Open Source Integrated Planner for Autonomous Navigation in Highly Dynamic Environments

Article *in* Journal of Robotics and Mechatronics · August 2017

DOI: 10.20965/jrm.2017.p0668

CITATIONS
18

READS
5,271

9 authors, including:

Hatem Darweesh
Nagoya University
9 PUBLICATIONS **31** CITATIONS
SEE PROFILE

Yoshiki Ninomiya
Nagoya University
75 PUBLICATIONS **1,556** CITATIONS
SEE PROFILE

Y. Morales
Standard Cognition
95 PUBLICATIONS **933** CITATIONS
SEE PROFILE

Naoki Akai
Nagoya University
62 PUBLICATIONS **283** CITATIONS
SEE PROFILE

Paper: Rb29-4-8537:

Open Source Integrated Planner for Autonomous Navigation in Highly Dynamic Environments

Hatem Darweesh, Eijiro Takeuchi, Kazuya Takeda, Yoshiki Ninomiya, Adi Sujiwo, Yoichi Morales, Naoki Akai, Tetsuo Tomizawa, Shinpei Kato

Nagoya University, Takeda Lab, Integrated Building (IB Kan) North, 8F, room 828, Furo-cho, Chikusa-ku, Nagoya, Aichi, Japan
E-mail: hatem.darweesh@g.sp.m.is.nagoya-u.ac.jp, takeuchi@coi.nagoya-u.ac.jp, kazuya.takeda@nagoya-u.jp, ninomiya@coi.nagoya-u.ac.jp , sujiwo@ertl.jp, morales_yoichi@coi.nagoya-u.ac.jp, akai@coi.nagoya-u.ac.jp, tomizawa@nda.ac.jp, shinpei@pf.is.s.u-tokyo.ac.jp

Planning is one of the cornerstones of autonomous robot navigation. In this paper we introduce an open source planner called "OpenPlanner" for mobile robot navigation, composed of a global path planner, a behavior state generator and a local planner. OpenPlanner requires a map and a goal position to compute a global path and execute it while avoiding obstacles. It can also trigger behaviors, such as stopping at traffic lights. The global planner generates smooth, global paths to be used as a reference, after considering traffic costs annotated in the map. The local planner generates smooth, obstacle-free local trajectories which are used by a trajectory tracker to achieve low level control. The behavior state generator handles situations such as path tracking, object following, obstacle avoidance, emergency stopping, stopping at stop signs and traffic light negotiation. OpenPlanner is evaluated in simulation and field experimentation using a non-holonomic Ackerman steering-based mobile robot. Results from simulation and field experimentation indicate that OpenPlanner can generate global and local paths dynamically, navigate smoothly through a highly dynamic environments and operate reliably in real time. OpenPlanner has been implemented in the Autoware open source autonomous driving frameworks Robot Operating System (ROS).

Keywords: autonomous driving, path planning, open source software

1. Introduction

Autonomous robot navigation requires perception, localization, control and planning. Although there are currently many open source resources available to researchers for perception, localization and control, it is hard to find an open source planner that is general enough to be used directly or which can be easily modified to suite a particular application, because planning is the core module that connects everything together and it is also application dependent.

In this study we concentrate on two types of planning, path planning and behavior planning, since both are needed in completely autonomous navigation systems for mobile robots. Museum tour guide robots are one example of autonomous navigation in indoor robots [**?**]. Studies on long-range outdoor robot navigation [**?**, **?**] have shown that state-of-the-art techniques can achieve good results. Moreover, the well-known DARPA Urban Challenge has shown that robotic navigation of car-like vehicles operating in real traffic is feasible [**?**, **?**]. Even though there have been successful implementations of autonomous vehicles, autonomous navigation is still a difficult problem given the vast number of possible conditions in dynamic environments. As a result, there are few open source planners that can deliver reliable results. Therefore, our goal in this study was to develop an open source autonomous navigation system that can handle dynamic environments while being extensible enough for the open source community to use, customize and enhance.

The architecture of OpenPlanner is illustrated in Figure 1. It includes a global planner that generates a global reference path from a vector (road network) map. The system then uses this global path to generate an obstacle-free local trajectory from a set of sampled roll-outs. At the center of the planner, the behavior generator uses predefined traffic rules and sensor data to function as an orchestrator, using collision and traffic rule cost calculations, selection of optimum trajectories, replanning commands and velocity profiling.

