Paper:

End-to-End Autonomous Mobile Robot Navigation with Model-Based System Support

Alexander Carballo*, Shunya Seiya, Jacob Lambert**, Hatem Darweesh**, Patiphon Narksri*, Luis Yoichi Morales*, Naoki Akai*, Eijiro Takeuchi**, and Kazuya Takeda****

*Institutes of Innovation for Future Society, Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan
E-mail: alexander@g.sp.m.is.nagoya-u.ac.jp
**Graduate School of Informatics, Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan
[Received February 23, 2018; accepted July 7, 2018]

Autonomous mobile robot navigation in real unmodified outdoor areas frequented by people on their business, children playing, fast running bicycles, and even robots, remains a difficult challenge. For eleven years, the Tsukuba Challenge Real World Robot Challenge (RWRC) has brought together robots, researchers, companies, government, and ordinary citizens, under the same outdoor space to push forward the limits of autonomous mobile robots. For the Tsukuba Challenge 2017 participation, our team proposed to study the problem of sensors-to-actuators navigation (also called End-to-End), this is, having the robot to navigate towards the destination on a complex path, not only moving straight but also turning at intersections. End-to-End (E2E) navigation was implemented using a convolutional neural network (CNN): the robot learns how to go straight, turn left, and turn right, using camera images and trajectory data. E2E network training and evaluation was performed at Nagoya University, on similar outdoor conditions to that of Tsukuba Challenge 2017 (TC2017). Even thought E2E was trained on a different environment and conditions, the robot successfully followed the designated trajectory in the TC2017 course. Learning how to follow the road no matter the environment is of the key attributes of E2E based navigation. Our E2E does not perform obstacle avoidance and can be affected by illumination and seasonal changes. Therefore, to improve safety and add fault tolerance measures, we developed an E2E navigation approach with model-based system as backup. The model-based system is based on our open source autonomous vehicle software adapted to use on a mobile robot. In this work we describe our approach, implementation, experiences and main contributions.

Keywords: Tsukuba Challenge, end-to-end navigation, deep learning, path planning, obstacle avoidance

1. Introduction

Fully autonomous mobile robot navigation performs a series of tasks on the same environments used by people every day, having to deal with pedestrians, bicycles, children, and possibly other robots, no matter the weather, both indoors and outdoors, negotiating with cars and semaphores at crosswalks; that was the vision conceived more than eleven years ago by the pioneers [1–3] of the Tsukuba Challenge Real World Robot Challenge (RWRC) [a], year to year confronted by dozens of teams.

One of the basic rules of the Tsukuba Challenge (TC) is that the environment will not be modified in any way to suit the robot or teams: it is the same outdoor environment used by people everyday for their business. Therefore, robots should take and adapt the environment as it is, no matter weather changes. Furthermore, it is a fundamental responsibility for all teams that robots should not harm or endanger in any way humans transiting the designated environment; several safety measures have been taken and improved during the years.

From our past experiences [4–9], we know TC also involves moving from outdoors to indoors and back, identifying individuals among the crowd, constant blocking and occlusions by curious children and adults alike, dealing with fast moving bicycles and robots out of control, abrupt steps and curbs, very limited time between semaphore changes to cross a road transited by vehicles, season changes affecting appearance of the environment, and so forth.

For many years, the dominant implementation strategy of most teams is called model-based: detection, actuation, localization and navigation algorithms for a robot are implemented following some defined set of rules, conditions, thresholds, and so on (see [4, 6, 10–12] for some model-based implementations on previous TC). In recent years, teams have started considering machine learning, in particular deep neural networks (DNN), to achieve higher performance for object detection and classification (see [13–15]). As far as we know, no other team has considered the so called End-to-End navigation (sensors-to-

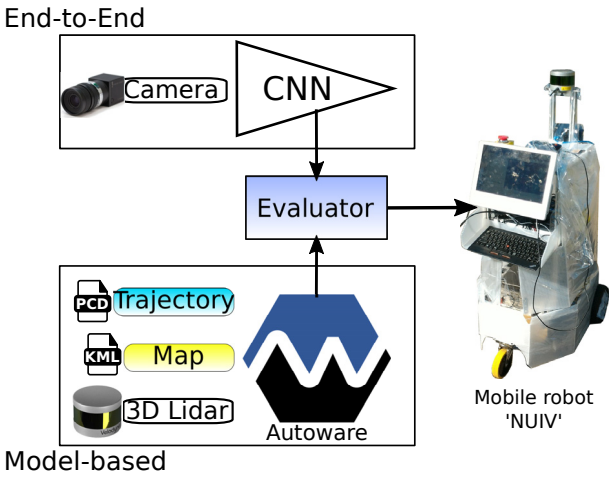


Fig. 1. Our system for mobile robot navigation.

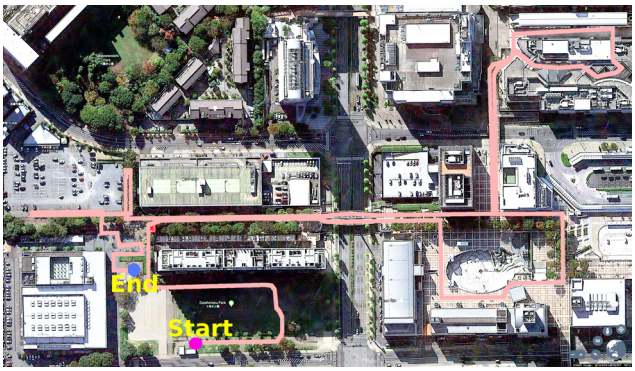


Fig. 2. The complete Tsukuba Challenge 2017 course (satellite view from Google Earth).

actuators) approach.

In this work, we propose a system for autonomous mobile robot navigation with two components: End-to-End (E2E) navigation and model-based navigation as supporting system. In E2E, camera images are fed to a convolutional neural network (CNN), it generates as output the direction to the next target waypoint used to navigate towards the defined goal. While E2E navigation has been studied previously for mobile robots [16] and in recent years has been considered for autonomous vehicles [17–19], our approach is more simple to realize particularly for a mobile robot.

To increase safety and robustness of our system, E2E is complemented with a model-based component: when the robot is deviated from its trajectory or there is some obstacle, an evaluator program chooses the model-based system to steer the robot and then switches back to E2E. For the model-based system we adapted our open source platform for autonomous vehicles driving, called *Autoware* [20], to suit our mobile robot. **Fig. 1** shows a concept representation of our system.

Figure 2 shows a satellite view of the Tsukuba Challenge 2017 course. It consists of a 2 km path starting at a park (lower left), then moves to a promenade, then a plaza, then goes indoors and back to outdoors. For our participation in Tsukuba Challenge 2017 (TC2017) using the proposed E2E approach, we chose a small segment of about 200 m from the complete 2 km course. Due to distance limitations, we conducted several experiments and trained the E2E navigation neural network on a different course at Nagoya University, of similar attributes our TC2017 trajectory: a park covered with trees, diverse road surfaces, walkways with many branches (intersections), and season changes affecting the visual appearance. A comparison of both training and testing environments is presented on **Fig. 3**.

The main contributions of this work are summarized as follows:

- E2E method for mobile robot navigation based on deep learning,
- effective transition between E2E and model-based methods for safety and robustness,
- adapting a autonomous vehicle software platform to a mobile robot, and
- lessons learned on the implementation of this navigation system.

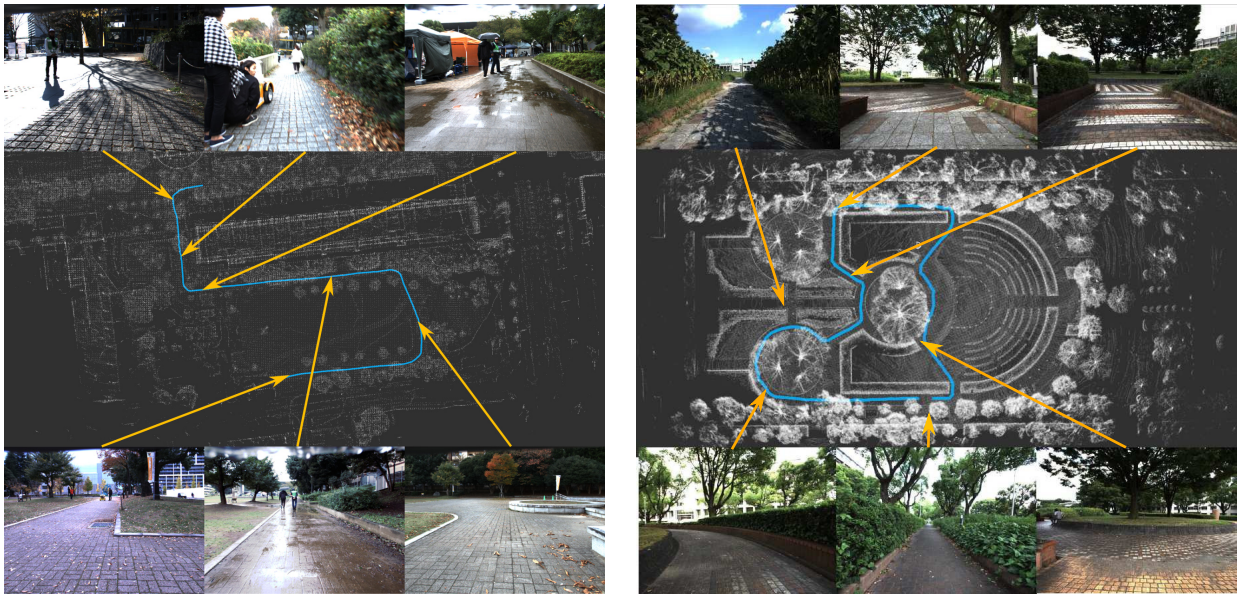
This work is structured as follows: in Section 2 we present a survey of related research followed by an overview of our system in Section 3. In Section 4 we discuss in detail our E2E approach while in Sections 5 and 6 we explain our model-based and backup systems, respectively. In Section 7 we present experimental results of our evaluations at Nagoya University and at Tsukuba Challenge, and conclude in Section 8.

2. Related Works

Tsukuba Challenge is an autonomous mobile robot event held at Tsukuba City in Japan for more than eleven years. Every year, dozens of teams confront the challenge (for TC2017, as many as 53 teams enrolled) and their collective experiences have been published in several media.

Our team, *Meidai Autonomous Driving (MAD)*, was formed by several members with previous experiences at former editions of TC. Our first participation was at the very first TC held in 2007 (see Morales et al. [4]) for successful 1 km autonomous navigation and in 2008 we had another participation (see Morales et al. [5]). Details of participation by the Nagoya University itself are detailed on [7–9].

Sensors-to-actuators (E2E) based navigation has been studied previously, where the dominant sensor is so far vision. Pomerleau et al. [21–24] are among the pioneer works on E2E vehicle navigation; their ALVINN (autonomous land vehicle on a neural network) takes images from a camera and computes several images transformations (what they call virtual camera) by shifting and rotating the original image to left and right 14 times to consider situations where the vehicle is deviated from its trajectory.



(a) (b)

Fig. 3. Evaluation environment of E2E, (a) at Tsukuba Challenge 2017, (b) at Nagoya University.

These images then are input to a basic artificial backpropagation neural network with 30 outputs to indicate steering direction.

The DARPA Learning Applied to Ground Robots (LAGR) program brought back learning methods for E2E. LeCun et al. [16] proposed an E2E approach for a mobile robot using stereo images and a convolutional neural network (CNN) with two outputs to code left turns and right turns. In their approach, a human operator steers the robot several times along a designated path, which includes obstacles, the robot learns from images how to follow the trajectory and avoid obstacles, with a steering performance very similar to the human operator. Their system was trained under several illumination and weather conditions. Bajracharya et al. [25] also proposed an E2E system for mobile robot off-road navigation using stereo camera images and two stages of supervised/self-supervised learning based on support vector machines (SVM), short-range to understand geometric features of the road, and long-range for traversable road and path following.

In recent years, E2E navigation has been considered again on autonomous driving vehicles. Chen et al. [26] proposed a model-based and deep learning-based method for autonomous vehicle driving on a simulator. Their target is to measure driving affordance using a CNN to compute over 13 indicators from road images, including the angle of the car to the road, distance to lane markings, and distance other cars. These indicators are fed to a model-based driving controller.

Bojarski et al. [17], similar to LeCun's work, used CNN for E2E navigation with emphasis on lane keeping, over diverse roads achieving over 10 miles. For training data, they collected actual steering data and images from three cameras installed in the front of the vehicle (left, center and right), under several weather and road conditions. Three cameras allowed to capture data from different perspectives regarding the traffic lane center. They also augment the data set using virtual images generated by changing each camera's viewpoint using a random rotation angle and shift similar to Pomerleau. Steering information is included during training; for the generated images, virtual steering information is generated so the vehicle returns to the center of the lane after two seconds. Including this augmentation their training data amounts to over 72 hrs. Their viewpoint transformation approach allows them to train the model including diverse lane center deviation conditions and then being able to return back to the center.

