International Journal of Computational Intelligence Systems
Vol. **12(2)**, 2019, *pp.* 1197–1211
DOI: https://doi.org/10.2991/ijcis.d.190930.001; ISSN: 1875-6891; eISSN: 1875-6883
https://www.atlantis-press.com/journals/ijcis/

Adaptive Fuzzy Mediation for Multimodal Control of Mobile Robots in Navigation-Based Tasks

Josip Musić[†], Stanko Kružić[*,†], Ivo Stančić, Vladan Papić

University of Split, Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, Ruđera Boškovića 32, Split, Croatia

ARTICLE INFO

Article History
Received 22 Mar 2019
Accepted 17 Sep 2019

Keywords
Fuzzy mediation
Adaptive control
Mobile robot
Neural networks
Teleoperation

ABSTRACT

The paper proposes and analyses performance of a fuzzy-based mediator with showcase examples in robot navigation. The mediator receives outputs from two controllers and uses estimated collision probability for adapting the signal proportions in the final output. The approach was implemented and tested in simulation and on real robots with different footprints. The task complexity during testing varied from single obstacle avoidance to a realistic navigation in real environments. The obtained results showed that this approach is simple but effective.

1. INTRODUCTION

Autonomous mobile robots and vehicles are getting out of structured and laboratory-type environments into more dynamic environments [1–5]. Due to the uncertain nature of these environments, a robot's behavior needs to be adaptive in order to work properly without human intervention or with minimal human intervention. These environments range from air-, water- and ground-based ones to space-based ones [1,4]. The paper focuses on ground-based autonomous mobile robots, but some of its conclusions could be extended to other domains (like air and water). Thus, the term *autonomous mobile robots* will refer to ground-based mobile platforms.

When considering the application of autonomous mobile robots, usually two main concerns are of interest: long-term autonomy and safety [1], especially in light of human–robot cooperation. This has been highlighted by the fact that several recent EU-, USA-, and Japan-based projects consider safety as the main challenge in human–robot interaction [2]. In order to alleviate these concerns, researchers usually provide a robot with an appropriate form of artificial intelligence. The current state-of-the-art approaches use some form of Deep Neural Network (DNN) [6] in conjunction with a fusion of data from several sources which is referred to as deep multimodal learning [3]. While deep multimodal learning approach has found many interesting applications within recognition and classification fields (e.g., Ref. [7]), applications to autonomous robot navigation are scarce (but do exist, e.g., Refs. [8–10]). One notable exception is semi-autonomous cars [3,11,12], which also have well-known and large databases available for testing and benchmarking new approaches [13,14]. The fusion itself can be achieved on three levels [3,15]: feature level (sometimes referred to as *early fusion* or *pre-mapping fusion*), decision level (sometimes referred to as *late fusion* or *post-mapping fusion*), and mid-level (sometimes referred to as *intermediate fusion* or *midst-mapping fusion*). Some variations exist within each of these high-level fusion architectures (e.g., Ref. [16]). Regardless of which fusion scheme is used "*most multimodal deep-learning architectures proposed to date are meticulously handcrafted*" [3]. Even a minor modification to the neural network (NN) task usually require re-training the whole network or training it from scratch, which is a time-intensive procedure (especially in multimodal networks). This is due to the fact that (D)NNs are designed to do one specific task for which they require large amounts of training data (usually images/videos) [17]. Additionally, one would like to have an approach which enables combining (D)NNs with other control algorithms already present in robotics (like Proportional-Integral-Derivative [PID] controller) as well as integrating human input into control scheme (like in teleoperation scenarios).

In the field of control theory there have been approaches where more complex behavior is achieved through mediation between expert and novice controllers [18]. The ultimate goal of the mediation is to "*help the autonomous capabilities of an individual . . . and to help individuals with different conflicting interests or diverging opinions to mediate situations and reach consensus*" [19]. Thus, more complex tasks (such as robot navigation) could be split into several simpler ones (which we refer to as task primitives), and each can be addressed by a different controller including (but not limited to)

[*] *Corresponding author. Email: skruzic@fesb.hr*
[†] Authors contributed equally

(D)NNs. By using the proposed control architecture the modifications in task primitives (e.g., adding another task primitive) should require less time-consuming interventions. Also, such modularity would ensure easier upgrade procedure as well as the implementation of more specialized (and better performing) controllers for the particular task (which could be done even on the fly). To the best of our knowledge, this is the first time to date such an adaptive fuzzy-based controller, which can include NNs, is used in mobile robot navigation tasks and its performance demonstrated in real-world scenarios.

1.1. Related Work

Since state-of-the-art approaches use predominantly NNs to tackle the intelligent robot control, we will focus on them in the review.

In Ref. [20] authors implemented end-to-end motion planning for mobile robots in which they trained (in a supervised manner) a Convolutional Neural Network (CNN) using simulated sensor data gathered in *Stage 2D* environment. The term *end-to-end control* refers to the fact that the input data was directly mapped to the (translational and rotational) steering commands. The training data was recorded while navigating in a simulation on a known map using Robot Operating System (ROS) [21] *2D Navigation stack* with its local (Dynamic Window Approach [DWA]) and global (Dijkstra) planners. However, when testing was done (both in a simulation environment and in real-world using Kobuki-based Turtlebot), no map was given to the robot but only the relative distance to the target. Thus, in essence, the CNN was applying navigation strategies learnt during training. The approach showed promising results, being able to navigate to the desired location in the real-world without the need for a map while avoiding moving obstacles (although testing was limited). However, the authors outlined several situations where the control algorithm needed a bit of help to get "*unstuck*": large open spaces and convex dead-end regions. The incidence rate of human assistance dropped if additional fully connected (FC) layers were added (this was not true for simulation-based environments). It should be noted that work from Ref. [20] has some similarities with our own work: using 2D Light Detection and Ranging (LiDAR) data instead of a video stream for NN training, using simulation as a source of data for self-supervised learning and navigating without the need for a map. However, our approach achieves those goals in a different manner within the framework which we believe to be more flexible and which can integrate additional, standard, control algorithms. We also use both positive and negative examples for training. Successful obstacle avoidance is regarded as a positive example, while the crash is considered a negative example in the process.

Similar works that used some form of NN to learn end-to-end policies can be found in Refs. [22,23]. In Ref. [22] it was examined if end-to-end training of both the visual and the control system provides better performance than in the case when they were trained separately. These policies were represented by deep CNNs with supervised learning approach via guided policy search. While the paper mainly deals with the application of visuomotor polices on robot manipulator tasks (e.g., screwing a cap onto a bottle, placing a coat hanger on a rack), some of its conclusions can be transferred to a mobile robot case. One such conclusion is the fact that the end-to-end approach outperformed approach in which vision layer was trained separately. Although some drawbacks associated with generalization and visual distractors were noted, authors suggest straightforward ways of dealing with them. In Ref. [23] an end-to-end approach was also used with steering angles obtained from raw input images. This was achieved using CNN with six layers, trained in a supervised manner using data obtained from a human operator in real-world in various weather conditions and terrains. The training set consisted of 95, 000 image pairs (since two cameras were used). Authors reported good navigation and obstacle avoidance capabilities even at speeds of 2 m/s and noted that the proposed approach eliminates the need for any form of calibration and parameter tuning as well as a need to select features from raw images. It is worth noting that end-to-end control policies are also finding their way into the realm of autonomous cars [24] where additional, domain-specific problems exist, which can also be addressed by DNNs and/or recurrent neural networks (RNN), like negotiating safe driving [25] or anticipating (and correcting) human actions [12].

Additional works in the area of autonomous navigation include Refs. [8,11,25–27]. In Ref. [26] authors did not use end-to-end approach but developed a long-range vision system based on DNNs which interacted with path planning and obstacle avoidance algorithms in order to generate steering commands. The long-range vision module was trained in real-time in a self-supervised manner with stereo supervisor taking care that data and label are not changed. The supervisor involved a number of algorithms (like ground plane estimation, and statistical false obstacle filtering) which interacted in a complex manner. The long-range classifier demonstrated obstacle detection capabilities from 5 to over 100 meters. The approach was tested in two separate scenarios. First, the classification network was tested as a standalone system where different network architectures were applied. It was shown that hybrid unsupervised/supervised approach performs the best (with error rates of about 8%). After that, the NN was tested as a part of the navigation system. It ran at 1–2 Hz enabling strategic navigation from 5 m to the goal (also a short-range algorithm run in parallel at 8–10 Hz in order to avoid obstacles). The testing was done on several different courses where long-range module proved useful and enabled error (and intervention) free navigation. The improvement was also evident in total time and distance taken to reach the goal.