The research presented in this paper was originally developed for autonomous driving applications, and the system can also be found under feature/OpenPlanner of the Autoware website [**?**]. Autoware is an open source autonomous driving framework developed by Nagoya University which is used by many researchers for autonomous driving research and development [**?**]. Autoware is based on the Robot Operating System (ROS) described in [**?**]. It is a collection of ROS packages such as OpenPlanner, plus additional helper libraries. OpenPlanner is general enough to work with any mobile robot by simply adjusting its parameters, and has been

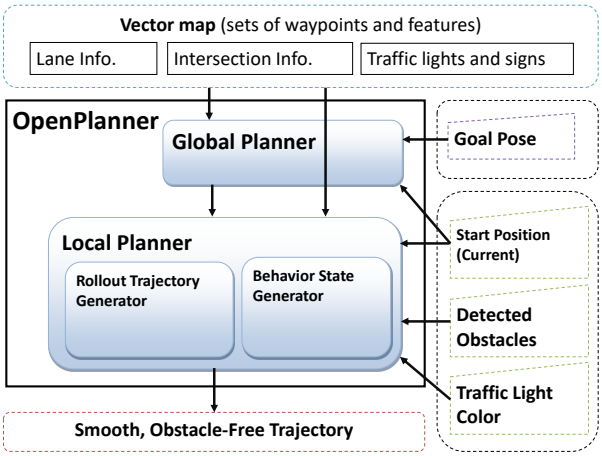


Fig. 1. : OpenPlanner architecture.

Fig. 2. : Tsukuba Real World Robot Challenge vector map.

used with both differential drive and non-holonomic robots. In this study we use an Ackerman-based steering robot, based on a mobility scooter, which was used in the Tsukuba Real World Robot Challenge (RWRC) [**?**] where we tested the planner.

RWRC is an annual mobile robot challenge held in the city of Tsukuba, Japan. The robots participate in the event must be able to achieve accurate localization and autonomous navigation in a dynamic environment, handle traffic lights and street crossing situations, navigate through an automatic sliding door, go inside a shopping mall and search for a designated person. Our goal in participating in the RWRC was to use OpenPlanner to achieve as many of these tasks as possible. Many innovative and effective planning algorithms are developed for this challenge every year, but unfortunately most of these planners are proprietary. Every year, new participants much develop their own planning systems from scratch, and only a limited number of the outlines and details of these systems are described, usually quite briefly, in published papers. Thus, one of the motivations for us to develop an open source planner was to provide the robotics community with a planner that is easy to understand and use which can also be continuously developed by its users.

In Section 2, we survey related state-of-the-art work. In Section 3, we provide an overview of the OpenPlanner system and its architecture. In Section 4, we give a detailed explanation of the systems global path planning method, including the use and structure of vector maps. The local planner is discussed in Section 5, and behavior state generation and the design of the state machine are explained in Section 6. In Section 7, our experimental set-up is described and our experimental results are evaluated quantitatively and qualitatively. We present our conclusions in Section 8. In Appendix A, we discuss current implementation and the wide range of platforms OpenPlanner can be used with. We also provide tips for users and developers for extension and parameter tuning. Algorithms are listed in Appendix B.

2. System overview

Figure 1 shows the general architecture for OpenPlanner, the main three components of which are a global planner, a behavior state generator and a local planner.

2.1. Overview of global planner

The global planner handles path routing. It takes the vector map, a start position and a goal position as input and then finds the shortest or lowest cost path using dynamic programming [**?**]. The global planner used by OpenPlanner can support complicated vector maps, but for this study the maps used were very simple. The entire map for the Tsukuba RWRC is shown in Figure 2. We annotated the map manually with traffic rules and features, such as traffic lights and stop lines, from start to goal. More details will be provided in Section 4.

2.2. Overview of behavior state generation

The behavior state generation module of OpenPlanner functions as the decision maker of the system. It is a finite state machine in which each state represents a traffic situation. Transitions between states are controlled by intermediate parameters calculated using current traffic information and pre-programmed traffic rules. Figure 3 shows the currently available states in the OpenPlanner system. More details are provided in Section 6.

2.3. Overview of trajectory generation

Inputs for the local path planner are the global reference path and the current position. Several candidate trajectories are then generated as roll-outs and the local planner selects the one with the lowest collective cost. Figure 4 shows seven possible rollout trajectories, including the center one. We used a modified version of the Stanford approach presented in [**?**]. The implementation algorithms will be explained in Section 5.

