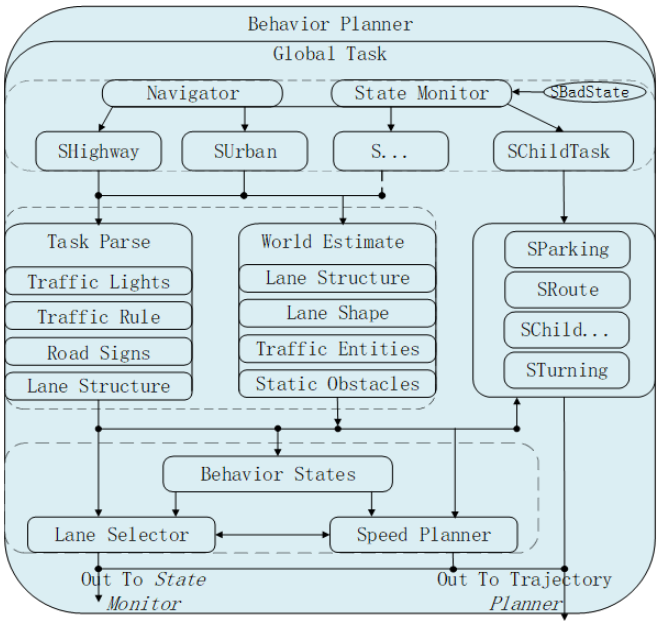


Fig. 10. The hierarchical behavioral reasoning chart. The top dashed box shows the phased global tasks. The middle dashed box shows two parallel running processes: the *Task Parse* and the *World Estimate*. The bottom dashed box shows the foremost behavior module.

(*SHighway*), urban task (*SUrban*) and some special sub-tasks (*SChildTask*), as shown in the top dashed box, where state names are prefixed with *S*. And the *S...* represents other analogous tasks which are too tedious to present here. The *SChildTask* represents some common but non-trivial tasks such as parking(*SParking*), turning(*STurning*), passing no traffic area (*SRoute*) and emergent stop(*SStop*), etc.

The *SChildTask* mainly deals with missions related to unstructured environment. Parking scenario is a representative example. The vehicle must park itself in the parking berth , and the berth may be in different orientations to the lane with or without obstacles around. When the vehicle comes to the parking task point, *State Monitor* module will trigger the *SParking* module. The parking mode, back-in or head-in , will be selected automatically according to the distribution of parking berths in a reasonable manner. When the vehicle is not close enough to the parking berth, *SParking* module will not output its planning result, thus avoiding unfeasible large turnning and slowing down for an accurate detection of lane markings. It is worth mentioning that *Surban* does run simultaneously unless *SParking* finishes the parking task.

The planner determines which high-level tasks to choose by using feedbacks *SBadState* from low-level behavior decision module and information from the navigator. A typical scenario of IVFC is that the vehicle has to seamlessly transition between the urban way state (*SUrban*) and a narrow country road state (*SRoute*) shown in Fig. 4. When the perception system identifies no correct lanes and only a narrow road available, the planner will automatically transit to *SRoute* for the country road driving. Another frequently occurring scene is the transition between the urban way state (*SUrban*) and the city highway state (*SHighway*). When the

vehicle drives in urban area and the navigator reports new road-level (e.g. highway, express road, country road, etc) information, then the planner's global task will generate lane and speed limitations and transit to *SHighway* smoothly.

2) Task Parse: For on-road driving environment, the middle dashed box provides two simultaneously running modules, *Task Parse* and *World Estimate*. In *Task Parse* module, traffic lights are explained as go, stop. While some signals are ignored considering lanes or intersections with specific navigation task. The lanes provides all alternative lanes according to the position and direction of the vehicle. The traffic rules and road signs indicate which lane is able to drive according to the vehicle's position. Other self-defined rules are also parsed here.