Xu et al. [18] proposes an E2E method to learn driving models from monocular video data where time-sequence is important. For that, they developed a dilated fully convolutional network (FCN) combined with long short-term memory (LSTM). Their system allows to predict driving actions (straight, stop, left, right). Yang et al. [19] used several CNNs for feature classification aimed to analyze E2E vehicle control on a simulator. Their networks extract features like sky area, road-side area and road surface area. Each CNN output is the steering angle and samples come from a human operating the driving simulator. Their work does not seem to consider image transformations.

Our work has been inspired by the works of Bojarski et al. [17] and Pomerleau [22]. We use a similar CNN architecture as [17], however, our work uses one single camera on a mobile robot. Bojarski's work is aimed at lane keeping while our approach is aimed at robot steering in complex trajectories with branching. In Seiya et al. [27], we presented a first evaluation this E2E model for a mobile robot. We use virtual camera images by viewpoint transformation approach similar to [22].

Different from previous works, a novel element of our

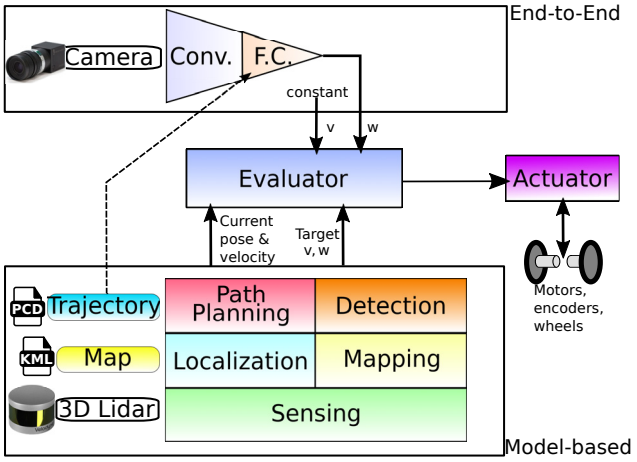


Fig. 4. Implemented navigation system.

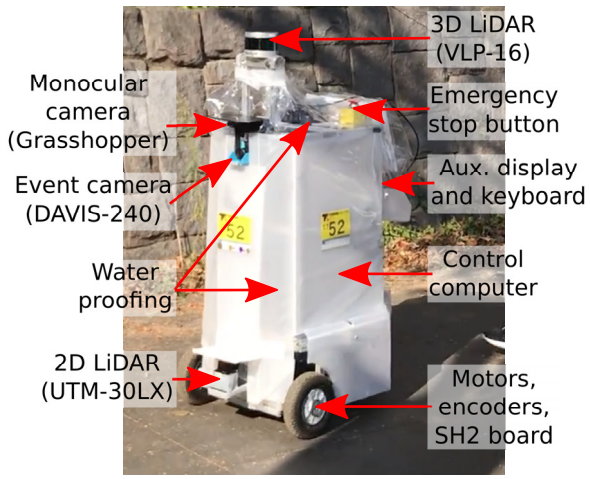


Fig. 5. The NUIV robot.

approach is that steering information is included in the CNN during training: the direction to the next waypoint on the trajectory is encoded and added to the fully connected (FC) layers.

Regarding our model-based system, Autoware (see Kato et al. [20] and [b]) is a collection of modules, libraries and nodes for autonomous vehicles. It covers most of the required functions, including: driving, mapping, localization, trajectory generation, path planning, object detection (pedestrians, other cars, traffic signs, etc.), drivers for multiple sensors (2D/3D LiDARs, cameras, radars, GPS/GNSS, IMU, car computers and in-car sensors), different map formats, an so on. This open source tool started at Nagoya University, tested in vehicles running autonomously in many cities, is currently maintained by TierIV Inc. [c], with a large community of users and developers worldwide. One key element of Autoware is its implementation based on ROS (robot operating system), therefore it becomes accessible to others using current vehicles, robots, sensors and rosbag log files.

3. System Overview

The system implemented for our participation in TC2017 is shown in **Fig. 4**. Two major sub-systems work simultaneous and complementarily for maximum safety and robustness: our End-to-End system based on CNN and our model-based system using Autoware.

The E2E system uses camera images and the direction to the next target waypoint as input data; the CNN was trained and tested at Nagoya University and later tuned for the Tsukuba Challenge course. The model-based system uses the 3D LiDAR, a priori maps generated off-line from 3D LiDAR data during trial runs, and a global trajectory as inputs; several modules like mapping, localization, path planning and detection cooperate so the robot navigates towards the goal.

An evaluator program gathers output data from both sub-systems, robot current pose and velocity, target linear velocity υ and angular velocity ω. Under normal circumstances the system runs on E2E mode so the evaluator uses the target $\langle \upsilon, \omega \rangle$ pair and sends it to the robot control board. When the evaluator judges the robot has deviated over some margin from its trajectory, it switches to the model-based target $\langle \upsilon, \omega \rangle$ pair.

The robot of our team, called NUIV (**Fig. 5**), has multiple sensors: a Velodyne VLP16, a Hokuyo UTM-30LX, a Point Grey Grasshopper3 camera, a iniLabs' Dynamic and Active-pixel Vision Sensor (DAVIS-240) event camera (only to collect data) and rotation encoders; positioning sensors like GNSS, IMU, etc. were not used. A SH2 board was used to as actuator to control the robot motors and read odometry information (not used in our participation) running SH-Spur software [d].

The robot's main computer is a HP Zbook Studio 3, with Intel Xeon CPU E3-1545M with 4 cores/8 threads, NVIDIA Quadro M1000M GPU with 4 GB GPU memory and 512 GPU cores, 32 GB main RAM and a external 2 TB SSD storage for log data recording. It is important to mention that SSD had to be formatted with EXT4FS file system different from the default NTFS file system to be able to match the log writing speeds demanded by our system (2 cameras, 3D LiDAR, 2D LiDAR, etc. with high data rates). Another important lesson in this regard is that heat coming from notebook and other devices affects writing speed of SSD so it had to be well cooled and conditioned. Autoware version 1.5.1 was modified to suit our needs as described above. For robot control we used SSM/SH-Spur libraries [d]. adapted to use with ROS.

4. End-to-End Navigation

4.1. CNN

The model of our convolutional neural network (CNN) is shown in **Fig. 6** and was inspired on [17]. The network has 11 layers: one normalization layer (Norm.), 5 convolutional layers (Conv.), and 5 fully connected layers (FC). The input image RGB planes are passed to the network.

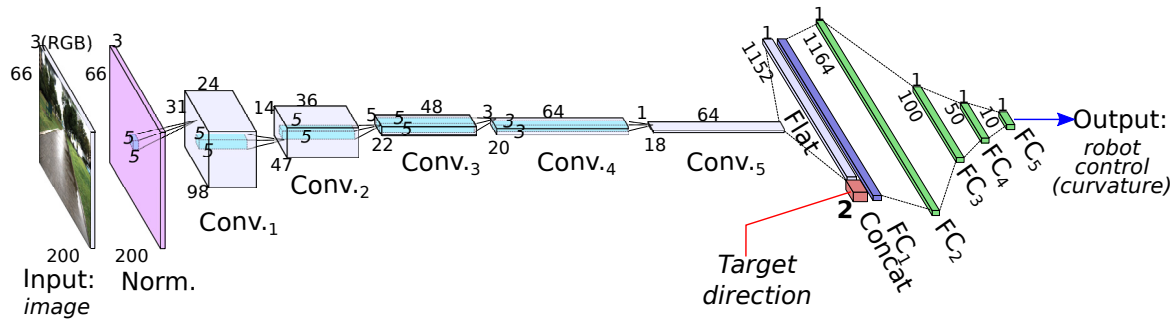


Fig. 6. Convolutional neural network model of our E2E approach, kernels sizes are also illustrated.

As shown in **Fig. 6**, the first 3 convolutional layers have a 5×5 kernel (2×2 stride) and the remaining 2 layers a 3×3 kernel (1×1 stride).

The convolutional layers correspond to feature maps: from low level features in the initial convolution layers towards higher level features with relation to robot control (steering) in the final layers. The output of CNN is the angular velocity ω to steer the robot towards the next target. While it is not generated by the neural network, the E2E system also outputs the linear velocity υ to drive the robot (shown in **Fig. 4**), set here as a constant value of 20 m/s.

One questionable element of this network is that this target direction vector (2 neurons) is very small when compared to the flattened feature map, thus its contribution will be insignificant. Therefore, we added a $\mathbf{L}^2$-norm of the image feature map weight coefficients on the error function, achieving good balance on both sets of feature vectors for training. Our loss equation corresponds to Eq. (1).

$$loss = \frac{1}{N}\sum(f_i - \mathscr{Y}_i)^2 + \frac{\eta}{2}\sum W_1^2 \quad . \quad . \quad . \quad . \quad . \quad (1)$$

where N is the number of training data, f_i is the predicted value at the output layer and $\mathscr{Y}_i$ is the actual desired response value (curvature κ to the next target). W_1 is the set of weights from the first fully connected layer (FC_1), and η is a mixture coefficient defined experimentally ($\eta = 0.05$).

With this method our robot learns the necessary control signals to move towards the next target. In fact, the network learns how to steer the robot (local navigation) to follow the specified trajectory, and is able to deal with unknown trajectories as well.

4.2. Data Augmentation

As the robot follows the trajectory using camera images, it is likely that it deviates: vibrations from rough terrain, wheel slippage, localization errors, illumination changes, and inaccurate model training are some of the causes. Deviations from the path means that the camera viewpoint changes so the visual appearance of the next target waypoint also changes.

To cope with these deviations, we augment the set of images used for training so that the robot knows how to recover. We used Zhang [28] calibration method to extract the camera's intrinsic and extrinsic matrix necessary for the viewpoint transformation, and based on Hartley et al. [29] we computed 9 different images by rotating the viewpoint by some angle α in intervals of 5° in the range of $[-20°, 20°]$. This process is shown in **Fig. 7**, the central original image corresponds to 0°.

Regarding validation of such configuration, in Seiya et al. [27] we evaluated different cameras layouts on a mobile robot and also virtual images by changing each camera viewpoint. Training data was collected for the following configurations (all cameras with identical specifications and synchronized):

- single camera (at front-center) and no transformation,
- single camera (idem) with viewpoint transformations,
- three cameras (at front-left, front-center, front-right) and no transformation,
- three cameras (idem) with viewpoint transformations

From experimental results we found that one single camera with data augmentation by viewpoint transformation minimized the lateral error on the defined lane boundaries (see [27] for details).

While three cameras with viewpoint transformations cover a larger field-of-view (FOV) and seemed more adequate for the problem, we found that lateral cameras data dominated over the central camera during training; Bojarski suggests randomization to control the amount of lateral cameras data. Nevertheless, in this work we chose to use a single camera with viewpoint transformation, this simplifies the system and minimizes processing time and resources.

For each training image, we used PurePursuit trajectory following algorithm [30, 31] to generate the necessary paths to return from the transformed camera viewpoint to the robot trajectory angle θ (in the robot's reference system) after N camera frames (here, $N = 20$ frames, about 2 seconds). We recorded previously the array of robot poses (position and heading) over time, with camera images synchronized with robot pose using timestamps. Therefore, the robot position N frames ahead is simply $i + N$ where i is the current posture index.

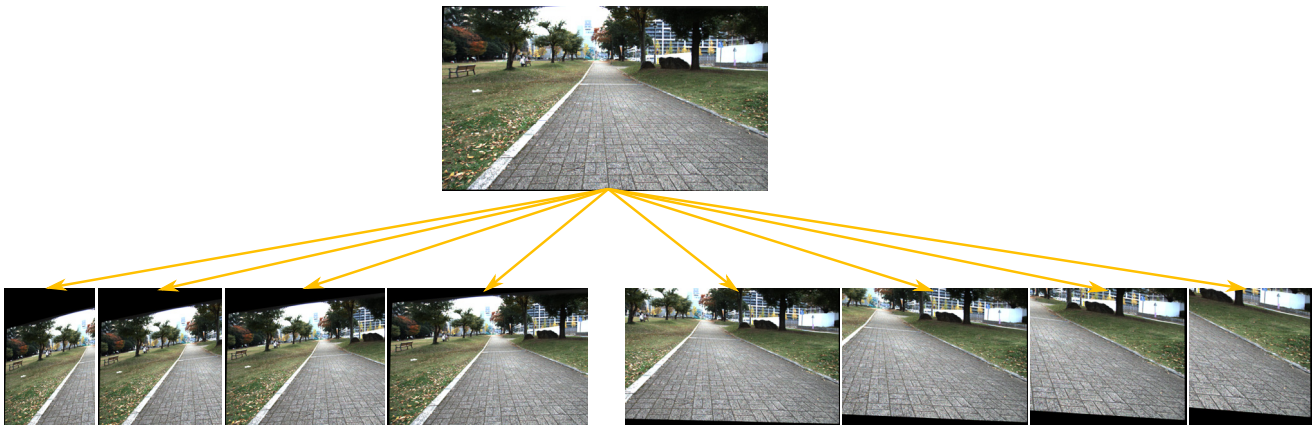


Fig. 7. Data augmentation by viewpoint transformation.