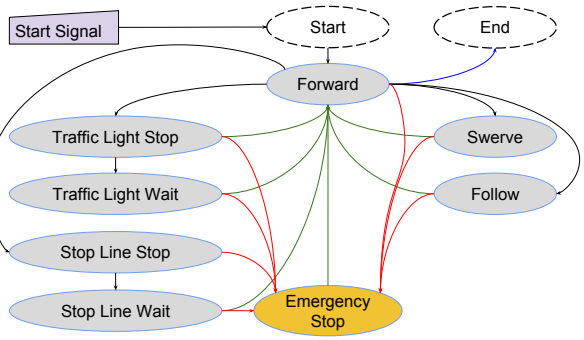


Fig. 3. : Current System Behavior States.

Fig. 5. : Experimental vector map with stop lines and traffic light.

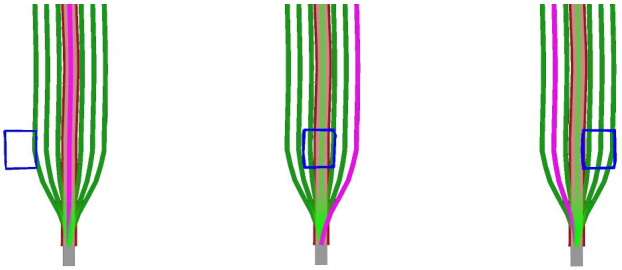


Fig. 4. : Local Planner in action, in a) central trajectory is free, in b) obstacle blocks central trajectory so the most feasible one was the most right trajectory, in c) the most feasible one was the second on the left.

3. Experimental setup and results

The first field test was conducted on the campus of Nagoya University, and a diagram of the route is shown in Figure 18. The second field test took place at the Tsukuba RWRC event, the vector map of which is shown in Figure 5. The objectives of these experiments were to test global planning performance from any current position to any goal position on the map and to evaluate performance of obstacle avoidance, stops at stop signs and stops at traffic lights. Additionally, we also evaluated smoothness, performance, accuracy, practicality and usability. In the simulation environment, we used real life vector maps composed of lines, intersections, stop lines and traffic signs as shown in Figures **??** and **??**. For field experiments, we added traffic information to the map manually.

The experiment platform is shown in Figure **??**. It is a modified mobility scooter so it could be controlled by computer. It includes one HDL32 Velodyne LIDAR sensor which is used for localization and object detection. In addition to the 3D LIDAR we use three 2D LIDAR for curbs and near obstacles detection. For the software part we had multiple ROS nodes for localization, obstacle detection, control, global planning, local planning and path following. In appendix A we provide technical information about OpenPlanner for ROS users.

In this section we will discuss the qualitative results first, then present key results from our simulation, experiments on the Nagoya University campus and participation in the Tsukuba RWRC event. Both the simulation and field tests had positive results.

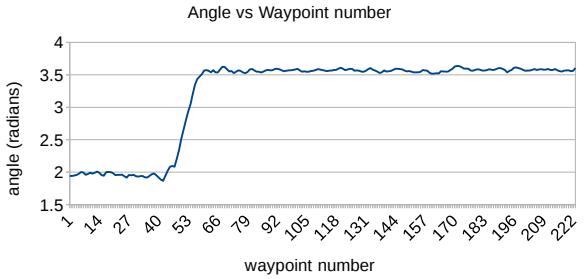
3.1. Qualitative results

The first qualitative aspect of our experiment is to evaluate stability, which means OpenPlanner should work all the time. Even if faulty data is provided, meaningful error messages should appear but the planner should never stop operating or crash. We tested this in a simulated environment by running the planner from any start point on the map, stopping localization at some point, and by jumping from point to point on the map manually and then switching to autonomous mode.

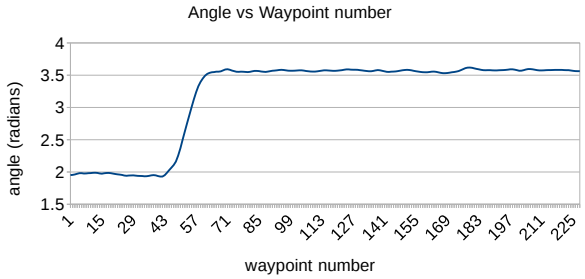
The second qualitative aspect is completeness, which means the system delivers smooth switching between different states and never remains stuck in one state. Experiments showed that the local planner stops successfully at stop signs and at traffic lights when they are red. Switching between follow, forward and obstacle avoidance states also worked properly, but smoothness of transitions depended on the reliability of the obstacle detection node. When encountering an obstacle-free path while navigating the map in Figure 5, the resulting sequence of driving behaviors were: Forward, Stop Sign, Wait Sign, Forward, Light Stop, Wait Light, Forward, Stop Sign, Wait Sign, Forward, Light Stop, Wait Light, Forward, Stop Sign, Wait Stop, Forward, Finish.