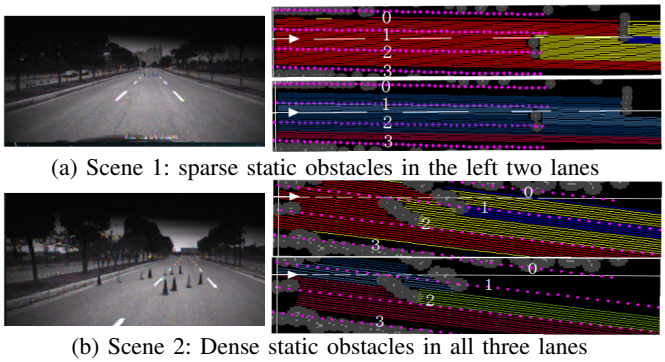
3) World Model: One of the *World Estimate* module's objects is to evaluate the relationship between traffic entities (e.g. vehicles, pedestrians and bicyclists) and the road model (represented by lane structure). All entities localize themselves to the road model. Some are associated with a lane *IDs*, while others are ignored in specific contexts (e.g. One entity moving relatively far away can be ignored for distance keeping scenario as the nearer entity limits the moving space of the vehicle tightly enough).

The other object is to generate the unblocked area of interest. Static obstacles are classified by a distance threshold D_{obs} which is an empirical value mainly decided by the sensors. The near static obstacles in the range of D_{obs} produce an static obstacle cost map. To get unoccupied planning space, we scan a lane using a group of shifted center lines in the cost map. For a lane with irregular obstacles, the shifted lines are split into segments as shown in Fig.11. A convenient way to evaluate the smoothness of a lane is to estimate the length of near line-segments, the number of line segments and the connection between line segments. We can generate several rectified corridors according to the connectivity between line segments and the lane they belong to. Empirically, three segments in each reference line are enough for most on-road scenario. The *line-segment* method can thoroughly search the whole road space. As a byproduct, the generated corridors clearly declare the topolgy for different maneuvers.

Different from near static obstacles, a far static obstacle, which is outside the range of D_{obs}, is labeled as an entity. The reason for distinguishing far static obstacles from near static obstacles is that far static obstacles are sparsely detected with a higher uncertainty. The far obstacle entities associated with lanes are then clustered into groups according to their distances. Besides, the obstacle entity cluster is re-examined every cycle. If no any single obstacle has been assigned to a cluster within a certain amount of time, the cluster expires and is ignored. For unstructured regions of the environment, the *World Estimate* module provides major information for *SChidTask*.

B. Behavioral Reasoning

Different from traditional behavioral reasoning systems by Boss and Junior[2], the top level of our behavior planner



(a) Scene 1: sparse static obstacles in the left two lanes

(b) Scene 2: Dense static obstacles in all three lanes

Fig. 11. Two examples of obstacle-free space search on a three-lane road. The left side is the front view of the vehicle. The top figures in both (a) and (b) show the shifted lines split into segments by static cost map. The bottom figures show the rectified segments. The white arrows represent the vehicle's driving direction. The red dots represent four road lines. The gray points in the right are static obstacles.

decompose the global tasks, such as parking, u-turn, etc, into several *SChildTask* for handling more complicated and elaborate tasks.

HQ3's *Behavior States* mainly considers the on-road lane change maneuver and precedence at intersections. For regular on-road driving, the *Behavior States* monitors surrounding entities at each cycle. As for intersection, the *Behavior States* chooses the target lane, and the decision on queuing or merging is implemented through speed control. The *Behavior States* selects an appropriate lane to keep track at a desired speed. The strategy for interacting with other dynamic entities is similar to the *path velocity decomposition* (PVD)[12] approach, while the difference is that we only select a lane instead of planning a path at this level.

1) Lane Selector: Selecting which lane to drive is a complex decision with multiple objectives. HQ3 aims at driving faster without loss of safety and comfort, while obeying the traffic rules. At road intersection area, the lanes could be generated by the *Task Parse* module. The difficulty exists in on-road situation where taking over or merging happens frequently.