In Ref. [27] visual-based system (from a first-person view) was also used, but now for an indoor environment and in an end-to-end manner. Here, a complex structure of several NNs was used in order to map the space and navigate through it. Used NNs were of convolutional type (ResNet-50 pretrained on ImageNet). The process is a reminiscence of Simultaneous Localisation and Mapping, but authors note that their approach differs in several ways from classical approaches one being that it can learn statistical regularities of the environment. Thus, the approach enables tracking of visited parts of the environments as well as semantically driven navigation (e.g., "*go to a table*"). The proposed approach was developed in the simulated environment, based on Stanford large-scale 3D Indoor Spaces (S3DIS) data-set which was obtained by 3D scanning of six large-scale indoor spaces within three different buildings. Obtained results demonstrated that the proposed approach outperforms classical approaches such as reactive strategy and standard memory-based architectures (e.g., Long-Short Term Memory).

Here, it is interesting to note the use of simulation-based data for training of CNNs (or other types of NNs). Due to a large amount

of data needed to properly train different NN architectures, collecting them in real-world is a time-consuming task and one in which certain level of danger to environment and robot itself cannot be avoided [28]. The simulation makes it easy to run extensive and elaborate experiments while providing necessary data in a safe and controlled manner. While in some works like Refs. [11,27,29] the training and the testing were achieved in the simulation environment, this is not optimal since a gap between simulation and reality exists [30]. However, recently a number of works have emerged where testing of simulation trained CNNs was done in real environments with no or little re-training [8,31]. This is mainly due to recent advances in the field of computer graphics which makes possible a rich and vivid representation of real-world environments as well as their physics, thus narrowing the beforementioned gap. However, another more recent approach is to use a combination of photo-realistic data in conjunction with nonphoto-realistic domain randomized data to leverage the strengths of both [32]. This approach was then used to learn the object's 3D pose from RGB images, outperforming state-of-the-art NNs trained on real images.

In Ref. [8] authors used deep siamese actor-critic model within deep reinforcement learning (RL) framework for target-driven visual navigation. For the approach model was given an RGB image of the current scene and additional RGB image of the target, providing as output required action: moving forward, moving backwards, turning left, and turning right. Authors highlight that in this manner their RL approach becomes more flexible to changes in task goals (in contrast to more standard approaches like Ref. [33] in which the goal is hardcoded in the NN). In order to achieve this, the authors developed custom 3D simulation framework named AI2-THOR (The House Of inteRaction) by providing reference images to artists which then created realistic 3D environments (32 scenes in total). The approach was tested both in simulation and in the real environment with SCITOS mobile robot. Obtained results from simulation showed shorter average trajectory length for the proposed approach when compared to baseline methods, while results from real-world experiment demonstrated that network trained in simulation and fine-tuned on real data converged faster than network trained from scratch. Authors especially emphasize that their approach is generalizable to new scenes and new targets.

In Ref. [31] authors went a bit further, developing a NN model solely in simulation and applying it in the real-world without any modification. It was used for searching for control policy which can process raw data (e.g., images) and output control signals (velocity), and which relied on RL. Since, as authors outline, RL needs at least some collisions (negative examples) during training it can be a challenging and risky undertaking to train a mobile robot on real-world data. To avoid real-world training authors used Blender, an open-source 3D modelling suite, to construct a number of hallway-based environments (24 using 12 different floor-plans) for generating data used for training deep fully CNN with dilation operations, built on VGG16 architecture. It is interesting to note that authors intentionally rendered environments with a lower degree of realism and/or visual fidelity and ones with randomized textures, lighting conditions, and obstacle placement and orientation. This was based on the hypothesis that generating a large number of such environments with numerous variations (of textures which there were 200) would produce a model which generalizes well. This hypothesis was verified in experiments both in synthetic and real environments. The simulation-based testing showed that the proposed RL approach outperforms used baseline methods (Straight controller and Left, Right and Straight [LRS] controller), but in real-world testing experience some collisions. This led the authors to conclude that the proposed approach is viable and that further testing is needed such as the inclusion of additional sensors (like RGB-D cameras). A similar approach that employs RL-based approach for control policy search can also be found in Ref. [34].

From the literature review, it can be concluded that deploying state-of-the-art autonomous capabilities to a mobile robot usually entails developing complex NN-based architectures and training them on large data-sets (either in a simulation or in a real-world environment). If a different behavior is required, a new network needs to be developed or existing one retrained (and new training data generated/recorded). In order to avoid this, we propose the implementation of a mediator which would enable fusion of control signals from several sources (including, but not limited to, NNs). One possible function on which adaptive control in mobile robot navigation tasks could be based is collision probability (CP) [35] since obstacle avoidance usually has the highest priority in the navigation stack hierarchy. The mediation can be implemented in several forms, and one of them is based on fuzzy set theory [36]. It is worth noting that several *mediation engines* can be used in the process, as well as several decision functions based on which the mediator infers the final decision and thus adapts the robot's behavior [18].

Finally, it is worth noting that the implementation of ensembles of NNs is not a new idea. In Ref. [37] they were used in conjunction with a plurality consensus scheme for classification applications, with focus on optimizing NN parameters and architectures. The approach was tested in two (simple) scenarios: generalized exclusive disjunction (XOR) operator and region classification in the 20-dimensional hypercube. While the obtained results were an improvement over single copy networks, they are applicable only to NNs (i.e., no other algorithm can be introduced into the approach). There have also been approaches that utilize a single DNN for object detection, decision-making and inferring control commands [38], inspired by the way a human brain works. While this work focuses on robotic applications (using RGB-D camera as main sensors) and presents promising real-world results for indoor robot exploration tasks (including obstacle avoidance), it has two potential drawbacks: as Ref. [38] it is intended only for NN-based strategies, and since it uses CNNs it requires large amounts of data for training (which were recorded with human driver exploring an environment). Additionally, the use of fuzzy logic variants (e.g., fuzzy integral) for fusing outputs from several NNs has also been proposed before in Ref. [39] (for dynamic nonlinear process modelling) and Ref. [17] (for robust character classification purposes). However, in Ref. [17] nonadaptive fusion mechanism was used (i.e., the contribution of each network was known in advance based on its loss function). This is not suitable for intended application where, due to the dynamic nature of the environment, different controllers can have variable importance. Also, the approach (as is the case in Ref. [39]) does not consider a way to include other types of controllers (than NNs) or human input. Some work has been done on adapting the fusion scheme, like in Ref. [40] but without the use of fuzzy-based logic, using gating network that makes a stochastic one-out-of-n decision about which single network to use

for a particular instance. The approach was tested in a multispeaker vowel recognition task. Also, as can be seen from the above discussion, in many such examples primarily intended application is not related to the field of robotics.

1.2. Contributions

The goal of the research was to develop a procedure to combine together outputs from two control algorithms (including NNs), each performing different task primitive in robot navigation. Fuzzy logic based mediation was chosen as a good framework since it: (a) enables combining individual controller outputs in a complex signal. In the process, different kernel functions (*mediation engines*) can be used to obtain different data fusion results., (b) its functionality is transparent and can be easily understood by an average user. Additionally, it has low computational complexity, making real-time implementation possible, and (c) depending on the global task, the decision (adaptation) variable can be freely chosen while maintaining generality of the framework and ensuring different behavior profiles.

Our contributions are thus as follows:

- To the best of our knowledge, this is the first real-world application of fuzzy-based mediator for robot navigation-based tasks to date.
- Development of NN-based obstacle avoidance and navigation entirely in simulation using LiDAR measurements (and not camera data, as is often done in the literature) and application of it in real-world navigation tasks without any modification (retraining) with good results. The approach provides both positive and negative examples and is self-contained since minimal human intervention is needed (e.g., specifying simulation environment parameters) during training data gathering and subsequent NN training.
- Comparison of performance in different tasks and on different robots (with different footprints) which goes to the generality of the proposed approach.
- Implementation of the mediation scheme in the teleoperation control for fusion between the obstacle avoidance controller and human operator. In this manner, a seamless control handover between the human operator and autonomous part of the robot is achieved ensuring safe movement of the robot in the environment where the human operator does not necessarily always have knowledge of all obstacles in the space (i.e., lower spatial awareness).

The remainder of the article is structured as follows. In Section 2, main operational principles of our approach are presented and discussed, while in Section 3 experimental setups for all test cases are presented with explanations for made design decisions and used algorithms. Section 4 presents achieved results using previously described experimental setups, as well as analyses and discusses the results. It also provides a comparison with a standard approach in the robotics. Finally, Section 5 concludes the article with the highlight of the main research findings, as well as proposed possible improvements and future research directions.

2. THE PROPOSED ADAPTIVE FUZZY MEDIATOR FOR MOBILE ROBOTICS

Overview of the proposed adaptive fuzzy control scheme can be seen in Figure 1 which shows that the approach uses two controllers (NNs or any other type of controller) whose inputs are combined together within fuzzy set theory based on CP (p_{COL}) as the decision variable. The CP parameter is explained in more detail later on.