3.2. University campus experiment

Here we will concentrate on one section of the campus field test map, shown in Figure 5, which consists of two straight lines, one curve, one stop line and one traffic light. Figure 6 (a) shows the curvature of the generated path, and the smoothed output of that path is shown in



(a)



(b)

Fig. 6. : The angle of each way-point shows curvature of the generated path with and without smoothing.

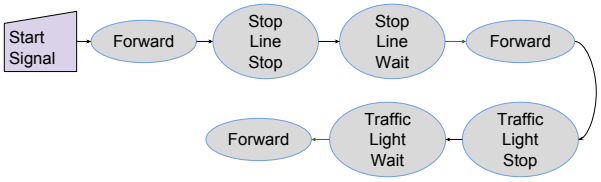


Fig. 7. : Simulation test results, navigating without obstacles.

Figure 6 (b). Smoothing improves path following performance without the need to relax the control parameters.

Results when navigating the map section shown in Figure 5 are displayed in Figure 8. , which shows the simulation results when navigating this section without any obstacles. Behaviors are represented by the blue line and behavior ID values. The orange line represents the trajectory index, which is the currently selected roll-out number. In this experiment we used 7 roll-outs, with 3 representing the number of the center trajectory. Figure 7 illustrates the behavior transition flow during this experiment.

Using the OpenPlanner in simulation mode enabled us to insert obstacles of random sizes. We repeated the previous experiment after adding two obstacles while the robot was moving, one obstacle before the stop sign and the other after the stop sign. Figure 9 shows that both state transition and trajectory switching results were smooth. Figure 10 is a diagram of the behavior

Fig. 8. : Simulation test results when navigating without obstacles. Behavior ID values: 2 = move forward; 5 = stop for traffic light; 6 = wait for traffic light; 7 = stop for stop sign; 8 = wait for stop sign; 11 = avoid obstacle.

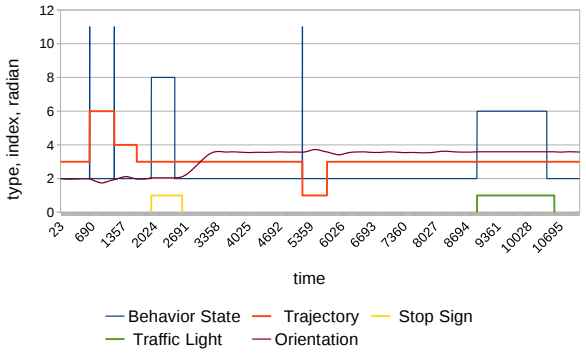


Fig. 9. : Simulation test results when navigating with obstacles. Behavior ID values: 2 = move forward; 5 = stop for traffic light; 6 = wait for traffic light; 7 = stop for stop sign; 8 = wait for stop sign; 11 = avoid obstacle.

state transition flow when navigating this section with obstacles.

Field experiment results for the same map section, shown in Figure 11, were similar and generally stable, with the vehicle stopping at the stop line and traffic light and avoiding obstacles when necessary. However, noisy input data resulted in an additional state transition.

3.3. Tsukuba RWRC experiment

One of the most difficult tasks during the Tsukuba event was the street crossing, which required stopping at a traffic light, crossing the street, making a very tight U-turn, stopping at a second traffic light and crossing the street again. By using vector maps we were able to dynamically choose to continue through this stage or bypass it. The default behavior of the planner is to find the shortest route from the current position to the goal, so normal route selection would result in the route shown in Figure 12 (a). If we wanted to attempt the task and cross the street, we

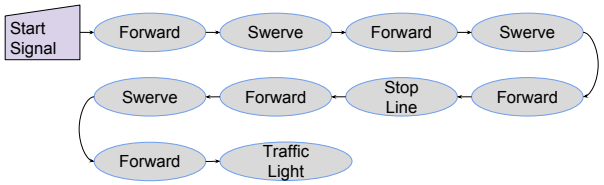


Fig. 10. : Behavior state flow while navigating the simulated vector map with obstacles.