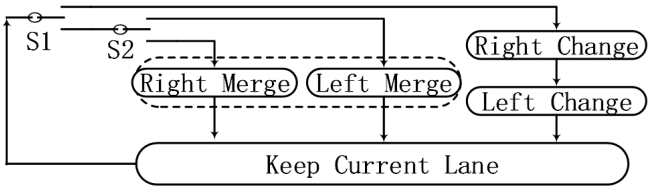


Fig. 12. The transitions between maneuver states. S1 and S2 are the bilateral enable switch.

The *Lane Selector* selects a lane from five basic on-raod maneuver states in *Behavior States*: *Left Merge* (LM), *Left Change* (LC), *Keep Current Lane* (KCL), *Right Merge* (RM) and *Right Change* (RC). The difference between lane change and merge is clear. LC tries to take over the current front vehicle (*Vcf*) to drive faster. RC is triggered for going back to the original lane after the taking over maneuver. LM and RM

aim at merging to the next lane by following task instructions such as transition between ramp and highway. Fig.12 shows the maneuver states transition priority. Supposing the white car in Fig.13 wants to go to the ramp, then the enable switch *S1* and *S2* switch to *Right Merge*. The *Behavior States* will cycle between *RM* and *KCL*.

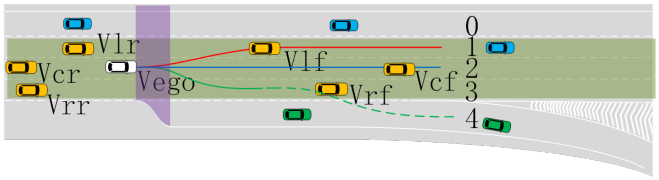


Fig. 13. The Relative Lane Selection chart near the ramp. The white car is an intelligent vehicle. The yellow cars on the pale green strap are policy-related cars to the white car. The others are policy-free ones. The right rear and left front vehicles are represennted as *Vrr* and *Vlf* for short. Vehicle's perception system reports four lanes on the left side of the purple strap and five lanes on the right side.

Fig.13 describes the *Relative Lane Selection* (RLS) for *Lane Selector* to reason the transition between states. HQ3 maintains the information of three abstract lanes, namely, *running-lane*, *selected-lane* and *target-lane*. In Fig.13, if the white car wants to reach *lane-0* and the red line is the immediate path. So the lane IDs for three lanes are *lane-0*, *lane-1* and *lane-2*. Except KCL, all maneuvers consider vehicles on the *runing-lane* and the lane HQ3 tries to reach. To evaluate whether the maneuver can be realized or not, HQ3 uses the time-to-collision and a minimum safety gap to restrict the maneuvers. For the sake of uncertainty, HQ3 also conservatively combines extra safety space and reaction time.

2) Speed Planner: Another key module for safety and smoothness of intelligent vehicle is *Speed Planner*, which must be synchronized with the *Lane Selector* to realize maneuvers. For different traffic scenarios, *Speed Planner* uses adjacent vehicles to determine the necessary safety gaps between vehicles and controls the vehicle's speed accordingly. Feedback control is used to reduce the difference between the desired and the actual inter-vehicle gap. The gap to the followed vehicle is related to the vehicle's speed.

Vehicles crossing intersections froms a kind of interrupt-and-go scenario, where HQ3 does not need to change lanes. An appropriate speed profile can solve this problem. At an intersection without traffic lights in Fig.14, all entities are split into five regions: a strap expanded by the green path and four quadrants split by the path in the forward direction. Two yellow vehicles may interrupt into the white car's path. Four speed profiles can be generated in temporal space in the right side of Fig.14. However, *profile1* and *profile4* are impossible for that the speed limit $tan\alpha$ can not be reached at the intersection. For safety and smoothness, *profile3* is the best choice.

Once the target speed and the *seleted-lane* are decided, information of lane boundary and vehicle states is delivered to a hierarchical real-time trajectory planning module [13][14], and the planned trajectory is implemented in the

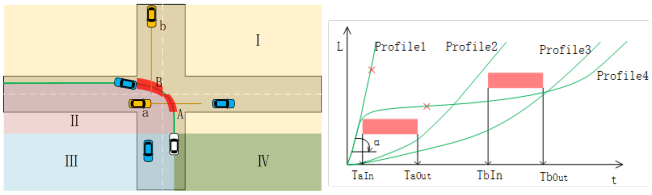


Fig. 14. Speed planner for interrupt-and-go. The yellow vehicle a and b interrupt into white vehicle's path at red cross region A and B in the left picture. Four green lines represents the length along path for different speed profile.

ribbon model based control module[15].