In the proposed approach, one controller was used for reaching the goal position without regard to obstacles (navigation controller), while the other controller was used solely for obstacle avoidance without regard to the goal position. Both controllers (in the case of NN) were trained using data obtained in simulation, which made the proposed approach simple and safe.

NN-based obstacle avoidance consisted of three sub-NNs (left, right and forward). In order to train them, a simulation environment was created with a number of obstacles of various shapes and sizes scattered randomly within it. Then, a robot was spawned in its center with random orientation and was given a constant linear velocity and kept moving until hitting an obstacle (thus providing both positive and negative samples). A similar approach was used in Ref. [28]. While in motion, LiDAR collected data of nearby obstacles (x_O, y_O). The procedure was repeated 4,000 times with 20 different obstacles configurations, giving a total of 396, 377 training examples (which includes both the positive and the negative examples). The sub-NN for forward motion used scans of 45° to both left and right of forward direction (90° in total), while sub-NNs for left and right motion used scans of 90° to the left and to the right of forward direction, respectively. Each of the three sub-NNs was designed as a simple feed-forward network with 1 hidden layer and 60 neurons. It should be noted that this architecture was chosen experimentally as the simplest one with satisfactory performance (according to the mean square error parameter). Based on outputs from these three sub-NNs, logic for controlling the robot was devised (i.e., obstacle avoidance controller). The output from this controller was angular (ω_A) velocity needed to avoid the obstacle (while linear velocity v_A could obtain one of two pre-defined values as will be described later on). The equation used for calculation of angular velocity was defined as

$$\omega_A = \begin{cases} 2 \cdot NN_{left}, NN_{left} \geq NN_{right} \\ -2 \cdot NN_{right}, NN_{right} < NN_{left} \\ NN_{left}, NN_{right} \in [0, 1]. \end{cases} \quad (1)$$

Equation (1) was implemented only if NN_{front} output was smaller than a pre-defined threshold (0.6 in our case). Otherwise, the robot would not rotate. The approach was tested both in simulation and in real-world with good results. It should be noted that the value

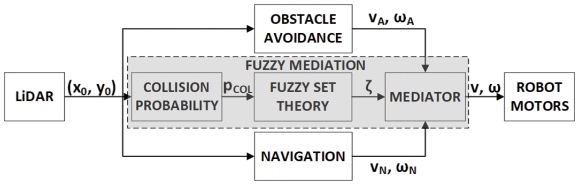


Figure 1 | The general control architecture of the proposed approach.

of threshold (0.6 or 60%) was determined experimentally (both in simulation and in real-world) and presents the belief of the NN (in percentages) that there are no obstacle in front of the robot. The value is thus dependent on the used NN structure and training procedure, as well as the sensor used (but not the robot itself) and might change if different training/sensor setup is used.

In the case of the development of NN for navigation, the following procedure was taken. First, a single user was asked to drive a Turtlebot 2 mobile robot in Gazebo simulation from a random starting point to a random goal within a 25x25 m environment without any obstacles. The user was told to give equal importance to the accuracy and the speed of driving. During the driving the robot position was recorded, thus making it possible to calculate the distance from the goal. This procedure was repeated 700 times, giving a total of 380, 541 training examples. After that, the recorded data was used to train simple NN (for regression) with 4 layers of 30 neuron each. This architecture was (as was the case for NN-based avoidance) determined experimentally as the one with good (enough) performance. As an input, the network was given the error in position (i.e., distance to the goal) and error in orientation (i.e., the difference between the current and the final orientation). As the output, the network was given both the linear velocity and angular velocity recorded during the trials (and this was its output during testing). After the training, the approach was tested with success in environments without obstacles, both in simulation and in real life.

Additionally, when a simple Proportional (P)-type controller was used, its output was such that it kept the linear velocity (v_N) constant till the goal was reached, while angular velocity (ω_N) was changed depending on the difference between current and desired position (both in the odometer frame of reference). In this manner, both the simple and more complex (NN) cases of navigation controller were taken into account.

The outputs from NN-based obstacle controller and from navigation controller (being either NN or P-type) were then fed to the fuzzy mediation block, where the consensus decision was made (based on ζ value) and adapted linear (v) and angular (ω) velocities generated. The block firstly computes CP (p_{COL}) based on LiDAR measurements and simplified robot kinematic model. The CP was designed in such a way to be reliable and computationally effective while providing the mediation algorithm with necessary information. Although similar variables do exist in the literature, they are not as simple and computationally effective [35] as the proposed one.

CP was calculated in several simple steps as described hereon. First, an estimated trajectory of the robot for the next 20 samples (about 2 seconds) was computed based on a simple kinematic model of motion. The interval of 20 samples was chosen through experimentation as the one that, taking into account planned robot velocity (0.2 m/s) and the distance at which the NN obstacle avoidance reacts, provides sufficient time horizon for reaction. During system development, different interval lengths were tested: longer ones produced trajectory estimates that were less likely to occur resulting in robot giving control to obstacle avoidance when in reality it did not need to, while shorter intervals produced much more likely trajectory estimates but reduced significantly reaction time for the robot resulting in a jittery motion and number of obstacle crashes. Thus, we concluded that for selected mediator parameters (and distance for obstacle avoidance for which NN was trained) the interval length is primarily a function of robot's linear velocity (but also other parameters to a lesser extent). This perceived dependency should, however, be revisited if in-place rotation is introduced. Thus, if different robot speed, sampling time and/or allowed distance to the obstacles during NN-based training (1 m in our case) was used, the interval value should be adjusted accordingly. The well-known simple kinematic model used in the study was defined in matrix form as:

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \\ \Phi_{i+1} \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ \Phi_i \end{bmatrix} + \begin{bmatrix} -v * \sin \Phi_i \\ v * \cos \Phi_i \\ \omega \end{bmatrix} \Delta T \tag{2}$$

where x_i, y_i, and Φ_i define robot's pose at time instance i, and ΔT is sampling period between time instances i and $i + 1$. During this projection in time, linear velocity was kept constant at a value measured at the time of calculation, while angular velocity was propagated through time in an exponentially decaying manner using formula $\omega_i = 0.85\omega_{i-1}$ where $i = 1, 2, 3, 4, ..., 20$. The main idea behind this was the fact that angular velocity usually has short term changes in order to adjust the robot's heading, and projecting just the current angular velocity value (like is done for linear velocity) would give unrealistically curved fast changing trajectories that (for most part) do not correspond to actual robot motion. Thus, the proposed decay term filters out these sudden changes and gives a more stable trajectory estimate. However, it should be noted that there are some instances (like robot turning for longer periods of time or in-place turning) where it is sub-optimal. To account for such and similar uncertainties associated with assumptions/errors in estimation/localisation algorithms (as well as any sensor and/or actuator errors), uncertainty ellipses were introduced in the CP calculation.

After the projected trajectory has been calculated, an uncertainty ellipse is constructed for each point on the trajectory. The ellipses have increasing axes lengths with each ahead sample in order to account for motion uncertainty. While this always positive increase might be considered too cautious in some instances (e.g., driving parallel with an obstacle/wall), we believe it is a prudent step since, in general, such situations where robots drive perfectly parallel with an obstacle are not too common. Also, with the appropriate selection of the initial ellipse dimensions as well as their increments (in respect to the distance used in NN obstacle avoidance training) this should not pose a serious limitation on robot performance (i.e., robot trajectory might be slightly longer, but the final objective would still be achieved). Ellipses' initial axes lengths and increments were determined experimentally but may change if the situation warrants it. For example, during the testing with Turtlebot 2 mobile robot, the following ellipse parameters values were used: $a_{init} = 0.3$ m, $b_{init} = 0.1$ m, $a_{incr} = 0.01$ m, and $b_{incr} = 0.005$ m (note that the a axis is perpendicular to the current robot orientation). This in turn (after 2 sec ahead in time projection, as explained later on) gives an ellipse with maximum dimensions of $a_{max} = 0.5$ m and $b_{max} = 0.2$ m (keep in mind that the robot base dimensions are of 0.36 m diameter). One possible improvement is to adaptively (on-line) adjust these parameters. However, it should be noted that although in the work two mobile robot platforms with very different footprints (both in size and in shape) have been used (please see Section 3 for more

details), only a limited number of parameters related to mediation were adjusted. Also, LiDAR readings were adjusted in order to better accommodated for LiDAR placement on the robot, as will be explained in more detail in Section 3. An example of an estimated robot trajectory including uncertainty ellipses is shown in Figure 2. Please note that in the figure, for clarity reasons, not all ellipses are included. It should be noted that the time horizon was incorporated into the approach at two places (for different reasons): once through increasing uncertainty ellipses dimensions, and once during CP calculation (as will be explained in a more detail below). However this was done in a different manner. For ellipses it was more of a binary type decision (i.e., thresholding for improved computational efficiency), while for CP the points that passed the time horizon threshold were used for a more precise (fine grained) calculation of their contribution to final CP.