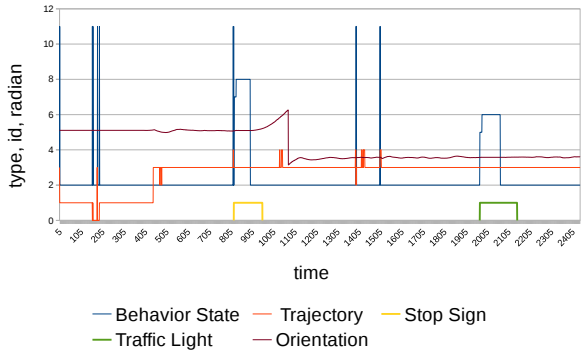
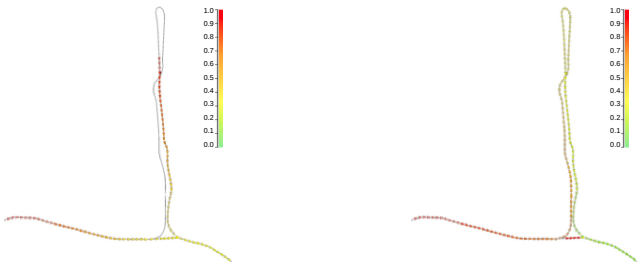


Fig. 11. : Field test results. Behavior ID values: 2 = move forward; 5 = stop for traffic light; 6 = wait for traffic light; 7 = stop for stop sign; 8 = wait for stop sign; 11 = avoid obstacle.

could simply increase the cost associated with the shortcut. Figure 12 (b) shows the planner choosing the longer route to avoid the high cost assigned to the shortcut. The resulting global paths are shown in Figures 13 (a) and (b) respectively. Behavior states when following both routes are simulated in Figure 14 (a) and (b).

3.4. Performance

In the campus field experiment, we were able to achieve real time performance of 14.6 iterations per second for local planning and behavior state generation, while detecting an average of 62 obstacles and an average total of 1,255 contour points (Figure 15). For each iteration we calculate the obstacles contours, track the obstacles using

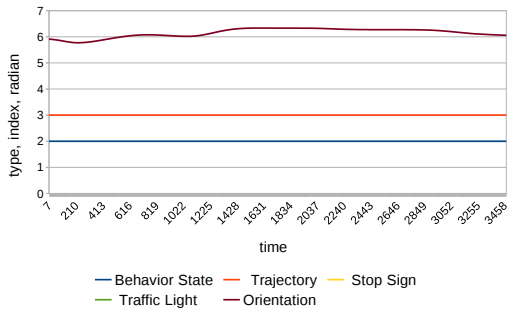


(a) (b)

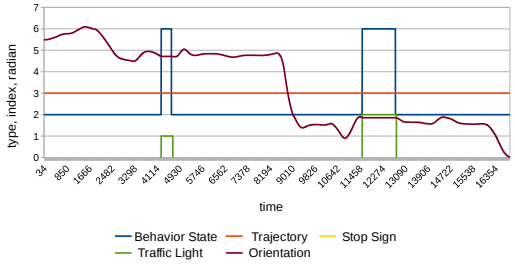
Fig. 12. : Global planning dynamic cost calculation, (a) without and (b) with a high shortcut penalty.

(a) (b)

Fig. 13. : Global planning final paths, (a) without and (b) with a high shortcut penalty.



(a)



(b)

Fig. 14. : Behavior state results when selecting the shortest route (top) or choosing to cross the street (bottom). Behavior ID values: 2 = move forward; 5 = stop for traffic light; 6 = wait for traffic light; 7 = stop for stop sign; 8 = wait for stop sign; 11 = avoid obstacle.

a Kalman filter, calculate the costs and then generated an new roll-outs if needed. Cost calculation is clearly a performance bottleneck. At this point, no performance enhancement techniques were used, not even a compiler optimization directive, but we were still able to achieve real time performance. The next objectives for OpenPlanner are smoother obstacle presentation and a larger number of generated roll-outs.