V. CONCLUSIONS

In this work, we introduce some key modules for HQ3, which enables it to perform well during the IVFC 2017. Our future work is to improve the reliability of each module, and thoroughly test HQ3 in more complicated scenarios.

REFERENCES

[1] S. Thrun et al., Stanley: The Robot That Won the DARPA Grand Challenge. Springer Berlin Heidelberg, 2007, pp. 1-43.

[2] M. Buehler, K. Iagnemma, and S. Singh, The DARPA Urban Challenge: Autonomous Vehicles in City Traffic. Springer Publishing Company, Incorporated, 2009, p. 6.

[3] O. S. Tas et al., "Making Bertha Cooperate-Team AnnieWAY's Entry to the 2016 Grand Cooperative Driving Challenge," IEEE Transactions on Intelligent Transportation Systems, pp. 1-15, 2017.

[4] H. Fu, R. Yu, L. Ye, T. Wu, and X. Xu, "An Efficient Scan-to-Map Matching Approach Based on Multi-channel Lidar," Journal of Intelligent & Robotic Systems, no. 3, pp. 1–13, 2017.

[5] M. Kaess, A. Ranganathan, and F. Dellaert, iSAM: Incremental Smoothing and Mapping. IEEE Press, 2008, pp. 1365-1378.

[6] T. Chen, B. Dai, Daxue, Liu and R, Wang, "Gaussian-Process-Based Real-Time Ground Segmentation for Autonomous Land Vehicles," Journal of Intelligent & Robotic Systems, vol. 76, no. 3-4, pp. 563–582, 2013.

[7] L. Xiao, B. Dai, D. Liu, T. Hu, and T. Wu, "CRF based road detection with multi-sensor fusion," in Intelligent Vehicles Symposium, 2015, pp. 192-198.

[8] D. Zhao, T. Wu, Y. Fang, R. Wang, J. Dai, and B. Dai, "Efficient Vehicle Localization Based on Road-Boundary Maps," in Pacific Rim International Conference on Artificial Intelligence, 2014, pp. 536-547

[9] T. Chen, B. Dai, D. Liu, J. Song, and Z. Liu, "Velodyne-based curb detection up to 50 meters away," in Intelligent Vehicles Symposium, 2015, pp. 241-248.

[10] Y. Fang, L. Sun, H. Fu, T. Wu, R. Wang, and B. Dai, "Learning deep compact channel features for object detection in traffic scenes," in IEEE International Conference on Image Processing, 2016, pp. 1052-1056.

[11] Q. Li, B. Dai, and H. Fu, "LIDAR-based dynamic environment modeling and tracking using particles based occupancy grid," in Proceedings of 2016 IEEE International Conference on Mechatronics and Automation, 2016, pp. 238–243.

[12] J. Johnson and K. Hauser, "Optimal acceleration-bounded trajectory planning in dynamic environments along a specified path," in IEEE International Conference on Robotics and Automation, 2012, pp. 2035-2041.

[13] Xiaohui Li, Dongpu Cao, Zhen He, and Qi Zhu, "Real-time trajectory planning for autonomous urban driving: framework, algorithms and verification," IEEE/ASME Transactions on Mechatronics, 2015.

[14] Xiaohui Li, Zhenping Sun, Dongu Cao, Daxue Liu and Hangen He, "Development of a new integrated local trajectory planning and tracking control framework for autonomous ground vehicles", Mechanical System & Signal Processing. 2017, pp 118-137.

[15] Z. Sun, Q. Chen, Y. Nie, and D. Liu, "Ribbon Model based path tracking method for autonomous land vehicle," in Ieee/rsj International Conference on Intelligent Robots and Systems, 2012, pp. 1220-1226.