Once estimated trajectory and uncertainty ellipses are constructed, in the next step, each ellipse is examined in order to find if any (parts) of obstacles (i.e., LiDAR points) are located within it. If they are, the ellipse is considered further or is otherwise discarded. Then, for each remaining ellipse, a CP is estimated using a sigmoid function with variable parameters (depending on distance to obstacle and distance in estimation time horizon). An example of sigmoid functions is included in Figure 2.

The general sigmoid function that was used in the work for calculation of CP for time instance i and in relation to obstacle j is of a form:

$$p_{COL_{(i,j)}} = \frac{1}{1 + e^{-a(i-c)}} \tag{3}$$

where a is a constant representing ellipse longer axis (and the one perpendicular to current motion direction), $i \in [1, 20]$ is the time instance for which the sigmoid/CP is calculated, and $c = \frac{1}{\sqrt{\left(x_O - x_{i,R}\right)^2 + \left(y_O - y_{i,R}\right)^2}}$ is the distance between the j-th obstacle and estimated robot pose at time instance i. Please note the following: 1) The maximum value a sigmoid function (i.e., CP) can obtain is 1.0 (i.e., 100%). The function value is plotted/calculated for 20 time instance interval, but only the value at i-th instance is taken into account for CP_i calculation., 2) Parameter a has a negative sign making the function open to the left. This parameter also defines the slope of the sigmoid (higher values give a steeper slope and vice versa)., 3) The term in round brackets $(i - c)$ for a constant a defines the rate of convergence of end parts of the function (e.g., see dotted and full line plots in Figure 2). In this manner, the shape of the sigmoid function (and consequently the value of the CP) is adaptively changed to accommodate longer time horizon in estimation (time variable) and proximity to obstacles (distance variable). This adaptive change in sigmoid function is depicted in Figure 2 alongside with related ellipses and robot's trajectory., and 4) In this manner, a maximum of $20 * j$ CPs are calculated. If no CP is calculated (i.e., there are no obstacles) its value is considered to be 0.

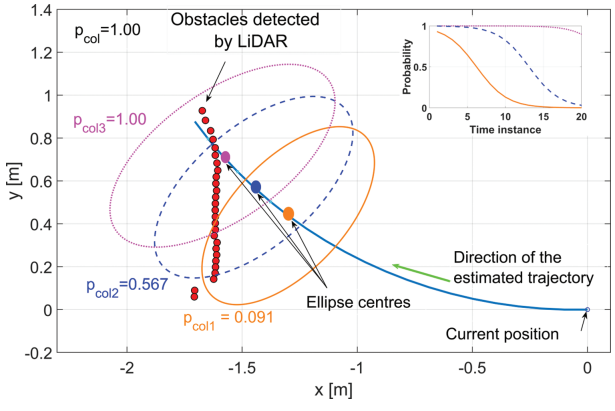


Figure 2 An example of collision probability (CP) calculation based on Light Detection and Ranging (LiDAR) data.

Finally, when all CP values are calculated, the highest CP is kept for that time instance. It is then used for the mediator block as the decision variable (p_{COL}) in the fuzzy set theory. Membership function was constructed in the manner depicted in Figure 3 and included five possible memberships: *no avoidance* (NA), *light avoidance* (LA), *balanced avoidance* (BA), *strong avoidance* (SA), and *full avoidance* (FA). Thus, based on calculated CP for time instance i ($p_{COL,i}$), probabilities of belonging to each of the membership sets are calculated.

Each of previously mentioned sets has its own control shift factor associated with it, which is then used to compute final output: 0, 0.25, 0.5, 0.75, and 1, respectively. This implies that the used mediation had a linear *mediation engine*, but other mediation engines (like exponential ones) can be used as is pointed out in Ref. [18]. Thus, the shift percentage is computed as

$$\begin{aligned} SP_i = SP_{NA} * \mu_{NA,i} + SP_{LA} * \mu_{LA,i} + SP_{BA} * \mu_{BA,i} + \\ + SP_{SA} * \mu_{SA,i} + SP_{FA} * \mu_{FA,i} \end{aligned} \tag{4}$$

where μ_i is set membership function for the respective group at time instance i, and SP_i is a shift percentage for the particular set. Then the value of calculated SP_i is used as the threshold in equation

$$\zeta_i = \begin{cases} \zeta_{i-1} + 0.35, & SP_i \geq 0.2 \\ \zeta_{i-1} - 0.15, & SP_i < 0.2 \end{cases} \\ \zeta \in [0, 1] \tag{5}$$

where ζ_i is mediation coefficient used for calculation of mediated linear and angular velocities at time instance i, using equation

$$\begin{aligned} v_i &= v_{A,i}\zeta + v_{N,i}\left(1 - \zeta\right) \\ \omega_i &= \omega_{A,i}\zeta + \omega_{N,i}\left(1 - \zeta\right). \end{aligned} \tag{6}$$

It was observed during the experimentation that coefficients used in Equation (5) dominantly depend on robots maximal allowed linear and angular velocities (and not the robot itself, although robot footprint was taken into account with LiDAR adjustments as described

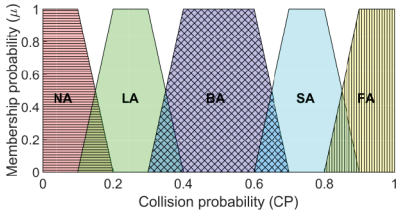


Figure 3 The fuzzy set membership function used in the calculation of mediation coefficient ζ.

previously). As a side note, please note that in Equation (5), calculated CP value could potentially be directly used for thresholding (circumventing fuzzy set approach), but in practical testing this proved to be too sensitive to estimated trajectory changes (uncertainties). Additionally, since CP does not need to be used as a decision variable (e.g., when a different task is considered) removing fuzzy set would reduce generality of the proposed approach, as well as possibility of using nonlinear control shift functions.

In Equation (6) v_i is mediated linear velocity at time i, ω_i is mediated angular velocity at time i, $v_{A,i}$, and $v_{N,i}$ are linear velocities coming from obstacle avoidance and navigation controllers, respectively, $\omega_{A,i}$ and $\omega_{N,i}$ are angular velocities coming from obstacle avoidance and navigation controller, respectively (all for time instance i). The values of ζ_i were then low-pass filtered in order to avoid sudden changes. It should be noted that in the current experiments calculated linear velocity v_i was not directly sent to the motor controller but was rather used for input to simple non-linearity which produced two values at its output: slow (0.1 m/s) or fast (0.2 m/s) velocity. In Equation (5) increase and decrease coefficients were determined experimentally and were chosen such that the mediation swings towards obstacle avoidance more quickly than it returns to navigation since obstacle avoidance had higher priority.

Here we demonstrate the whole calculation procedure on a simple example. Lets assume that the robot is going toward the goal point with linear velocity of $v_i = 0.2$ m/s and with angular velocity of $\omega_i = 0.8$ rad/s. At certain point in time (i), the robot starts to get close to an obstacle, thus the CP parameter starts to change and has a value of $p_{COL,i} = 0.65$. Based on Figure 3 the fuzzy set membership functions would be: $\mu_{NA,i} = 0$, $\mu_{LA,i} = 0$, $\mu_{BA,i} = 0.5$, $\mu_{SA,i} = 0.5$, and $\mu_{OA,i} = 0$, thus the shift percentage SP_i (Equation (5)) would be 0.625. This would in turn result in increase of mediation coefficient ζ_{i-1} from 0 (i.e., the robot was only navigating and did not sense any obstacle in its way) to a new value of $\zeta_i = 0.35$. If we assume that the outputs from navigation controller are $v_{N,i} = 0.2$ m/s and $\omega_{N,i} = 0$ rad/s, and from obstacle avoidance $v_{A,i} = 0.2$ m/s and $\omega_{A,i} = -2$ rad/s, then based on Equation (6) the mediated outputs would be $v_i = 0.2$ m/s (not taking into account the above mentioned threshold) and $\omega_i = -0.18$ rad/s. From the example the main idea of the work can be seen. Namely, since the CP is detected (based on current linear and angular velocity which in this case are 0.2 m/s and 0.8 rad/s), the mediation mechanism now takes into account the obstacle avoidance controller and shifts the control values in its favor (for angular velocity from 0.8 rad/s to –0.18 rad/s). If the CP still persisted then this move would in the next time instance $(i + 1)$ be even more in the favor of obstacle avoidance algorithm, but if not it would slowly return control to the navigation controller.