3.5. Accuracy

Our mobility scooter was able to avoid static and moving obstacles successfully, depending on localization accuracy. Our testing platform had an original localization

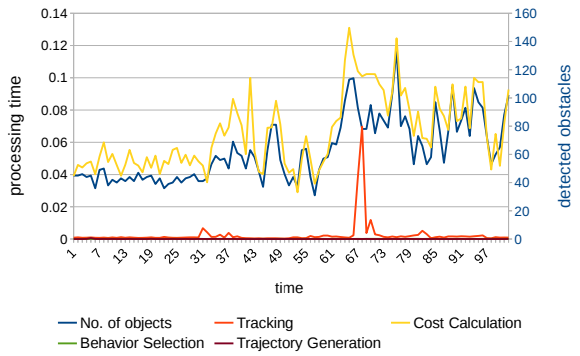


Fig. 15. : Results of campus field test. Y axis on the left shows processing time in seconds for tracking, cost calculation, behavior selection and path generation. Left Y axis shows the detected obstacles number.

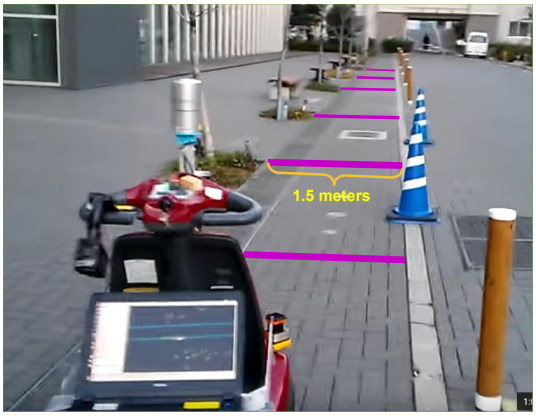
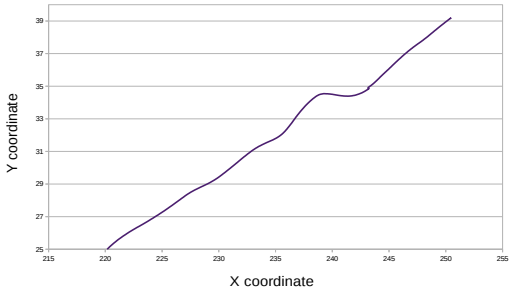


Fig. 16. : A narrow pathway was used in the campus testing area to evaluate performance when avoiding nearby obstacles.

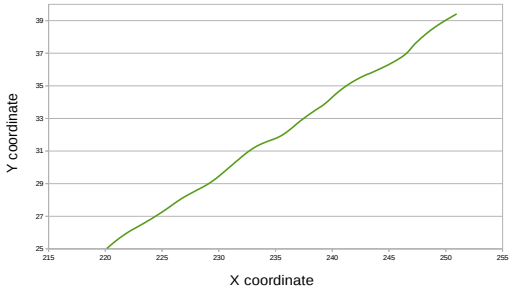
accuracy of within 10 cm, so we added 20 cm to its width and length as a safety margin for the robot As a result, in some cases the robot got as close as 5 cm to an obstacle while avoiding it. Recorded videos of experiments including visualization of environment is shown at OpenPlanner channel [**?**] on youtube. We tested several parameters using the same route section illustrated in Figure 16. Since we added a 20 cm safety margin to the width and length of the robot, it became difficult for the robot to drive in a straight line in the narrower sections. By changing the car tip and roll-in parameters, we were able to achieve a smoother driving path when traveling through this section. Figure 17 shows data from several trials while navigating through the section shown in Figure 16.

4. Conclusion

The main contribution of this paper is to provide OpenPlanner as an open source resource for the robotics community to use and enhance. The source code is available as a collection of ROS packages within the



(a)



(b)

Fig. 17. : Driving between cones and trees generated many changes of direction. In (a), using smallest roll-in margin, the motorized scooter changes direction more quickly but not smoothly. In (b) we used bigger car tip and roll-in margins, so the results became smoother.

Autoware project. It can be used as a stand-alone package or within the Autoware framework. It can use .kml format RNDF map files, which can be easily created and modified, and Autoware supported vector maps. The basic functionality of OpenPlanner is available as shared libraries, thus users can use it for development outside the ROS environment.

In this study we demonstrated that OpenPlanner can operate effectively and safely in dynamic environments by using a modified motorized scooter to successfully navigate through the Tsukuba Real World Robot Challenge and by navigating similar environments on the Nagoya University campus. Successful autonomous navigation requires avoidance of moving objects and pedestrians, the ability to follow moving objects along a path, navigating through narrow corridors, negotiating automatic doors, and stopping for traffic lights and stop signs. In our field tests, OpenPlanner demonstrated its ability to perform all of these tasks.

Acknowledgements

This research is supported by the Center of Innovation Program (Nagoya-COI; Mobility Society leading to an Active and Joyful Life for Elderly) from Japan Science and Technology Agency.