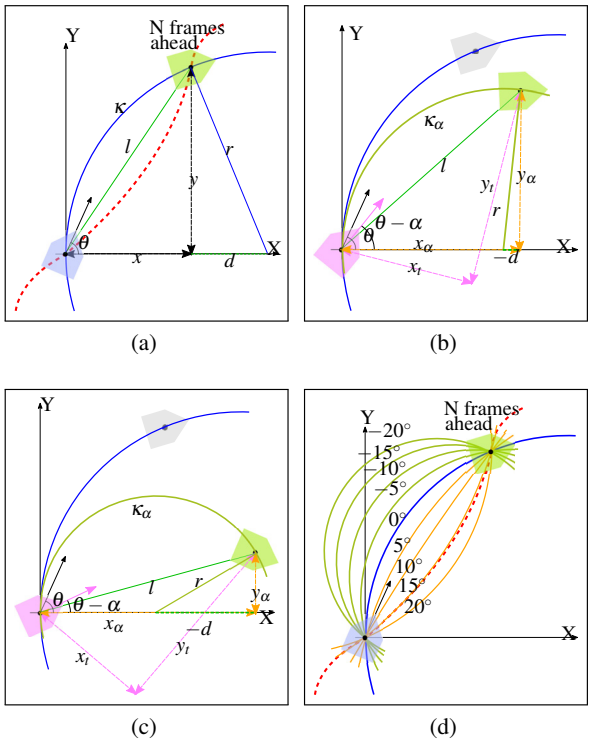


Fig. 8. Generation of trajectories using PurePursuit for data augmentation.

Based on Coulter [31], we computed the corresponding path curvature κ_α for each angle α for a lookahead distance N projected on the ground 2D plane. For the sake of completeness, details are as follows. From **Fig. 8(a)**, $l^2 = x_\alpha^2 + y_\alpha^2$, $r^2 = y_\alpha^2 + d^2$, and $d = r - x_\alpha$, therefore:

$$\begin{cases} r = \dfrac{l^2}{2x_\alpha}, \text{ and with } \kappa = \dfrac{1}{r} \\ \kappa_\alpha = \dfrac{2x_\alpha}{l^2} = \dfrac{2x_\alpha}{x_\alpha^2 + y_\alpha^2} \end{cases} \quad \ldots\ldots (2)$$

where l is the chord length of the arc (lookahead distance to the target position) with coordinates $\langle x_\alpha, y_\alpha \rangle$, r is the circle radius and d is an offset. The curvature κ_α corresponds to the label of each image during training.

Adding this data during training allowed us to learn the necessary control signals to return back to the center of the trajectory when the robot is deviated. In **Fig. 8** we show some examples of curvature generation, **Fig. 8(a)** shows the case for $\alpha = 0°$ (no deviation), **Figs. 8(b)–(d)** some subsequent deviation cases. Rotating all the generated trajectories back to the initial undeviating robot heading θ is shown in **Fig. 8(d)**.

4.3. Target Direction

Right before the FC_1 layer in **Fig. 6**, the direction of the next target waypoint is concatenated to the flattened $Conv_5$ layer. Target direction data is read from the robot trajectory obtained from model-based as shown in **Fig. 4**: from the pointcloud of the 3D LiDAR sensor we perform self localization and to find the nearest waypoint we used `waypoint_follower` from Autoware. The target direction does not come from the self localization system world coordinates but the target direction as seen from the nearest waypoint in robot local system.

Target direction T is computed from the robot's twist data (linear and angular information) and normalized T_x and T_y components (i.e., $T = \langle \frac{T_x}{\|T\|}, \frac{T_y}{\|T\|} \rangle$). Computation of T components is straightforward (refer **Fig. 9** for details):

$$\begin{cases} \omega = \dfrac{\upsilon}{r} \text{ with } \kappa = \dfrac{1}{r}, \\ \kappa = \dfrac{\omega}{\upsilon} = \dfrac{2T_x}{T_x^2 + T_y^2}, \text{ from Eq. (2)} \end{cases} \quad \ldots (3)$$

$$0 = T_x^2 - \frac{2T_x}{\kappa} + T_y^2, \quad \ldots\ldots\ldots (4)$$

$$\text{with } T_y = \upsilon \quad \ldots\ldots\ldots\ldots (5)$$

$$T_x = \min\left(\frac{1}{\kappa} \pm \sqrt{\frac{1}{\kappa^2} - \upsilon^2} \right) \quad \ldots\ldots (6)$$

As shown in **Fig. 9**, for the next target direction input, we used the curvature necessary to go from the current heading (no transformation, $\alpha = 0°$) to an artificial way-

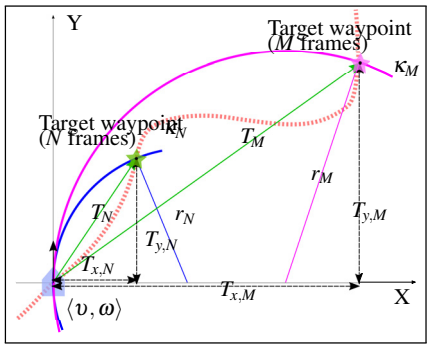


Fig. 9. Target direction κ_N and κ_M for N and M frames lookahead.

Table 1. CNN general training configuration.

Condition	Value
CNN input	RGB images from camera and direction vector
Image size	Horizontal 200 pixels Vertical 66 pixels
CNN output	Next target curvature κ
Activation function	Relu
Training times	200000 iterations
Mini-batch size	10
Learning rate	0.00001 Adam optimization

point set at M frames ahead (here, $M = 160$ or 16 seconds). As such, we can steer the robot using a smaller lookahead distance N while "keeping in sight" the next coming waypoint at a longer lookahead distance M. This longer value suits best for steady steering for non deviated cases and straight road segments.

4.4. Training

Network training was done at Nagoya University and the training configuration is shown in **Table 1**. As already explained, each image for training and validation is transformed by 9 different viewpoint angles $\alpha_{i=1,\ldots,9}$ and the corresponding robot direction label (desired output $\mathscr{Y}_i$) corresponds to the path curvature κ_α. The actual number of images used for training and validation will be explained in Section 7.

5. Model-Based Navigation: Autoware

For our participation in Tsukuba Challenge 2017, Autoware was the tool used for model-based navigation and it was customized to suit our mobile robot. Some parameters and programs originally designed for a large vehicle (a Toyota Prius PHV) were modified, including: vehicle dimensions, maximum speed and acceleration, lateral safety (minimum distance with obstacles at left and right), longitudinal safety (minimum distance with closest obstacles in the travel direction), number and separation of rollouts for local planning, certain planning policies, etc.

From Autoware, we used the modules discussed in the following sections.

5.1. Mapping

`ndt_mapping_tku` is an implementation of 3D NDT (Normal Distributions Transform) based on Takeuchi et al. [32] and Magnusson [33]. `ndt_mapping_tku` was fast and simple enough for the scale of this job. The map itself is quite correct from top-down view (**Fig. 10(a)**), but from the side view (**Fig. 10(b)**) the map has a persistent deformation on the vertical axis. This issue was confirmed before using different sensors and also by different teams. While the actual cause has not being identified yet, it is argued that this problem is due to the rather coarse resolution of the 3D LiDAR. Nevertheless, the same map was used during trials and the finals.

The map was generated off-line from raw scan data from the 3D sensor alone, using a downsampling resolution of 1.0 m over a total volume of 2600 m $\times$ 2600 m $\times$ 300 m. The output is a PCD (point cloud data) file which is used as the map reference in Autoware.

5.2. Localization

`ndt_matching` is the NDT implementation only for matching of scans against a 3D reference map (see also Akai et al. [34]). Depending on the map resolution and environment conditions (dynamic obstacles around the robot) initial matching convergence might take a potential long time. Our team relied completely on 3D LiDAR for robot position estimation and we did not use any other position reference sensors like GPS/GNSS, IMU, odometry or similar. Therefore, the initial position estimation was set manually using ROS visualization tool RVIZ "2D pose estimate" function which issues the "/initialpose" topic based on the user pose selection on the 3D map rendering. Correct and accurate localization usually took several iterations with RVIZ "2D pose estimate" function until satisfactory results were achieved. In Autoware's implementation this is the usual initialization method.

To help reducing the 3D data volume for localization, the `voxel_grid_filter` downsampling node was used with a leaf resolution of 2 m and a maximum laser range of 100 m. With this configuration and the grid volume described above, `ndt_matching` might require about 13 GB of RAM and heavy CPU usage.

5.3. Detection

`euclidean_cluster` is Autoware's implementation of PCL library's Euclidean clustering algorithm to group 3D pointcloud into 3D clusters which represent possible obstacles. We used a leaf size of 25 cm and a clipping window of 50 cm over the 3D sensor and 80 cm below the 3D sensor (including ground removal). `ground_filter` node was used to separate the 3D

(a) (b)

Fig. 10. Generated map using NDT for Tsukuba Challenge 2017: (a) top view, (b) side view (note the vertical error).

pointcloud into points which correspond to the ground and the rest of data.

NDT-based localization and clustering of 3D data are rather heavy tasks, in particular NDT localization is seriously affected by the OS process scheduler limiting it access to CPU and memory. We had several issues of localization convergence loss just by running other computational heavy processes like `euclidean_cluster`. To solve this we moved euclidean clustering computation to the available GPU on the computer.

5.4. Mission and Motion Planning

The `waypoint_saver` node was used to generate the robot trajectory. The robot estimated pose is computed by `ndt_matching` and `waypoint_saver` records the poses at 1 m intervals. The trajectory data include coordinates relative to the initial position, direction and velocity but for navigation we ignored the latter. `waypoint_follower` was used to generate the set of twist commands (linear and angular velocities) to follow the trajectory.

The `way_planner` node, part of OpenPlanner by Hatem et al. [9] and integrated on Autoware, was used for global planning, it can compute trajectories using vector maps for vehicles but in our case using the above trajectory. **Fig. 11** shows the robot trajectory for our participation in Tsukuba Challenge 2017.

The `dp_planner` node, also part of OpenPlanner, was used for local planning, from the current robot position to some target position (goal), the goal was defined also using RVIZ "2D Nav Goal" tool. The path planning is computed based on robot maximum velocity, number of rollouts (possible trajectories to avoid an obstacle, turn at a curb, change lanes, see **Fig. 19** in Section 7.2.3), avoidance distance, lateral and longitudinal safety margins, etc. Careful configuration of `dp_planner` node was necessary for our mobile robot: compared with a vehicle the robot is smaller and slower, yet it is more dynamic and moves in tighter spaces than a vehicle. **Fig. 19** shows `dp_planner` making a obstacle avoidance decision, the highlighted rollout corresponds to the path the robot will follow.

The `pure_pursuit` node is Autoware's implementation of the PurePursuit trajectory following algorithm [30, 31], which computes a circumference arc to smoothly join the current robot position and some goal point in the trajectory at a lookahead distance and arc radius. The `pure_pursuit` node computes the twist data (linear and angular information) for the robot to move.

The `twist_filter` node filters the acceleration for safe robot operation.

6. E2E with Autoware Backup

As shown in **Fig. 4**, an evaluator component gets target robot control data (linear velocity v and angular velocity ω pairs) generated by both E2E ($\mathscr{C}_E = \langle v_E, \omega_E \rangle$) and Autoware ($\mathscr{C}_A = \langle v_A, \omega_A \rangle$), and the current robot pose and current velocity. The evaluator then analyses the input data and will send either $\mathscr{C}_E$ or $\mathscr{C}_A$ robot control data to the SH-Spur board to drive the motors. Both $\mathscr{C}_E$ and $\mathscr{C}_A$ are synchronized using ROS's timestamps.