3. EXPERIMENTAL SETUP

The experimental verification and testing of the proposed approach consisted of three main parts: simulation-based one (Subsection 3.1), real-world-based one (Subsection 3.2), and application-specific one (Subsection 3.3). The main idea behind each of them was as follows. The goal of simulation experiment was to initially verify the functioning and viability of the proposed approach, while the goal of the real-world experiment was to build-upon simulation experiment and demonstrate that the approach is valid on real robots with different footprints and in different (realistic) situations. Finally, the application-specific testing had the goal of demonstrating that the proposed approach could be of practical use for solving or at least minimizing some of the real-world problems (teleoperation was chosen as one such problem). In all three cases, NNs used for the obstacle avoidance (as described in more detail in Section 2) were trained and deployed using TensorFlow open source machine learning framework (on top of *Keras*). Same machine learning framework was used for training and deploying NN navigation controller. These networks, along with other controller(s), mediation and all other algorithms were deployed in ROS environment which enabled us (in some cases) to run the system in a distributed manner (i.e., NN was running on a separate computer due to hardware requirements related to *TensorFlow*).

3.1. Simulation

Simulation experiments were carried out in Gazebo 7 simulation environment. A random environment was generated during this test, with randomly distributed obstacles of different shapes (cylinder of 0.5 m diameter and cube with a side length of 0.5 m). An example of the simulation environment used for testing is depicted in Figure 4. Please note that the overlap of obstacles was permitted.

The random environment generated during the testing had a fixed size (defined by a perimeter wall) of 25x25 m, and a fixed number of obstacles within it (24, out of which 12 were cylinders and 12 were cubes). This number of obstacles was chosen since it provided a sufficiently challenging environment (in our view), while not being too cluttered (please keep in mind the focus of the research was the fuzzy mediation algorithm and not the obstacle avoidance). Fifteen runs were executed in the environment for each navigation/obstacle avoidance controller pair giving in total 30 simulation runs. In each run, the robot started from the same position (center of the environment) and with the same orientation (looking down in Figure 4), and had to navigate to a random point without the map. No time limit was given, and trajectory and outcome (success/crash) were recorded. Results of this experiment are reported in Section 4.1.

3.2. Simple Real-World Scenarios

Real-world experiments were carried in the indoor environment at Faculty of Electrical Engineering, Mechanical Engineering, and Naval Architecture, University of Split. During the experiments, two mobile robot platforms with very different footprints and characteristics were used: customized Turtlebot 2 depicted in Figure 5,

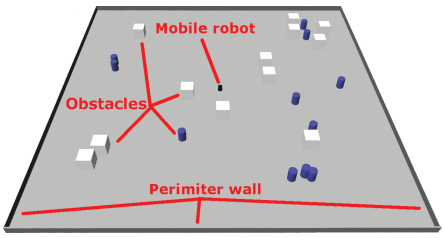


Figure 4 Example of Gazebo simulation environment used for initial testing of the proposed approach.

and a custom-built mobile platform intended to be used as an automated pallet carrier depicted in Figure 6. Please note that in the case of a custom-built robot raw LiDAR data was not fed to the NN obstacle avoidance but rather an adjusted one. The adjustment was achieved by simply adding/subtracting from the measured distance (in an on-line manner) which took into account robot footprint and sensor placement within it, so to be in line with the one used for NN training. If training was achieved with a different/actual footprint this step would not be needed, but nevertheless, it shows how the approach can be expanded to work with different robots.

The main idea behind the use of such different mobile bases was to demonstrate the effectiveness and generalization of our mediation approach. This was important, since a number of parameters are needed for the approach (as described in Section 2), and the question of generality (i.e., the need for fine-tuning them) naturally arises. As will be discussed later on in Section 4.2 it was found it is sufficient to fine-tune only two parameters (related to uncertainty ellipses) to ensure the desired behavior of the mobile robot. Fine-tuning other parameters could possibly improve the performance, but optimisation was not in the research focus.

Next, two main test scenarios were devised for each of the mobile bases. First, for the Turtlebot 2, the experiments consisted of three distinct navigation and obstacle avoidance tasks. In all three cases,

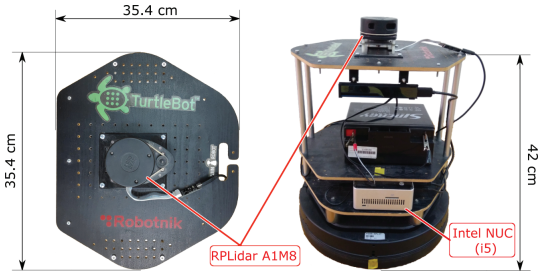


Figure 5 | Overview of customized Turtlebot 2 mobile robot platform. Please note that the robot (with the battery) weights around 10 kg.

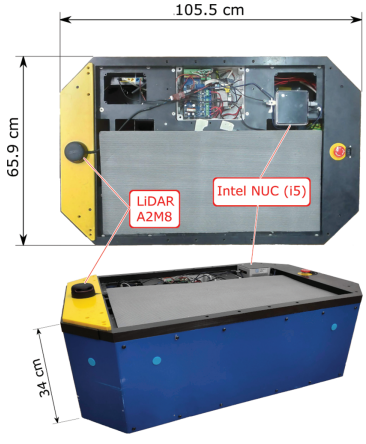


Figure 6 | Overview of a custom-built robot platform (the robot was built by our partner Statim d.o.o.). Please note that the robot has a weight of about 100 kg.

three setups were used: proposed approach with NN navigation and with P controller navigation (both with NN-based obstacle avoidance), as well as default ROS navigation stack with DWA as a baseline method. First two tasks were conducted in an environment augmented with Z- and U-shaped (convex dead-end) obstacles. Five instances of these tasks were run per each setup (15 in total). The last task was navigation in a large real-world environment with two distinct navigation goals—2 examples) and it was run once per each setup (6 runs in total). The real-world environment was the fourth floor at the Faculty building. For the custom-built robot, exactly the same test cases (namely Z- and U- shaped tests) could not be used due to its larger size. Thus the testing was slightly modified. Firstly, a test course was constructed (as depicted in Figure 12a) within which the robot had to reach (from the same starting position and orientation) three different goal points (final orientation was not important). Since three setups were used (NN-based navigation and P-type controller—both with NN-based obstacle avoidance, as well as ROS navigation stack), 9 test runs were recorded. The distance between the starting point and goal point ranged from about 10 m to about 16 m (measured in a straight line). Next, the same larger environment for real-life testing that was used for Turtlebot 2 was also used here (with the same goal points, with the only exception that the goal circle was enlarged so that the robot could more easily meet termination condition). During this testing we also introduced simplified path planning and line following algorithm (as a fourth test case) so that three additional way-points could be given to the algorithm and they were connected with straight lines (running through walls) as can be seen in Figure 11 (X-type marker in lower right corner of the figure and associated dashed straight line). This test was introduced to test if the approach could be integrated with a simplified path planning algorithm within an arbitrary navigation approach and to test if the addition of way-points could improve the performance of mediated navigation. Results obtained for all abovementioned cases are presented in Section 4.2.

3.3. Teleoperation Scenario

The purpose of the final test was to demonstrate the practical applicability of the proposed approach. Thus a simple teleoperation scenario was envisaged in which users had to avoid a single obstacle and reach a goal point. Teleoperation was chosen since it lends itself as a natural testing ground for automatic obstacle avoidance and human–robot interaction approaches. An overview of this simple experimental setup is depicted in Figure 7.

Total of 9 test subjects (3 females and 6 males) participated in the mini-study. The mean age of test subjects was 32.78 with the standard deviation of $\pm$10.52 years. The test subjects were students and staff of the Faculty (please note that none of the staff was involved in the development of the experiment). Out of all test subjects, four (44%) had some previous experience with teleoperation. Test subjects were informed about the goal of the study as well as a remote controller setup used for controlling the robot. The test subjects were placed in a different room than a mobile robot (as depicted in Figure 7) and the only feedback they had from a robot was a video stream coming from the camera mounted on a mobile base system (Manhattan Webcam 500, having 55° horizontal viewing angle, and running in 25 fps with resolution set to 680 × 480). The camera

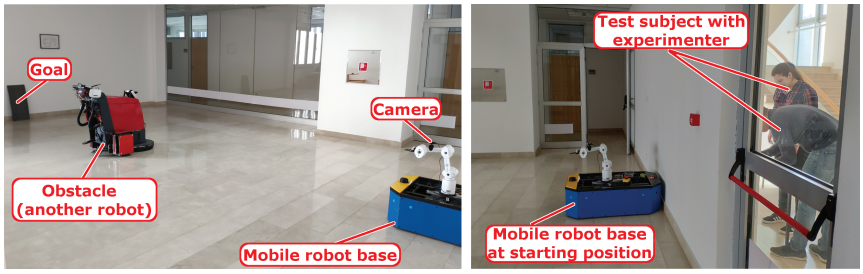


Figure 7 Overview of a simple teleoperation scenario used in the experiment.

was positioned in such a manner that the area near the front of the mobile base was not visible. This was done in order to simulate the lack of situational awareness during teleoperation. The robot was running the proposed mediation algorithm, in which NN-based obstacle avoidance was one input to the mediator and human controller input the other one. Thus, if the user was too close to an obstacle the NN-based avoidance would gradually (depending on CP value) take over the control, steer it from danger and (gradually) give back the control to the user. Users were informed that two cases would occur (in random order): no unplanned obstacles case and unplanned obstacles case (thrown in front of the mobile robot by a second experimenter in such a manner that it cannot easily be seen by the camera). This procedure was repeated several times, and collision/no-collisions events recorded. Also, at the end of the test runs each user had to fill out a simple questionnaire (and share her/his experience in a free-form interview) with the following questions (other questions not reported here pertained to demographics):

1. Did you feel in full control during the teleoperation?
2. Do you feel that automatic obstacle avoidance through mediation helped you during teleoperation?
3. Was completing the required task easier with or without the mediation?