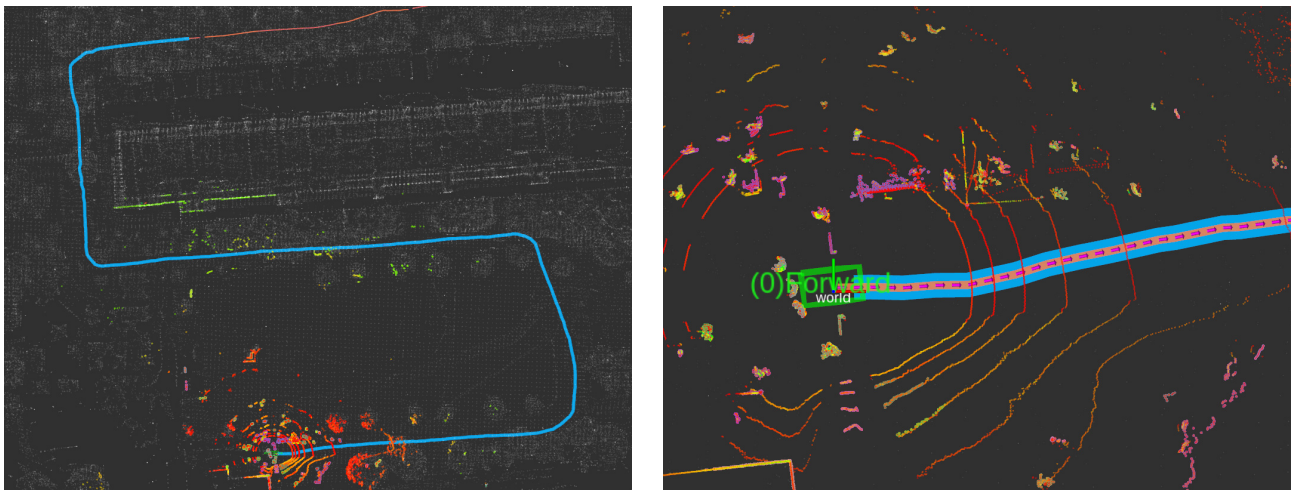
Both systems, E2E and Autoware are running in parallel so robot control data roughly corresponds to the same target waypoint in robot local coordinates. This ensures robustness for robot operation, in case one system fails the other can keep the robot in operation, and if both fails the evaluator can stop the robot.

Details are given on **Algorithm 1**. The criteria to switch between E2E and Autoware is simple, using current pose $\mathscr{P}$ if the robot is deviated over some threshold $\mathscr{T}$ from the current waypoint $\mathscr{W}$, the evaluator will switch robot control output to $\mathscr{C}_A$ for some amount of time $\mathscr{F}$ so that the position error is minimized by localization and path planning, and then returns back to $\mathscr{C}_E$.

The main parameters used in our system are summarized in **Table 2**.

7. Experimental Results

In the following subsections we present results for experiments held at different evaluation locations, namely the Nagoya University course (Section 7.1) and the Tsukuba Challenge 2017 course (Section 7.2), shown in **Fig. 3**. While both courses are rather short compared with the complete TC2017 course (**Fig. 2**), the first is 211.43 m



(a) (b)

Fig. 11. (a) way_points and global trajectory of the robot using OpenPlanner, (b) enlargement of starting position.

Algorithm 1 Model evaluation and switching

while true **do**
  Read $\mathscr{C}_E \leftarrow \langle v_E, \omega_E \rangle$ /*from E2E*/
  Read $\mathscr{C}_A \leftarrow \langle v_A, \omega_A \rangle$ /*from Autoware*/
  Read $\mathscr{P}$ and $\mathscr{W}$ /*from Autoware*/
  $d \leftarrow \|T\|$ from $\mathscr{P}$ to $\mathscr{W}$ using Eqs. (5) and (6)
  if $d > \mathscr{T}$ **then**/*deviation is large*/
    $counter \leftarrow \mathscr{F}$
  end if
  if $counter \neq 0$ **then**
    $\mathscr{C} \leftarrow \mathscr{C}_A$
    $counter \leftarrow counter - 1$
  else
    $\mathscr{C} \leftarrow \mathscr{C}_E$
  end if
  Write $\mathscr{C}$ to robot controller
end while

and the second is 299.6 m, they are considered sufficient for our study on E2E autonomous navigation and to test our system.

7.1. Nagoya University Environment

This section details the experiments at Nagoya University, **Fig. 3(b)** shows the map for this course. We separated the evaluation scenarios into the following cases:

- Model-based (Autoware) only
- E2E only
- E2E with Autoware backup

In the following subsections we show the different results.

7.1.1. Model-Based

In **Fig. 12** we present the navigation performance of Autoware itself, as expected robot navigation follows correctly the defined trajectory. The robot starts at location $\langle 0\,\text{m}, 0\,\text{m} \rangle$ and followed the trajectory in a counter clockwise (CCW) direction. The path following positioning error (perpendicular distance from the current position to the target trajectory) expressed as the RMSE from the robot actual position with respect to the defined trajectory (waypoints) is about 12.8 cm, with a maximum error of

Table 2. Navigation system parameters.

	Component	Parameter	Value
Robot		max. linear velocity	0.5 m/s
		max. angular velocity	0.4 rad/s
		max. acceleration	0.02 m/s^2
		max. angular accel.	0.04 rad/s^2
		dimensions (width,depth,height)	$\langle 0.5, 0.7, 1.2 \rangle$ m
E2E		linear velocity v_E	0.2 m/s
		image width	200 pixels
		image height	66 pixels
Autoware	setup	baselink localizer	$\langle 0.3\,\text{m}, 0\,\text{m}, 1.2\,\text{m}, 0^\circ, 0^\circ, 0^\circ \rangle$
		$\langle x, y, z, roll, pitch, yaw \rangle$	
	voxel_grid_filter	leaf size	2 m
	voxel_grid_filter	measurement range	100 m
	ground_filter	max slope	15°
	ground_filter	gap threshold	0.17 m
	ground_filter	clip threshold	0.5 m
	ndt_matching	error threshold	2 m
	ndt_matching	resolution	1 m
	ndt_matching	max. iterations	30
	euclidean_cluster	remove ground	true
	euclidean_cluster	leaf size	0.25 m
	euclidean_cluster	cluster min. size	30
	euclidean_cluster	clip. min. height	−0.8 m
	euclidean_cluster	clip. max. height	0.54 m
	euclidean_cluster	use GPU	true
	dp_planner	max. velocity	0.5 m/s
	dp_planner	plan distance	10 m
	dp_planner	rollouts number	6
	dp_planner	avoiding distance	2 m
	dp_planner	avoiding limit	0.5 m
	dp_planner	lateral safety	0.2 m
	dp_planner	longitudinal safety	0.5 m
Evaluator		Model switch distance threshold $\mathscr{T}$	0.4 m
		Autoware model switching time	10 s

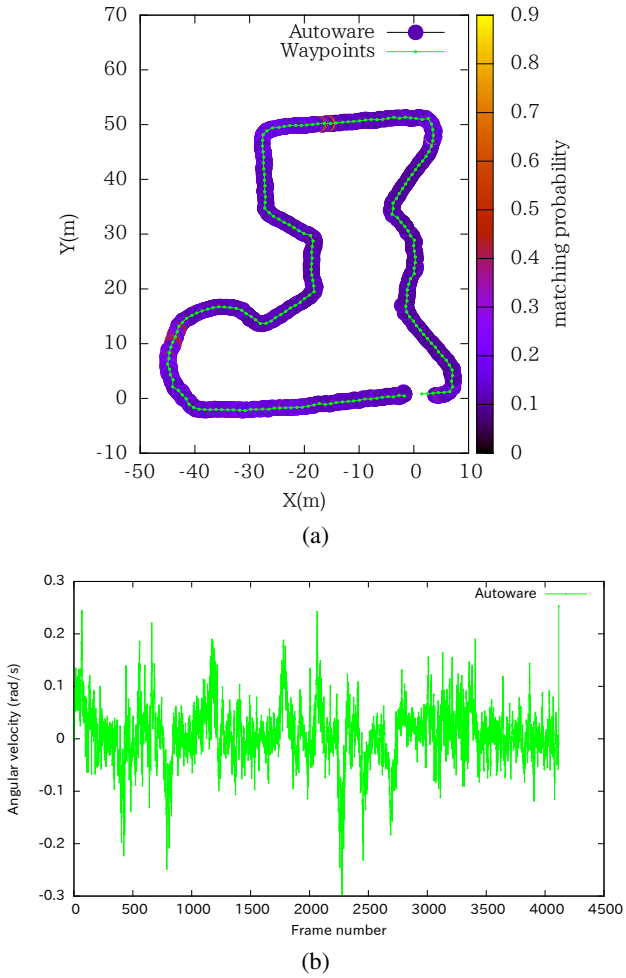


Fig. 12. Autoware's behaviour: (a) shows the traveled path and (b) the angular velocity response.

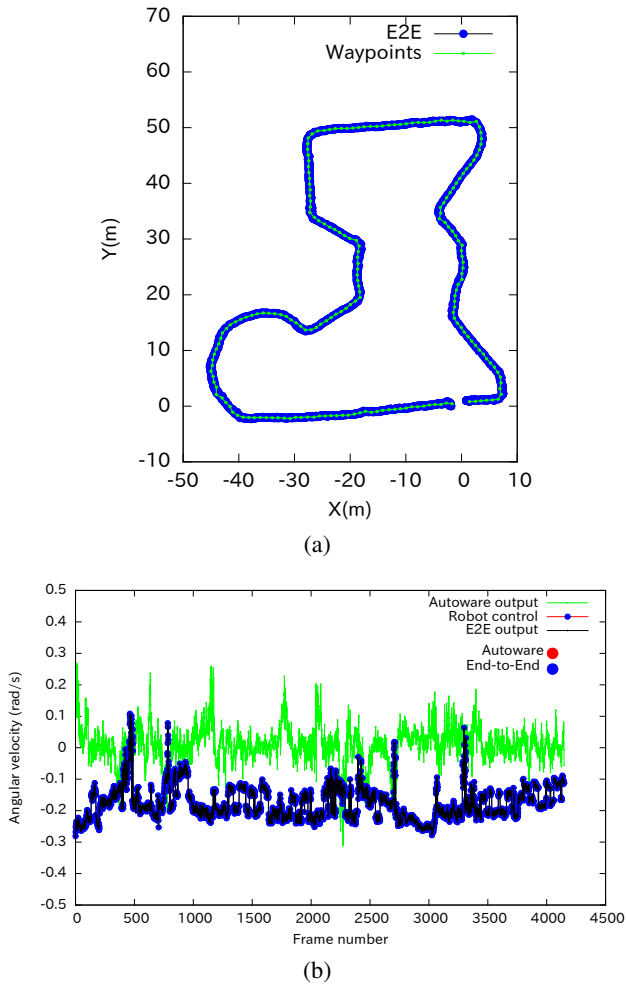


Fig. 13. E2E behaviour: (a) shows the traveled path, (b) the angular velocity response.

75.1 cm. **Fig. 12(b)** also shows the angular velocity (ω) profile send to the robot for during the complete course. As steering decisions are affected by localization accuracy and small changes in robot position every 100 ms (localization period), the ω profile itself is not smooth. As the robot steering response time is limited, there are no noticeable steering jerks.

Please note that the positioning error here is not considered as absolute measure of performance but to give a score to one single experiment run. In fact, positioning error is relative to the performance of robot localization software, which in turn depends on the the dynamics of the outdoor environment: parked vehicles during map generation not present on localization tests, other moving vehicles, seasons change in foliage covered areas, noise on 3D LiDAR measurements, and so on, have a negative effect on instantaneous localization results. Therefore, the robot will not be able to follow the trajectory with same positioning results.

Figure 12(a) also shows the matching probability, i.e., the probability score of the transformation which is applied to the input cloud to match the given map, taken from the score function proposed by Magnusson [33]. This matching probability serves to evaluate the stability of our robot localization, lower probability values having a direct relation with possible localization errors. On the Nagoya University environment the probability score was rather low and during experiments localization often failed. One distinctive attribute of this environment, in comparison with TC2017, is the pedestrian roundabouts and semi-circular amphitheatre (see **Fig. 3(b)** for details). In such circular spaces finding the correct robot heading from 3D LiDAR data is challenging.

7.1.2. End-to-End

On the second experiment, we test the performance of E2E system to steer the robot on the designated trajectory, navigation results are shown in **Fig. 13**. Again, the robot starts at location $\langle 0\,\text{m}, 0\,\text{m}\rangle$ and followed the trajectory in CCW direction. The RMSE of path following positioning with respect to the defined trajectory (waypoints) is about 12.02 cm, with a maximum error of 66.42 cm.

Please note that E2E training uses data from different locations at Nagoya University, some portions of this course were also included. However, the complete data from this particular course and driving direction was not

used for training. Furthermore, the training model was created for different illumination and environmental conditions than the present experimental results.

As stated on Section 4, E2E uses a constant linear velocity (υ) and the CNN generates the steering direction (ω) used to drive the robot. **Fig. 13(b)** also shows the angular velocity (ω) profile send to the robot for during the complete course. To compare results, the independent $\langle \upsilon, \omega \rangle$ pair values from both E2E and Autoware, and also the final robot control values, were recorded, however, only that of E2E was used to control the robot, as shown in the figure.

7.1.3. E2E with Autoware

On the third experiment, we show the performance of our E2E with Autoware backup system on the designated trajectory, navigation results are shown in **Fig. 14(a)** running the same CCW direction. The RMSE of path following positioning with respect to the defined trajectory (waypoints) is about 13.88 cm, with a maximum error of 80.68 cm.