Answers for the first two questions were on a Likert-like scale with three labelled answers (e.g., "Strongly agree," "Neutral," and "Strongly disagree") and the continuous line between them. The users had to mark a place on the line which best matched their response. This, in turn, enabled us to obtain continuous variable data from the responses. Please note that the users were not allowed to practice the mediated teleoperation so that they would not get accustomed to it. They were, however, free to test the remote controller and video feedback. Results for this experiment are presented in Section 4.3.

4. RESULTS AND DISCUSSION

The results reported throughout Section 4 are summarized in Table 1 for easier reading. They will be explained in more detail in the following subsections. Please note the bottom row pertaining to total distance covered during reported testing, as well as the time and number of instances tested. This is indicative of the extent to which the proposed approach was tested (keep in mind when interpreting the results this was all indoor testing).

4.1. Simulation

Example trajectories (random 10 out of 15 for both test cases) obtained in the simulation are presented in Figure 8.

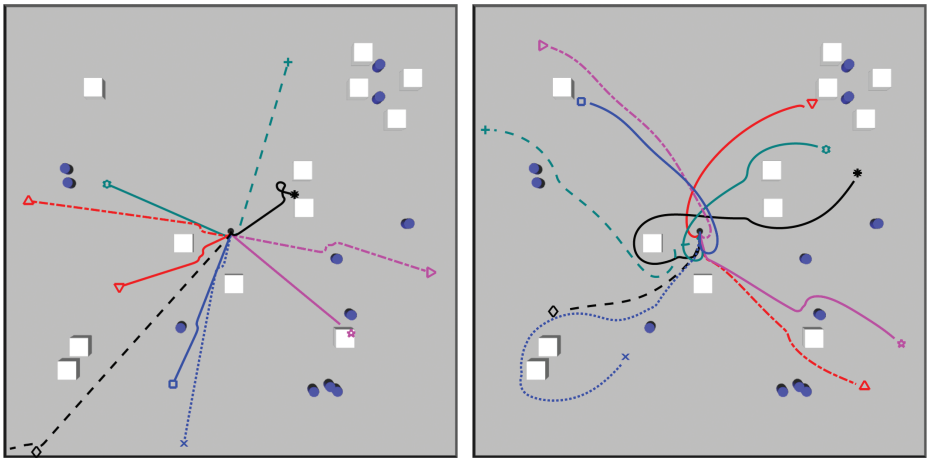
It is worth noting that for NN-based navigation controller the mobile base never crashed, while for the P-type based navigation controller in four cases (26%) crash occurred. Observing the robot behavior during the experiment, it was concluded this was mainly due to the "aggressive" nature of the P-type controller (especially in rotation) which tries to get to the goal point in a straight line (while NN-based navigation tries to do the same but with a slightly arched trajectory). This then makes it hard for NN-based obstacle avoidance to recover (or even results in LiDAR failure due to an obstacle being too close). However, additional parameters, like goal points too close to the obstacles or in physically unreachable cases might have, in certain cases, contributed to the crash event.

Looking at the presented trajectories in Figure 8 several interesting observations can be made. First, it is clear that P-type controller results in a more direct approach to the goal (i.e., "aggressive" nature mentioned before), while the NN-based navigation takes a more circular trajectory even when the goal is in the line of sight with no obstacles in-between. From Figure 8a it can also be observed that when the goal is too close to the obstacle (trajectory with the dark dashed line and circle goal marker) the robot might not reach its intended goal (since mediation algorithm switches it to obstacle avoidance). Some improvement might be possible here if adaptive CP calculation is employed using remaining distance to the goal as well as adaptive robot speed (but obviously there is a limit to this approach). Next, when the goal is placed within the obstacle (especially if it is within a corner type environment - dark full line trajectory with pentagram goal marker) the crash is inevitable since the obstacle avoidance does not react appropriately (there is no stop-and-turn in-place mechanism). This, however, does not in our view represent the failure of the proposed fuzzy mediation approach (since it worked as intended) but highlights shortcomings of the developed obstacle avoidance approach (and ways of improving it, e.g., in-place rotation). Finally, if the target is close to the obstacle, but still far enough so that obstacle avoidance is not activated (dark full line trajectory with full circle goal marker), it might take a while for a robot to reach the goal (switching several times between navigation and obstacle avoidance controller - note the end part of the trajectory), but it will reach it. Looking more closely at the Figure. 8b it can be seen once more that the robot does not take the most direct route in NN-based navigation and that obstacles sometimes interfere with the navigation (i.e., trigger the obstacle avoidance) as depicted by dark full line trajectory with asterisk goal marker and dark dotted trajectory with x goal marker. This extends

Table 1 | Summary of experimental results.

Env.	Robot	Scenario	Controller setup[a]	ADT ± STD [m]	ATT ± STD [s]	Trials
Simulation	Turtlebot 2	Obstacle course	NN/NN	13.08 ± 4.3	68.8 ± 18.87	15 each
			NN/P	7.44 ± 3.32	38.4 ±16.15	
			NN/NN	8.75 ± 1.89	43.76 ± 1.94	
		Z-shaped obstacle	NN/P	8.20 ± 0.17	46.32 ± 1.62	5 each
			DWA/ROS	9.80 ± 2.77	37.91 ± 8.18	
			NN/NN	12.25 ± 0.69	69.15 ± 3.75	
	Turtlebot 2	U-shaped obstacle	NN/P	12.66 ± 1.12	73.06 ± 6.57	5 each
			DWA/ROS	7.64±1.80	39.61 ± 30.73	
			NN/NN	51.11 ± 14.55	316.78 ± 29.32	
Real-world		Navigation	NN/P	75.76 ± 10.92	466.55 ± 18.68	2 each
			DWA/ROS	47.41 ± 12.78	245.50 ± 70.80	
			NN/NN	13.37 ± 2.89	73 ± 13.11	
		Obstacle course	NN/P	12.39 ± 3.10	68.33 ± 17.21	3 each
			DWA/ROS	13.93 ± 1.79	110.33 ± 11.50	
	Custom built		NN/NN	47.55 ± 15.95	301.5 ± 112.43	
		Navigation	NN/P	53.11 ± 14.98	403.60 ± 60.81	2 each
			DWA/ROS	47.51 ± 7.82	223.5 ± 19.09	
			NN/line following	49.31 ± 20.41	288 ± 93.34	
			TOTAL	1,466.78 (78.27%[b])	8,401.29 (80.28%[b])	83 (79.52%[b])

Env. = Environment; ADT = Average Distance Travelled; ATT = Average Time Taken; STD = Standard Deviation; DWA= Dynamic Window Approach; NN =neural network; ROS = Robot Operating System.
(a) obstacle avoidance controller type / navigation controller type. (b) percentage of total value recorded during testing of the proposed method (i.e., not including the baseline method).



(a) P-type navigation controller. (b) NN-based navigation controller.

Figure 8 | Trajectories acquired during testing mediation in a simulation environment using Turtlebot 2 model. Please note that goal points are marked with different symbols.

distance travelled, but the robot ultimately reaches its goal, even when the goal point is close to the obstacle (trajectories with a full dark line and triangle end marker and full dark line with square end marker).

Despite several crashes, and based on the above discussion, we concluded that the proposed fuzzy mediation algorithm is a viable algorithm and it performed as expected in the simulation. Thus, we were encouraged to deploy it in real-world scenarios and on real mobile robots. It should be noted that during the transfer no additional re-training of the NNs was performed (i.e., they were fully trained in simulation).

4.2. Simple Real-World Scenarios

As was explained in Section 3.2 first real-world testing was carried out using modified Turtlebot 2 mobile robot in case of Z- and U-shaped obstacles. Example of obtained trajectories for one random case per setup can be seen in Figure 9 along with associated CP and mediation coefficients.

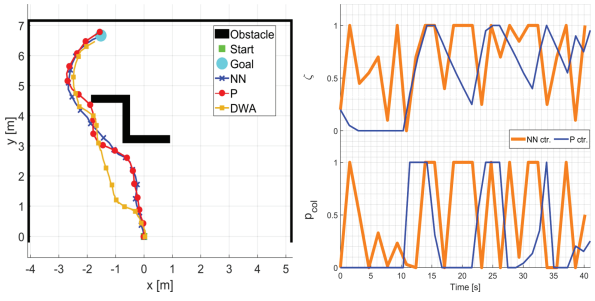
The figure shows that for all three used setups the robot reached goal area (there is a small error for DWA end position, but we contribute this to AMCL/encoder related issues in the recorded data, since the physical robot reached the target). As expected, DWA reached the goal in a more direct trajectory in both cases, while the proposed approach had a few direction changes due to the fusion of navigation and obstacle avoidance parts. Here, again a more "aggressive" nature of the P-type controller is seen: there are more direction changes in its trajectory than in a case of NN driven navigation which is smoother. However, in all particular cases, the proposed mediation algorithm performed as expected, enabling the robot to execute a simple navigation task.

It is interesting to note that DWA also had some issues with U-shaped obstacles. This resulted in longer navigation run than in both setups for the proposed approach (94.93 s versus 62.94 and

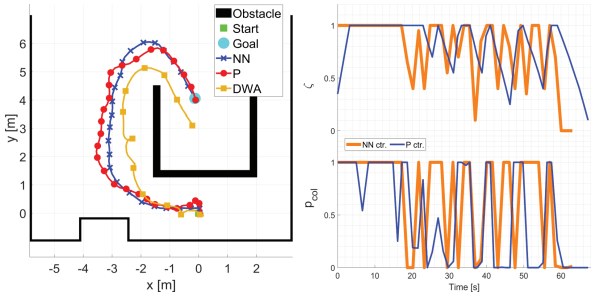
67.36 s for NN and P setup, respectively). Completion times for Z-shaped obstacle were similar for all three setups (DWA: 41.63 s, NN: 39.06 s, and P: 41.27 s, for this particular example). However, on average ROS/DWA was the fastest method in all cases, but not always the one with the shortest distance travelled (please see Table 1). In all experimental runs except one (for U-shaped obstacle) robot went to the left of the obstacle to complete the course. In that one case, the robot went to the right of the obstacle and into the narrow space (1.4 m) between the wall and the obstacle. This resulted in the longer run (102.32 s) with more direction changes due to the fact that during the entire run, control was dominated by obstacle avoidance. These tests in our view demonstrated that the proposed approach can be applied even in more demanding cases like U-shape obstacle (but with certain limitations).

The change in CP and ζ parameter values for both test cases (where mediation was used) from the example in Figure 9, can be seen in (b) and (d) parts of the figure. Please note that the value of 1 for parameter ζ means that the robot is in full obstacle avoidance mode, and 0 that it is in pure navigation mode. All the values in between mean that the mediation approach is taking into account both the navigation and obstacle avoidance controller (in appropriate proportion).

Next, a more demanding navigation task was used for the Trutlebot 2 robot. Examples of the obtained trajectories (for two different goal points) are depicted in Figure 10. Please note that although obtained trajectories are plotted on the map, the robot itself does not have access to the map in the proposed approach (but does have it, without the obstacles, for ROS/DWA case). In essence, the robot path planning is a straight line, going from start to the goal position in the proposed approach. From the figure, it can be observed that robot successfully reached given goals in all cases, with ROS/DWA again having a more direct route which is reflected in completion



(a) Trajectories (Z) (b) p_{col} and ζ values (Z)



(c) Trajectories (U) (d) p_{col} and ζ values (U)

Figure 9 Examples of simple navigation task results with U- and Z-shaped obstacles.

times in Table 1. The main reason for this is that (upper) corridor in which robot had to enter was narrow (1.8 m) and resulted in activation of obstacle avoidance part which turns the robot away from it if approach angle was not appropriate. It is also evident that trajectories obtained using NN-based navigation controller resulted (again) in smoother trajectory, compared to ones with P-type controller which was a consequence of the fact that P-type controller was too direct (i.e., aggressive) in trying to reach the goal. In terms of speed, that is, completion time, the following results were obtained: 295.56 and 195.44 s for DWA case, 337.51 and 296.05 s for NN case, and finally 479.76 and 453.34 s for P case. It should be noted that for certain start-goal point configurations robot would get stuck at certain areas in space, switching constantly between obstacle avoidance and navigation. A similar effect was reported in Ref. [20], and could potentially be reduced/eliminated by insertion of several waypoints.

In the next stage of testing a larger (custom built) robot with different footprint was used. It was first tested on a larger obstacle course where it had to go from start to a goal position. The experimental setup and the obtained trajectories are presented in Figure 12 and results summarized in Table 1.

From the results, it can be seen that the robot successfully (i.e., without the crash) completed the task in all test cases (please note again that in mediation cases the robot did not have access to the map, while for ROS/DWA it had but without obstacles in it). As in all cases before, the ROS/DWA seems to have a more direct trajectory. This is however misleading since this approach had, in all cases, number of in-place turns (which are not visible in the figure, but are reflected in time results in Table 1) and even some backward driving (as seen in the figure as a bottom dotted dark line trajectory). This is confirmed when completion times are examined (for goal points from top to bottom): NN navigation had times of 59 s, 75 s, and 85 s, P-type navigation controller had times of 49 s, 74 s, and 82 s, while ROS navigation stack had times of 99 s, 122 s, and 110 s, respectively. Looking at times it can be concluded that due to more direct approach P-type controller had the fastest times and shorter

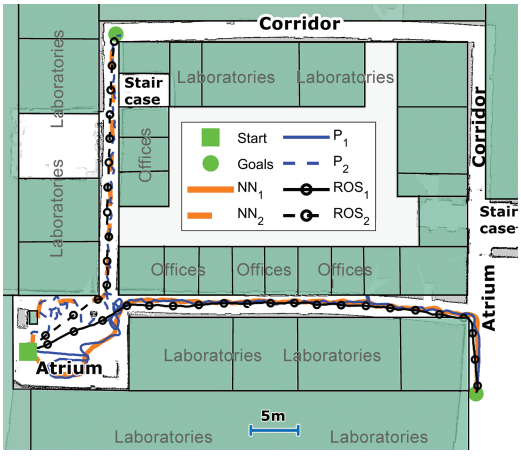


Figure 10 Examples of navigation trajectories for a realistic environment. A map depicts part of the 4th floor at authors' home institution (overlaid over robot generated map using Robot Operating System [ROS] *gmapping* package).

distance travelled (Table 1). Regardless of achieved times the proposed fuzzy mediation algorithm performed as intended, mediating between two distinct robot behaviors. Please note that improved/-faster performance might be achieved if numerous ROS/DWA navigation stack parameters are better tuned. During our experiments, we had to tune four DWA parameters in order to get the algorithm to work properly with the robot. For the proposed approach, on the other hand, the only parameters that were changed (alongside adjusted footprint) compared to the Turtlebot 2 case were the dimensions of uncertainty ellipses' (they were increased due to increased dimensions of the robot).

As the final test, a more complex navigation task with the custom-built robot was introduced. The environment was the same as in the complex navigation task of Turtlebot 2. Obtained results are depicted in Figure 11.

From the trajectories presented in the figure, it is clear that in all cases the robot successfully completed the given task. However, the manner in which it was completed was slightly different, especially for the P-type navigation controller. To be more precise, the mobile robot, in that case, took numerous direction corrections (as can be seen best from thin solid dark line trajectory) due to the fact that it aggressively changed its direction, and due to robot's larger dimensions in relation to corridor width, thus often engaged obstacle avoidance. This behavior was not detected in the case of NN-based navigation, where the trajectory was smoother. It should be noted that even ROS navigation stack with DWA had several in turn rotations when coming into the narrow corridors. If completion times are examined the following values are obtained (to the first and the second goal point respectively). The P-type controller finished the course in 446 s and 360 s, while the NN-based navigation approach finished it in 381 s and 222 s. On the other hand, our simple/custom line path planning and line following algorithm (with three way-points going through the walls) finished the course in 354 s and 222 s, and ROS based navigation stack with DWA in 210 s and 237 s. Again, as before it should be noted that better performance could be achieved by parameter optimisation of ROS/DWA method as well as our approach, but that was not the aim of the research. From the completion times for the P-type controller, the effect of numerous rotations and direction corrections are evident, having the slowest time in both cases (by a large margin). The remaining three approaches demonstrated comparable performance in the case of the second goal point (right part of the figure), while for the first goal point, ROS/DWA based approach was the fastest. Also, it was noted that adding way-points, in general, decreased completion time and helped the proposed algorithm to deal with harder situations (like convex dead-end obstacle). Regardless of these times, the performance demonstrated that the fuzzy mediation-based approach can produce good and reliable results even when a larger robot is used in a realistic environment, especially with the addition of several way-points.