Figure 14(b) also shows the angular velocity (ω) profile send to the robot for during the complete course. Robot steering angular velocity comes from both Autoware and E2E. To compare results, the independent $\langle \upsilon, \omega \rangle$ pair values from both E2E and Autoware, and also the final robot control values, were recorded; we also record which subsystem output (Autoware's or E2E's) was used. **Fig. 14(b)** shows in red the actual robot's control values from Autoware, which matches the independent values from the same system; blue shows the actual control values from E2E, also matching the independent values. Similar to the E2E only results on Section 7.1.2, we can clearly identify the system (Autoware or E2E) controlling the robot at every moment, E2E is in charge most of the time.

In this experiment, we used a threshold value of 60 cm to switch between models. Regarding results in **Fig. 14(a)**, the robot starts roughly at coordinates $\langle 2\text{ m}, 1\text{ m} \rangle$ and the distance to the next waypoint is larger than the threshold therefore it starts running in Autoware mode; after the defined number of frames, it returns to E2E mode. At coordinates $\langle 3.5\text{ m}, 47\text{ m} \rangle$, there is a brief localization error (i.e., NDT failed to correctly match the input scan with the map) and the robot position jumps about 1.2 m in the horizontal axis (shown in detail in **Fig. 14(c)**, position error is shown with perpendicular lines to the target trajectory). As such, the distance to next waypoint is larger than the threshold and navigation switches to Autoware; the sudden steering value to return to the trajectory is also shown in **Fig. 14(b)**. Please note this particular model switch was not due to E2E deviation from the target trajectory, however this case shows the potential of the E2E with Autoware backup to recover from other sources of error.

The enlarged area in **Fig. 14(c)** also shows an important aspect of the localization system used: estimated robot position is never steady but varies by several centimeters

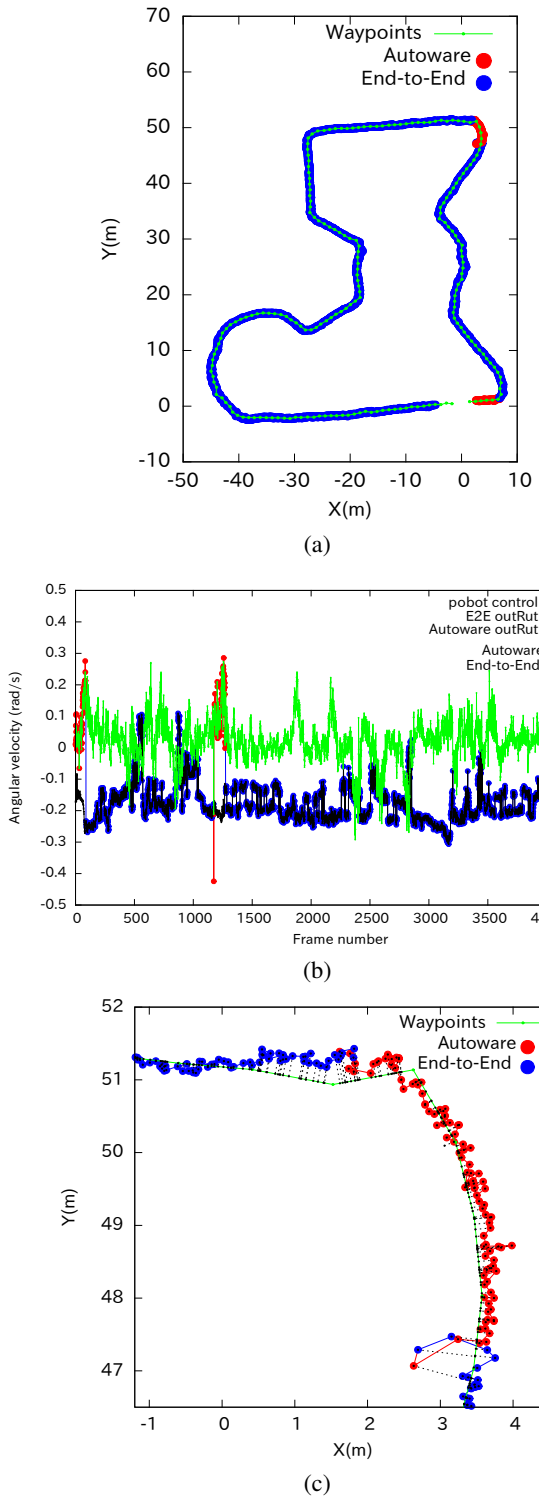


Fig. 14. System behaviour: (a) shows the traveled path with E2E and Autoware switching, (b) the angular velocity response, and (c) shows a situation of model switching.

even when the robot is standing still. As already stated, the dynamics of outdoors mean the environment is never the same; position is estimated every 100 ms, every new scan data is different from the previous, thus the position oscillations observed.

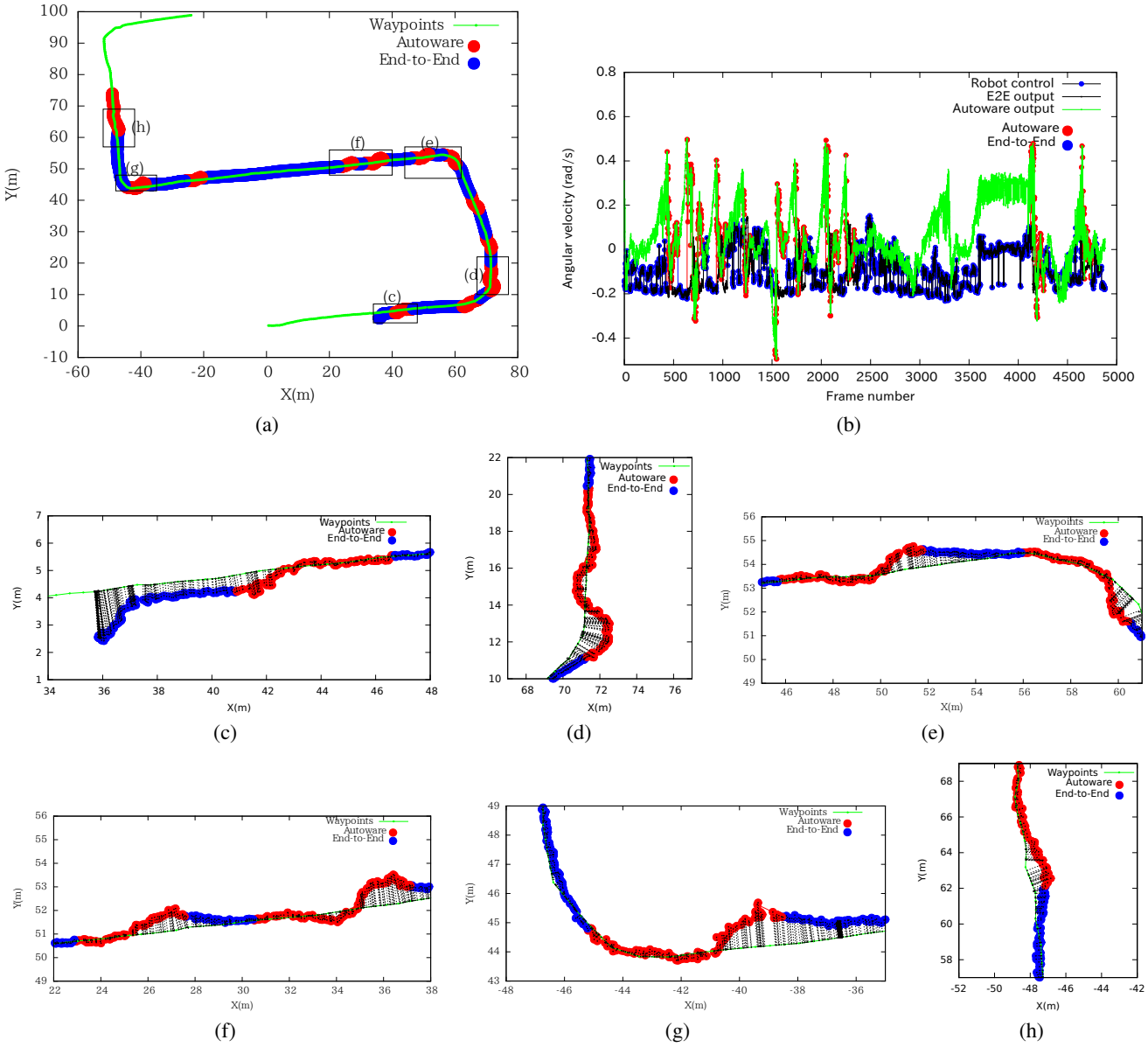


Fig. 15. System behaviour: (a) shows results for the TC2017 course, (b) the angular velocity response. Different areas during model switch are highlighted and details on (c)–(h), position error is shown with perpendicular lines.

7.2. Tsukuba Environment

Seven experimental runs were performed on the TC2017 course on July 8th, September 23rd, October 14th–15th, November 3rd–4th and the finals on November 5th, with environment changes from Summer to Autumn. On each occasion, we recorded rosbag log files for all sensor data, thus were able to create navigation maps and running trajectories already discussed.

7.2.1. E2E with Autoware Backup System Evaluation

Having tested the performance of the different components of our navigation system, we set to evaluate the E2E with Autoware system on the objective TC2017 course (**Fig. 3(a)**) of this study. Initially the CNN model trained at Nagoya University was used on the TC2017 course and it performed with acceptable results. But with seasons changing we used camera images captured on TC2017 environment to fine tuned our E2E model. Results are shown in **Fig. 15**.

Figure 15(a) shows the response of model switching during trials at TC2017 in terms of position and used model. E2E model ran about 70% of the experiment and model-based overtook control just to return the robot back to correct trajectory. The robot localization data starts 35 m past the first waypoint (bottom left) and a slightly to the right of the road due technical issues with the controlling computer, and it ends before the last waypoint (top right) because extra clearance was added during target path configuration. The RMSE of path following positioning from the robot actual position with respect to the defined trajectory (waypoints) is about 46 cm, with a maximum error of 1.84 m. **Fig. 15(b)** shows the angular

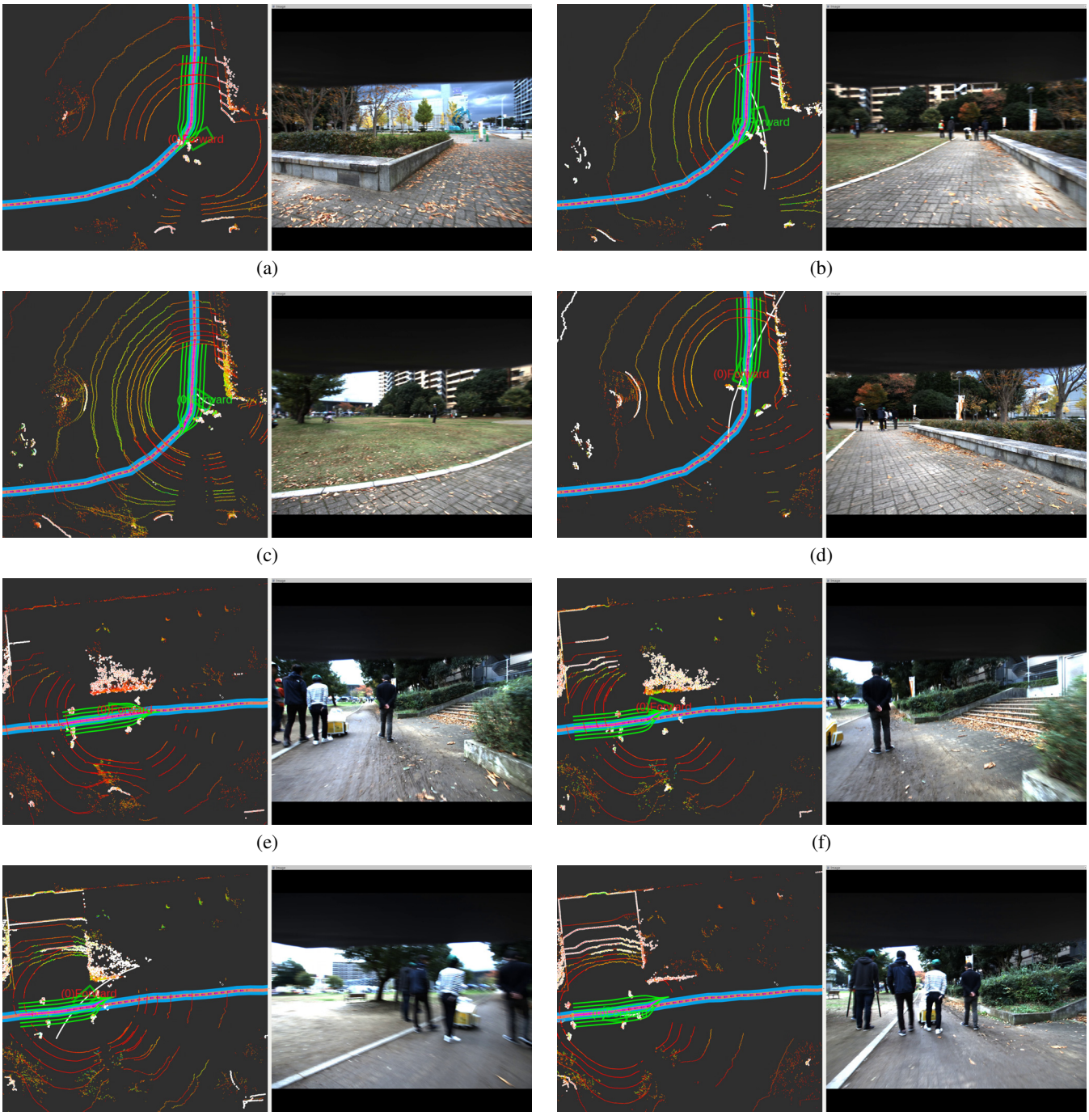
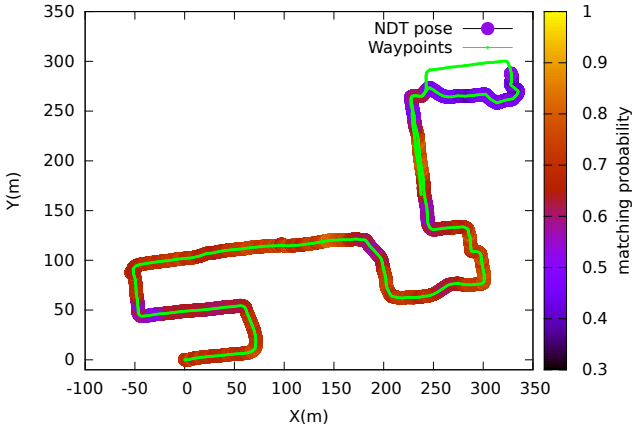


Fig. 16. (a)–(d) scenes of robot overshoot situation in **Fig. 15(d)**, and (e)–(h) scenes of obstacle avoidance situation in **Fig. 15(f)**.

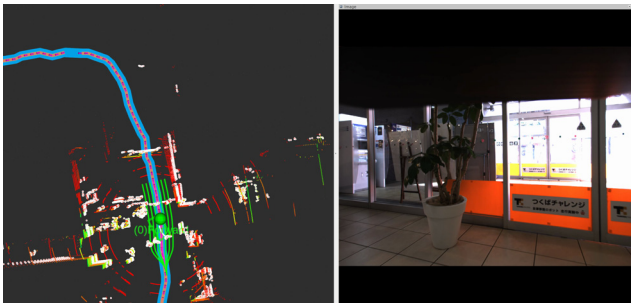
velocity response.

Figures 15(c)–(h) show some selected areas were E2E navigation caused the robot to deviate over some defined threshold $\mathscr{T}$ and after some amount of time forcing a switch to Autoware's navigation, the positioning error with respect to target waypoints is shown with dark dashed lines. When E2E deviates and Autoware overtakes to return back to target trajectory, sometimes the robot overshoots (as shown in **Figs. 15(d)** and **(f)**) and takes some extra cycles to return back to the normal course. This problem is due to parameter setting, the curvature to return to the target trajectory is too abrupt when it should have been more smooth, proper parameter tuning is necessary.

Overshooting is illustrated in **Figs. 15(a)–(d)**: during the first left turn, the robot has deviated and computes a rather tight curvature to return back to trajectory (shown in **Figs. 16(b)** and **(d)**), it overshoots and has to compute another curve to return back. There was also a mechanical issue in the rear chaster wheel which kept the robot oscillating left and right. **Figs. 16(e)–(h)** also illustrates obstacle avoidance in Autoware while also recovering from



(a)



(b)

Fig. 17. Over 1km autonomous run during trials, (a) shows the traveled path and localization performance, and (b) shows scenes of the stop situation.

E2E course deviation.

One important difference from the Nagoya University environment with the TC2017 environment is the lack of other robots and crowds in Nagoya. Camera images recorded from TC2017 include scenes of our robot avoiding and overtaking other robots and pedestrians and such images were part of the training set. The unintended consequence was a tendency of our robot to deviate and go towards the right.

7.2.2. Trial Runs

Figure 17 shows our maximum autonomous performance achieved during trials: a total traveled distance of 1047.3 m, in this case the robot ran on Autoware mode. This path exceeds the defined trajectory for E2E navigation but is used to evaluate the performance of model-based (Autoware) navigation. The robot was stopped with the emergency stop button due possible collision with some glass doors, **Fig. 17(b)** shows the corresponding scene, the trajectory itself is set towards the left door pane (not the center as it should), the robot deviated from its trajectory by 10 cm.

The reason behind this navigation problem is the map itself. The map was built with a rather coarse resolution of 1 m suitable for outdoors navigation, but when it comes to indoor navigation a higher resolution is desirable due to shorter ranges and presence of walls and other artificial structures. Our map was created with single resolution and no other approach was devised to overcome the indoor limitation. The second issue which affects our map is the already mentioned accumulated vertical error (see **Fig. 10(b)**), when projected in 2D the map is slightly shorter than it would be with correct vertical values. Also the trajectory was generated based on that same map, therefore it has errors.

In general, the RMSE of path following positioning with respect to the defined trajectory (waypoints) is about 55.14 cm, with a maximum error of 2.21 m.

Figure 17(a) includes also the matching probability. Using as reference the satellite view on **Fig. 2**, the matching probability was lower when the robot moved between buildings in the shopping center (top right part), also on the first right turn (coordinates $\langle -40\text{ m}, 50\text{ m}\rangle$) which is a rather crowded tent area for TC2017 participants, and some other scattered places with large number of people moving, restricting 3D scanning and thus lower localization accuracy.

7.2.3. Finals

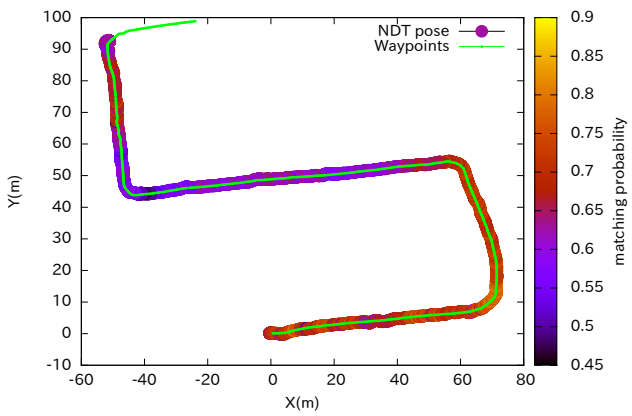
The TC2017 finals is where all participant teams officially prove their systems and a score is assigned based on total performance. It is also the moment where robot safety measures are put to the maximum test and systems deemed unsafe are discouraged by the organizers. E2E for robot navigation was not used during TC2017 finals for diverse reasons listed below:

- To increase safety we decided to reduce the switching threshold $\mathscr{T}$ value; however, illumination and weather conditions were contrary to the trained model and it performed erratically.
- On the weekend of the TC2017 finals a rather large crowd of participants, robots, judges and families, visited the park area we designated as target for this study; this situation was outside of CNN training conditions.
- The tendency to deviate towards the right of the CNN model was considered unsafe.

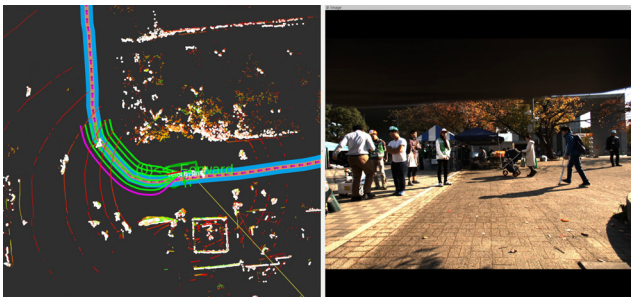
TC2017 finals special case is considered outside the scope of this E2E research. Therefore, we participated in TC2017 finals using our model-based subsystem, Autoware. Results of our participation are included here for completeness.

Figure 18 shows the performance of robot localization and path following on the selected trajectory for our participation. The robot localization data starts a few centimeters before the first waypoint (coordinates $\langle 0, 0\rangle$) due to preparations on the start line, and it ends before the last waypoint (top right) because extra clearance was added. The RMSE of path following positioning with respect to the defined trajectory (waypoints) is about 17.68 cm, with a maximum error of 98.4 cm.

Figure 18(a) shows the matching probability. Comparing the environment scenes in **Fig. 3** and the localization



(a)



(b)

Fig. 18. Navigation performance during TC2017 finals, (a) shows the traveled path and localization performance, and (b) shows scenes of obstacles negotiation.

results in **Fig. 18**, the matching probability was lower under the tree canopy and around the tents areas possibly due to environment changes from the time the map was built and the TC2017 finals. In particular, right before the right turn, there is a tent area full of participant teams members and other unrelated pedestrians on their business. As shown in **Fig. 18(b)**, the robot will turn right and chooses a path to avoid some obstacles, but localization in this dynamic area becomes difficult so matching probability decreases.

Our team completed the intended target trajectory for E2E navigation shown in **Fig. 18**. However, as for the Tsukuba Challenge 2017 defined goal, an impossible situation forced us to emergency stop the robot and were unable to continue the rest of TC2017 course, our total traveled distance was recorded by the referee as 280 m.

Figure 19 shows scenes of a situation of emergency stop during the finals: the robot encountered a large and dynamic group of people mostly very curious children and parents (**Figs. 19(a)** and **(b)**). The group gracefully decided to open a space for the robot to pass and the robot finds an avoidance path (**Figs. 19(c)** and **(d)**), when the robot is executing path following a person suddenly forced his way between the children and right in front of the robot (**Figs. 19(e)** and **(f)**) as close to 10 cm from the robot. As our robot cannot decelerate fast enough, the robot operator decided to press the emergency button to avoid hurting this person.

7.3. Discussion

After months preparing for the Tsukuba Challenge participation, we have several experiences and lessons learned to enrich and contribute with the research community.

7.3.1. E2E

One definitive deficiency on our E2E is the limited and ineffective training. During data capture for training at our environment in Nagoya University, we did not include crowds, other robots, overtaking people, etc. Illumination changes during the day and weather were not properly accounted, transformations of the environment with seasons change were ignored. All these topics should be addressed during training of a better model on a future work.

Regarding illumination and weather conditions, shadows and strong illumination on sunny/clear weather difficult detecting the environment and poor steering direction is generated. A hood to block and shadow direct sunlight was tried during TC2017, at the expense of occluding parts of the scene, and with no important advantages (the portion occluded by the hood on the images was removed during training and operation of the CNN). Other hardware solutions include polarized filters, and software methods like the illumination invariant image transformation (see Maddern et al. [35]), will be considered on a future development. Thus we can augment the training data with different illumination levels and filtering for a more robust and versatile model.

While we have demonstrated the performance of E2E for robot navigation, it cannot steer the robot to avoid obstacles nor stop the robot in case of possible collision. However, from the data collected and used for fine tuning of our CNN, the robot tends to go to the right as detailed in **Figs. 15(a)** and **16(e)–(h)**. While this is an unwanted behaviour during simple robot steering, it suggests the possibility of E2E to achieve obstacle avoidance. A possible implementation will have a CNN trained for robot steering and a second CNN or LSTM for obstacle and collision avoidance. Both networks can be joined at the FC layers to generated the proper steering signal for both cases.

7.3.2. Autoware

While Autoware is a very complete set of software tools for autonomous vehicles, it does not work out-of-the-box for mobile robots. Vehicles move with clearly defined rules, run on clearly marked and exclusive surfaces, with governing laws. Instead, the smaller mobile robots for outdoor environments have all freedom as pedestrians do: sudden stops, 180° reversals, moving over any drivable surface, following different trajectories, and even free motion direction of holonomic robots.

As such, we had to customize several pieces of Autoware, in particular localization and path planning. Some of the modifications were effective to steer the robot on narrow spaces and avoid close obstacles. However, path planning still has some behaviour, meant for vehicles, that

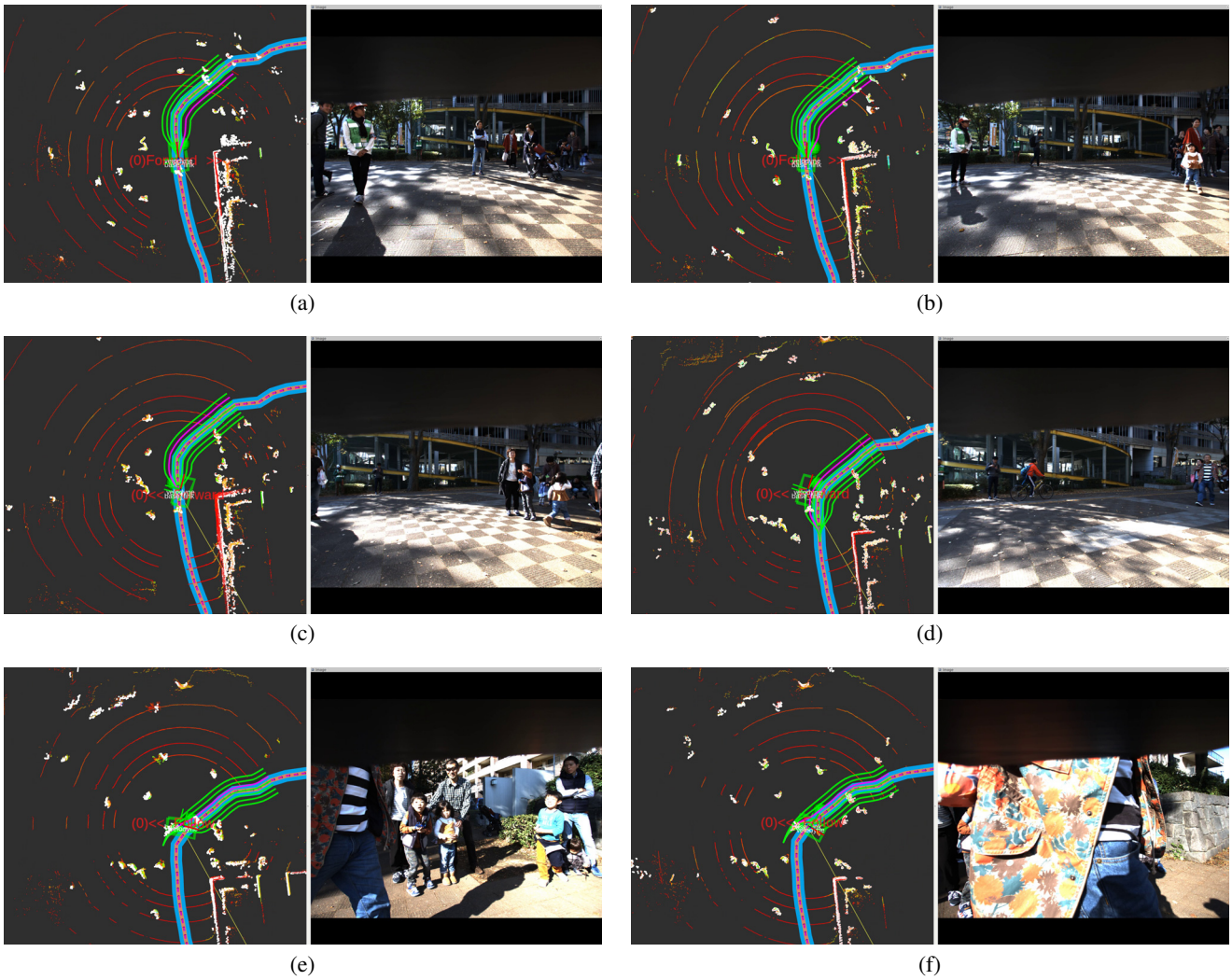


Fig. 19. Scenes of emergency stopping situation at TC finals, LiDAR pointcloud is shown in color points, clusters for obstacle avoidance are shown in white points, trajectory and navigation rollouts are also shown.

we did not identify in time. For example, `dp_planner` updates the search area (rollouts) with some defined frequency, obstacle avoidance will not work between updates; this is one of the reasons why obstacle avoidance was not handled properly on TC2017 finals (**Fig. 19**). Obstacle detection depends on the `euclidean_cluster` 3D pointcloud clustering, a very heavy process, and running it on GPU alleviated some of the CPU burden, but it cannot filter correctly the ground data; therefore we used an extra node for that purpose, the `ground_filter`, adding load upon the CPU.

Autoware's default map generation tool is the `ndt_mapping`, as it cannot process pointclouds on real time we used instead `ndt_mapping_tku`. It is possible to play rosbag files at a fraction of their recording frequency (ex., 1/10th) and `ndt_mapping` will render good quality maps. However, very detailed and high resolution maps means more processing power is required to match a new 3D scan from the LiDAR and infer the robot position. After diverse experiences, we used 1 m resolution maps to achieve good localization and realtime performance. Subsampling of the input 3D scan pointcloud is also necessary to improve response.

On the other hand, even if we have good performance on outdoors, the price to pay is poor localization in indoors, as discussed in Section 7.2.2. `ndt_matching` seems to have issues estimating the correct vertical position (Z-axis), thus the deformity in our maps, such problem needs to be addressed. Multiple resolution maps to improve matching convergence and adapt to environment conditions is a candidate direction for future implementations. A hybrid localization approach combining two or more existing technologies is also possible.

From experience we found that `ndt_matching` localization failure depends on the available CPU for its work: if this node is running in parallel with other processor intensive tasks, localization failure is highly likely; thus our efforts to reduce loads, use rather coarse maps, etc. We found that matching probability value (score) of `ndt_matching`, detailed in Section 7.1.1, can be used to let the robot know when localization failed and stop it as a safety measure. We added a `ndt_matching_monitor`

node to watch the health of the localization system: if the score is under some threshold, for a configurable number of frames, the `ndt_matching_monitor` will send a signal to the evaluator to reduce the robot speed; also if the score is under a second threshold stop the robot.

The `ndt_matching_monitor` also attempts to initialize `ndt_matching` (via the "/initialpose" topic) with the last known good position, computed using an average filter over a history of past positions with good score. A better solution would be to keep a particle filter to predict the robot position for initialization.

The 3D LiDAR (a Velodyne VLP16) was mounted on the robot about 1.2 m from the ground to have good coverage of the environment. However, for its height and limited field-of-view, it cannot detect objects close to the robot (less than 3.3 m) specially those at low height: cones, curbs, shrubs, other small robots become invisible. We attempted two last-minute solutions, one was to use the 2D LiDAR (a Hokuyo UTM-30LX) for stop the robot on very close objects; the second was to fuse the pointclouds of 2D and 3D LiDARs, then use the combined one with `euclidean_cluster` to detect low height obstacles. The second solution was not effective since `euclidean_cluster` also use the ring information (number of the laser) of the 3D LiDAR.

7.3.3. Evaluator

The implemented system has at its core an evaluator node which collects $\langle v, \omega \rangle$ pairs from E2E and Autoware and drives the robot, with preference for E2E signals. Model switching is simple and effective, yet its implementation is prone to errors. We discussed in Section 7.1.3 about oscillations in robot position estimation; small position changes and some incorrect localization results (such as **Fig. 14(c)**) may affect the distance to the next waypoint, and cross the switching threshold $\mathscr{T}$ unnecessarily.

A better implementation should consider smoothing the current robot position over time. Reacting to position changes with the same response time as the localization software (100 ms) is not required for the evaluator, thus a short history of robot positions can be used to decide when to switch.

7.3.4. Others

It was a great performance boost to use EXT4FS to record rosbag log files on the SSD drive instead of the default NTFS used by most manufacturers. While the default file system may suit some applications, having to record images from several cameras (2 in our case), 3D LiDAR point clouds, partial results of running programs (like current position or coordinates transformation), demand higher performance from the file system, and just a fast media is not sufficient.

Another issue with external SSD drives is their limited tolerance to temperature changes. While the external temperature has some influence, the internal temperature inside the robot case or close to the controlling computer deteriorates writing speed. Isolating the SSD drive from warm components was important.

One anecdotal incident was the fracture of a motor gear during experiments, this happened running over a pedestrian crossing with cars waiting for the semaphore. The NUIV robot is over 10 years old, already has faced several editions of Tsukuba Challenge. Some of its parts were 3D printed using strong materials; after years of operation the gears have worn out, new gears were 3D printed and replaced.

8. Conclusions

In this work we presented our navigation approach for an autonomous mobile robot, tested during our participation in Tsukuba Challenge 2017. Our system uses two diametrically opposite approaches: E2E, based on deep neural networks, to steer the robot directly from sensors to actuators; also a conventional model-based system based on deterministic rules and state-of-the-art algorithms. The E2E with Autoware support runs both systems in parallel and an evaluator chooses the best control signals to drive the robot towards the destination.

E2E navigation was set as the main topic of study for our participation: to achieve robot steering in complex environments including turning at intersections. However its current implementation lacks crucial features for safe robot navigation, in particular obstacle avoidance, fundamental on a crowded environment such as TC2017. Our backup system overcomes this problem, model-based adds safety features to complement E2E.

For the model-based system we chose a software platform for autonomous vehicles, Autoware. While most of the technology for autonomous cars comes from robotics research, vehicles run with clear rules, in exclusive and marked surfaces, and some basic behaviour can be assumed. Autonomous mobile robots are not so bounded, thus adapting a vehicle software to use on a mobile robot has some drawbacks. Nevertheless, the effort was done and a version of Autoware for autonomous mobile robots is available.

E2E for robot navigation with basic obstacle and collision avoidance, tolerance to illumination and weather changes, with a better trained model on diverse environments and similar conditions to TC2017, will be studied on a future work. Some modifications in path planning and localization on Autoware are still necessary to better suit autonomous mobile robots needs.

All source code (including E2E, modified version of Autoware to suit our robot, robot control), parameters, maps, and videos showing the performance of our participation are available at https://github.com/MAVRG/TC2017. Additionally, some rosbag log files, maps and navigation trajectories are available at Rosbag Store [e] https://rosbag.tier4.jp/rosbag_details/?id=212.

Acknowledgements

This work was supported by the Japan Science and Technology Agency(JST) project on Open Innovation Platform with Enterprises, Research Institute and Academia (OPERA). We are deeply grateful for the generous support from TierIV, Inc.

References:

[1] S. Yuta, H. Hashimoto, and H. Tashiro, "Tsukuba challenge – real world robot challenge (rwrc): Toward actual autonomous robots in our daily life," 25th Annual Conf. of the Robotics Society of Japan, 3D19, 2007.

[2] S. Yuta, M. Mizukawa, H. Hashimoto, H. Tashiro, and T. Okubo, "Tsukuba challenge 2009 – towards robots working in the real world: Records in 2009 –," J. of Field Robotics, Vol.23, No.2, pp. 201-206, 2011.

[3] S. Yuta, "Open experiment of autonomous navigation of mobile robots in the city: Tsukuba challenge 2014 and the results," J. of Robotics and Mechatronics, Vol.27, No.4, pp. 318-326, 2015.

[4] Y. Morales, E. Takeuchi, A. Carballo, W. Tokunaga, H. Kuniyoshi, A. Aburadani, A. Hirosawa, Y. Nagasaka, Y. Suzuki, and T. Tsubouchi, "1km autonomous robot navigation on outdoor pedestrian paths "running the tsukuba challenge 2007"," IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), pp. 219-225, 2008.

[5] Y. Morales, A. Carballo, E. Takeuchi, A. Aburadani, and T. Tsubouchi, "Autonomous robot navigation in outdoor cluttered pedestrian walkways," J. of Field Robotics, Vol.26, No.8, pp. 609-635, 2009.

[6] N. Akai, K. Yamauchi, K. Inoue, Y. Kakigi, Y. Abe, and K. Ozaki, "Development of mobile robot "SARA" that completed mission in real world robot challenge 2014," J. of Robotics and Mechatronics, Vol.27, No.4, pp. 327-336, 2015.

[7] A. Sujiwo, T. Ando, E. Takeuchi, Y. Ninomiya, and M. Edahiro, "Monocular vision-based localization using ORB-SLAM with LIDAR-aided mapping in real-world robot challenge," J. of Robotics and Mechatronics, Vol.28, No.4, pp. 479-490, 2016.

[8] A. Sujiwo, E. Takeuchi, L. Y. Morales, N. Akai, H. Darweesh, Y. Ninomiya, and M. Edahiro, "Robust and accurate monocular vision-based localization in outdoor environments of real-world robot challenge," J. of Robotics and Mechatronics, Vol.29, No.4, pp. 685-696, 2017.

[9] H. Darweesh, E. Takeuchi, K. Takeda, Y. Ninomiya, A. Sujiwo, L. Y. Morales, N. Akai, T. Tomizawa, and S. Kato, "Open source integrated planner for autonomous navigation in highly dynamic environments," J. of Robotics and Mechatronics, Vol.29, No.4, pp. 668-684, 2017.

[10] M. Suzuki, T. Saitoh, E. Terada, and Y. Kuroda, "Near-to-far self-supervised road estimation for complicated environments," Int. Federation of Automatic Control (IFAC) Proc., Vol.43, No.18, pp. 689-694, 2010.

[11] K. Hosaka and T. Tomizawa, "A person detection method using 3d laser scanner – proposal of efficient grouping method of point cloud data –," J. of Robotics and Mechatronics, Vol.27, No.4, pp. 374-381, 2015.

[12] T. Tomizawa and R. Moriai, "Using difference images to detect pedestrian signal changes," J. of Robotics and Mechatronics, Vol.29, No.4, pp. 706-711, 2017.

[13] K. Shigematsu, Y. Konishi, T. Tsubouchi, K. Suwabe, R. Mitsudome, H. Date, and A. Ohya, "Recognition of a traffic signal and a search target using deep learning for tsukuba challenge 2016," 17th SICE SI Division Annual Conf., pp. 1398-1401, 2016 (in Japanese).

[14] S. Bando, T. Nakabayashi, S. Kawamoto, and H. Bando, "Approach of tsuchiura project in tsukuba challenge 2016," 17th SICE SI Division Annual Conf., pp. 1392-1397, 2016 (in Japanese).

[15] R. Mitsudome, H. Date, A. Suzuki, T. Tsubouchi, and A. Ohya, "Autonomous mobile robot searching for persons with specific clothing on urban walkway," J. of Robotics and Mechatronics, Vol.27, No.4, pp. 649-659, 2017.

[16] Y. LeCun, U. Muller, J. Ben, E. Cosatto, and B. Flepp, "Off-road obstacle avoidance through end-to-end learning," Advances in Neural Information Processing Systems 18 (NIPS), pp. 739-746, MIT Press, 2005.

[17] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," arXiv preprint arXiv:1604.07316, 2016.

[18] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," The IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 2174-2182, 2017.

[19] S. Yang, W. Wang, C. Liu, W. Deng, and J. K. Hedrick, "Feature analysis and selection for training an end-to-end autonomous vehicle controller using deep learning approach," pp. 1033-1038, 2017.

[20] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada, "An open approach to autonomous vehicles," IEEE Micro, Vol.35, No.6, pp. 60-68, 2015.

[21] D. A. Pomerleau, "ALVINN: An autonomous land vehicle in a neural network," Advances in Neural Information Processing Systems 1 (NIPS), pp. 305-313, Morgan-Kaufmann, 1988.

[22] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," Neural Computation, Vol.3, No.1, pp. 88-97, 1991.

[23] D. A. Pomerleau, "Knowledge-based training of artificial neural networks for autonomous robot driving," Robot learning, pp. 19-43, 1993.

[24] T. M. Jochem, D. A. Pomerleau, and C. E. Thorpe, "Vision-based neural network road and intersection detection and traversal," IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS) 'Human Robot Interaction and Cooperative Robots,' Vol.3, pp. 344-349, 1995.

[25] M. Bajracharya, A. Howard, L. H. Matthies, B. Tang, and M. Turmon, "Autonomous off-road navigation with end-to-end learning for the LAGR program," J. of Field Robotics, Vol.26, No.1, pp. 3-25, 2009.

[26] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," IEEE Int. Conf. on Computer Vision (ICCV), pp. 2722-2730, 2015.

[27] S. Seiya, D. Hayashi, E. Takeuchi, C. Miyajima, and K. Takeda, "Evaluation of deep learning-based driving signal generation methods for vehicle control," Fourth Int. Symposium on Future Active Safety Technology: towards zero traffic accidents (FAST-zero), TuC-P1-4, 2017.

[28] Z. Zhang, "A flexible new technique for camera calibration. IEEE Transactions on Pattern Analysis and Machine Intelligence," Vol.22, No.11, pp. 1330-1334, 2000.

[29] R. Hartley and A. Zisserman, "Multiple View Geometry in Computer Vision," Cambridge University Press, 2003.

[30] O. Amidi and C. E. Thorpe, "Integrated mobile robot control," Mobile Robots V, Vol.1388, pp. 504-524, 1991.

[31] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Technical Report CMU-RI-TR-92-01, Carnegie-Mellon University, Robotics Institute, 1992.

[32] E. Takeuchi and T. Tsubouchi, "A 3-d scan matching using improved 3-d normal distributions transform for mobile robotic mapping," IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), pp. 3068-3073, 2006.

[33] M. Magnusson, "The three-dimensional normal-distributions transform: an efficient representation for registration, surface analysis, and loop detection," Ph.D. dissertation, Örebro Universitet, 2009.

[34] N. Akai, L. Y. Morales, T. Yamaguchi, E. Takeuchi, Y. Yoshihara, H. Okuda, T. Suzuki, and Y. Ninomiya, "Autonomous driving based on accurate localization using multilayer lidar and dead reckoning," IEEE Int. Conf. on Intelligent Trans. Systems (ITSC), pp. 1147-1152, 2017.

[35] W. Maddern, A. Stewart, C. McManus, B. Upcroft, W. Churchill, and P. Newman, "Illumination invariant imaging: Applications in robust vision-based localisation, mapping and classification for autonomous vehicles," Visual Place Recognition in Changing Environments Workshop, IEEE Int. Conf. on Robotics and Automation (ICRA), Vol.2, p. 3, 2014.

Supporting Online Materials:

[a] Tsukuba Challenge – Real World Robot Challenge – 2017. http://www.tsukubachallenge.jp/backnumber/tc2017/ [Accessed June 7, 2018]

[b] Autoware. https://github.com/CPFL/Autoware [Accessed June 7, 2018]

[c] TierIV. https://www.tier4.jp/ [Accessed June 1, 2018]

[d] SH-Spur of PC Yamabico Project. http://sourceforge.jp/projects/pc-ymbc [Accessed January 14, 2018]

[e] Rosbag Store. https://rosbag.tier4.jp [Accessed June 1, 2018]

Name:
Alexander Carballo

Affiliation:
Green Mobility Research Institute, Institutes of Innovation for Future Society, Nagoya University

Address:
Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan

Brief Biographical History:
1996-2006 Lecturer, School of Computer Engineering, Costa Rica Institute of Technology
2011 Received Ph.D., Intelligent Robot Laboratory, Tsukuba University
2011-2017 Research and Development, Hokuyo Automatic Co., Ltd.
2016- Part-time Lecturer, Graduate School of Engineering, Osaka City University
2017- Designated Assistant Professor, Nagoya University

Main Works:
• "People Detection using Range and Intensity Data from Multi-Layered Laser Range Finders," IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), pp. 5849-5854, 2010.
• "Reliable People Detection using Range and Intensity Data from Multiple Layers of Laser Range Finders on a Mobile Robot," Int. J. Social Robotics, Vol.3, No.2, pp. 167-186, 2011.
• "High Density Ground Maps using Low Boundary Height Estimation for Autonomous Vehicles," IEEE Int. Conf. on Intelligent Trans. Systems (ITSC), 2018.

Membership in Academic Societies:
• The Institute of Electrical and Electronics Engineers (IEEE)
• The Robotics Society of Japan (RSJ)

Name:
Shunya Seiya

Affiliation:
Department of Intelligent Systems, Graduate School of Informatics, Nagoya University

Address:
Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan

Brief Biographical History:
2013-2016 B.Sc., School of Engineering, Nagoya University
2017- M.Sc., Graduate School of Informatics, Nagoya University
2017- Director, BrainIV Inc., Japan

Main Works:
• "Evaluation of Deep Learning-Based Driving Signal Generation Methods for Vehicle Control," Int. Symposium on Future Active Safety Technology (FAST-zero), Japan, 2017.

Name:
Jacob Lambert

Affiliation:
Department of Intelligent Systems, Graduate School of Informatics, Nagoya University

Address:
Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan

Brief Biographical History:
2014 Received B.Sc., Honours Physics, McGill University
2017 Received M.A.Sc., Space and Terrestrial Autonomous Robotics Systems Laboratory, University of Toronto
2017- Ph.D. Candidate, Graduate School of Informatics, Nagoya University

Main Works:
• "Entropy-based sim(3) calibration of 2D lidars to egomotion sensors," IEEE Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems (MFI), pp. 455-461, 2016.

Membership in Academic Societies:
• The Institute of Electrical and Electronics Engineers (IEEE)

Name:
Hatem Darweesh

Affiliation:
Department of Media Science, Graduate School of Information Science, Nagoya University

Address:
Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan

Brief Biographical History:
1998-2002 B.Sc., Faculty of Computers and Information Sciences, Ain Shams University
2002-2013 Teaching Assistant, Moder Academy in Maadi
2004-2009 M.Sc., Faculty of Computers and Information Sciences, Ain Shams University
2004-2010 Co-Founder, NRG Solutions, Egypt
2013-2016 Robotics Engineer, ZMP Inc., Japan
2016- Ph.D. Student, Graduate School of Information Science, Nagoya University

Main Works:
• "Open Source Integrated Planner for Autonomous Navigation in Highly Dynamic Environments," J. Robotics and Mechatronics, Vol.29, No.4, pp. 668-684, 2017.

Name:
Patiphon Narksri

Affiliation:
Department of Electrical Engineering and Computer Science, Graduate School of Engineering, Nagoya University

Address:
National Innovation Complex (NIC), Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan

Brief Biographical History:
2011-2015 B.Eng. in Mechanical Engineering, Chulalongkorn University
2016- Master Student, Graduate School of Engineering, Nagoya University

Main Works:
• "A Slope-robust Cascaded Ground Segmentation in 3D Point Cloud for Autonomous Vehicles," IEEE Int. Conf. on Intelligent Transportation Systems (ITSC), 2018.

Name:
Luis Yoichi Morales

Affiliation:
Driving Scene Understanding Research Division, Institutes of Innovation for Future Society, Nagoya University

Address:
National Innovation Complex (NIC), Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan

Brief Biographical History:
2006 M.Eng., Intelligent Robot Laboratory, Tsukuba University
2009 Ph.D., Intelligent Robot Laboratory, Tsukuba University
2010-2016 Researcher, ATR Intelligent Robotics and Communications Laboratories
2016- Designated Associate Professor, Nagoya University

Main Works:
• "Autonomous Robot Navigation in Outdoor Cluttered Pedestrian Walkways," J. Field Robotics, Vol.26, No.8, pp. 609-635, 2009.
• "Probabilistic 3D Mapping of Sound-Emitting Structures Based on Acoustic Ray Casting," IEEE Tran. on Robotics, Vol.33, No.2, pp. 333-345, 2017.
• "Passenger Discomfort Map for Autonomous Navigation in a Robotic Wheelchair," J. Robotics and Autonomous Systems, Vol.103, pp. 13-26, 2018.

Membership in Academic Societies:
• The Institute of Electrical and Electronics Engineers (IEEE) Robotics and Automation Society
• The Robotics Society of Japan (RSJ)

Name:
Naoki Akai

Affiliation:
Driving Scene Understanding Research Division, Institutes of Innovation for Future Society, Nagoya University

Address:
National Innovation Complex (NIC), Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan

Brief Biographical History:
2013-2016 Doctor Student, Utsunomiya University
2016- Designated Assistant Professor, Nagoya University

Main Works:
• "Gaussian processes for magnetic map-based localization in large-scale indoor environments," IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), pp. 4459-4464, 2015.
• "Reliability estimation of vehicle localization result," IEEE Intelligent Vehicles Symposium (IV), pp. 740-747, 2018.
• "Mobile robot localization considering class of sensor observations," IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2018.

Membership in Academic Societies:
• The Institute of Electrical and Electronics Engineers (IEEE)
• The Japan Society of Mechanical Engineering (JSME)
• The Robotics Society of Japan (RSJ)
• The Society of Instrumentation and Control Engineering (SICE)

Name:
Eijiro Takeuchi

Affiliation:
Department of Intelligent Systems, Graduate School of Informatics, Nagoya University

Address:
Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan

Brief Biographical History:
2008-2014 Assistant Professor, Tohoku University
2014-2016 Designated Associate Professor, Nagoya University
2016- Associate Professor, Nagoya University

Main Works:
• "A 3-D Scan Matching using Improved 3-D Normal Distributions Transform for Mobile Robotic Mapping," IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), pp. 3068-3073, 2006.

Membership in Academic Societies:
• The Japan Society of Mechanical Engineering (JSME)
• The Robotics Society of Japan (RSJ)
• The Society of Instrument and Control Engineers (SICE)
• The Institute of Electrical and Electronics Engineers (IEEE)

Name:
Kazuya Takeda

Affiliation:
Department of Media Science, Graduate School of Information Science, Nagoya University

Address:
Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan

Brief Biographical History:
1983/1985 Received B.E./M.E. degrees in Electrical Engineering from Nagoya University
1986-1989 Advanced Telecommunication Research Laboratories (ATR)
1987-1988 Visiting Scientist, MIT
1989-1995 Researcher and Research Supervisor, KDD Research and Development Laboratories
1994 Received Doctor of Engineering degree from Nagoya University
1995-2003 Associate Professor, Faculty of Engineering, Nagoya University
2003- Professor, Department of Media Science, Graduate School of Information Science, Nagoya University

Main Works:
• Media signal processing and its applications, which include spatial audio, robust speech recognition and driving behavior modeling

Membership in Academic Societies:
• The Institute of Electrical and Electronics Engineers (IEEE)