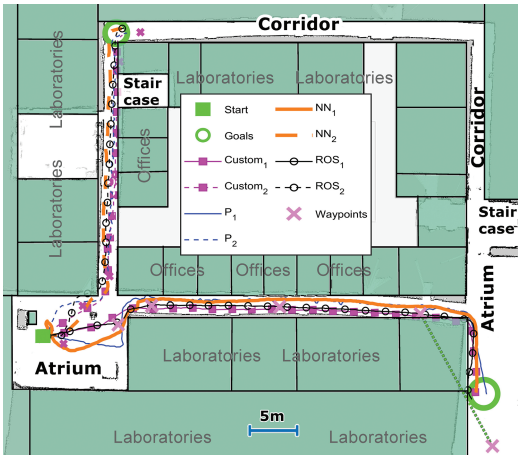
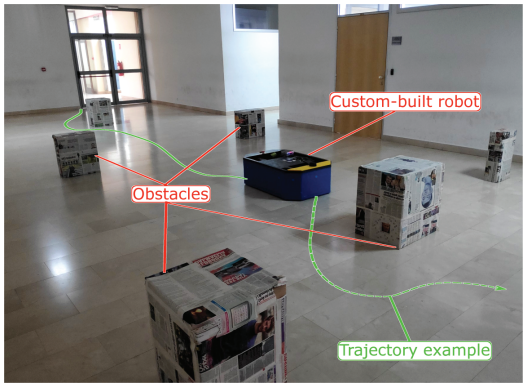


Figure 11 Examples of navigation trajectories for a realistic environment using a custom-built mobile robot platform. A map depicts part of the 4th floor at authors' home institution (overlaid over robot generated map using Robot Operating System [ROS] *gmapping* package).

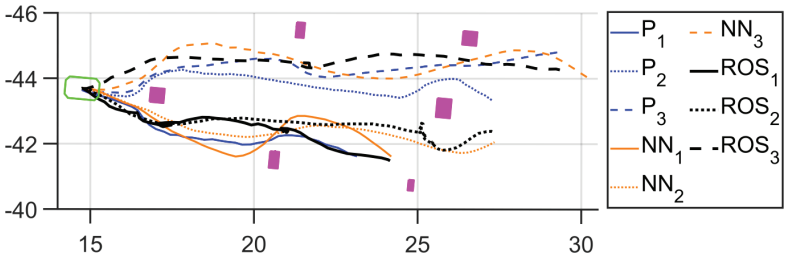
4.3. Teleoperation Scenario

During the application-based testing, users in all test cases successfully completed the given task. In 100% of cases with surprise obstacle the mediation algorithm activated the NN-based obstacle avoidance, while in 47% of cases with no surprise obstacles the fuzzy mediation activated obstacle avoidance. This, in our view, highlights the need for inclusion of such a safety mechanism (possibly fine-tuned by a teleoperator) in teleoperation, since with even one large obstacle (as was the case here) users did not manage to complete the task without the danger of damaging either the robot or the obstacle. It should, however, be noted that in 23% of all cases when mediator transferred the control to obstacle avoidance, the robot did not hit the obstacle but did slightly brush against it. This is an indication that additional work is needed in adaptive mediator parameter adjustment as well as improving the obstacle avoidance algorithm (possibly training it to react at larger distances than 1 m, as was the case and/or implementing in-place rotation). Another worthwhile remark, recorded during test subjects' post measurement interviews, is that test subject were not always sure (especially in cases when control was not fully transferred to obstacle avoidance) if the robot is in obstacle avoidance or is it simply an issue in communication/lag. Thus they suggested including a graphical user interface (or even tactile feedback) indicating when and how strongly did fuzzy mediator transfer control to obstacle avoidance (or any other controller for that matter).

Looking at the responses for the first two questions of the mini-survey which test subject completed after finishing the experiment, the following results were obtained. For the question *Did you feel in full control during the teleoperation?* the mean response had a value of 66.4 with standard deviation of 11.0 (where 0 represents "I was not in a full control." response, and 100 "I was in a full control." response, while 50 represented neutral response). For the question *Do you feel that automatic obstacle avoidance through mediation helped you during teleoperation?* the mean response had a value of 66.0 with a standard deviation of 15.1 (where 0 represented "I strongly feel it did not." response, and 100 "It strongly feel it did." response, while 50 represented neutral response). Finally for the third question *Was completing the required task easier with or without the mediation?* 8 out of 9 test subjects (89%) felt that the task completion was easier with the mediation.



(a) Experimental setup.



(b) Example trajectories with artificial obstacles marked as filled blocks (in scale). Please note that the test space was Atrium in the right part of Figures 10 and 11.

Figure 12 Experimental measurement with the custom-built robot on an obstacle course.

Based on the obtained results from the mini-study we can conclude that in general users found the approach to be helpful and felt that they were in the control of a robot (results which might have been slightly better if feedback about mediation status was provided). However, due to the small sample size used and simple tasks given, further analysis is needed to make definite conclusions on the impact of the fuzzy mediation in the teleoperation scenario. Nevertheless, the results are encouraging and they demonstrate the viability of the proposed fuzzy mediation approach in practical real-world problems.

5. CONCLUSIONS

The paper presents an idea of fuzzy-based mediation for (simple) mobile robot navigation tasks and implements it on a couple of real robots. It combines the outputs of two controllers, one dedicated to navigation to the goal point and the other for obstacle avoidance, to the final output which is (in general) a combination of the outputs of the two controllers. The approach is thoroughly tested, both in simulation and in real-world scenarios, using two mobile robots of different architecture, footprint sizes and shapes.

While it was previously shown that NN can be used to solve robot navigation task and obstacle avoidance task separately, the experiments in the paper demonstrate that the problem can be broken down into task primitives (with or without the use of NNs) with the application of adaptive fuzzy mediation, thus adapting robot behavior. The NN ensembles approach for fusing the output of two or more NNs has been demonstrated before, but what the proposed approach enables is a more transparent manner in which this is done and which can be fine-tuned intuitively to adaptively change. Also, it is not limited only to NN-based controllers, but other controllers like PID controller can be used. In this manner, more complex behavior can be achieved by combining task primitives while enabling modularity, that is, controllers addressing certain task can be changed, without influencing other task controllers.

Obtained results demonstrate that the inclusion of different types of controllers (while keeping the mediation mechanism unchanged) is possible, which opens up additional possibilities for adaptation and optimisation. Although results show good performance (especially in realistic tasks), comparison to a baseline method shows some differences (in general slower completion time and longer distance travelled, although there were some cases where the proposed approach was on average faster and with shorter trajectory length), but overall performance is satisfactory (and without any human intervention). However, there are some issues that should be noted. First, there were situations (in terms of obstacle shape and relative start-goal relation) with the proposed approach which resulted in robot getting stuck in the infinite loop type behavior (e.g., if the robot is placed deep inside U-shape obstacle and needs to go to start point as its goal point as depicted in Figure 9). This issue with a similar approach was also noted in Ref. [20]. We believe this could be addressed with RL or inclusion of automatically generated virtual way-points in cases when looping is detected. The second thing to note is that there were more crashes with the proposed approach than with the baseline algorithm (ROS/DWA). While this is certainly undesired behavior, it happened in a subset of events where the proposed fuzzy mediation worked as expected, but NN-based obstacle avoidance was simply not up to the task. The issue could be resolved, or at least reduced, by a better trained NN obstacle

avoidance controller (or other type of controller) or by fine-tuning the mediator algorithm (e.g., faster hand-off to the obstacle avoidance, or CP calculation further in time although this would have a limited effect since the hand-off would be made earlier, but current NN would not react in time since it was trained for 1 m distance). Thirdly, some minor parameter tuning had to be done after transferring the algorithm to a mobile robot with a different footprint, but the same had to be done for a baseline algorithm (ROS/DWA) as well. This could also be potentially addressed by an adaptive change of parameters, but we believe that ultimately certain engagement by a human operator in the tuning process will still be needed.

The possible improvements (and our future research directions) include the following: adaptive change of algorithm's parameters based on robot footprint and sensor placement, inclusion of better trained NNs and/or NNs with different architecture like RNNs, NNs capable of reliably avoiding dynamic obstacles, fuzzy set approach in obstacle avoidance NNs for switching between left and right turns, testing with nonlinear mediation engines, inclusion of other types of controllers as well as testing with other task primitives, developing the approach to accommodate more than two controllers, and testing on additional robots and in more complex teleoperation scenarios. Finally, we plan to implement a simulation-based tool for testing and fine-tuning parameter values used in the approach for different environments, robot setups, and requirements.

CONFLICT OF INTEREST

The authors declare that there is no conflicts of interests.

Funding Statement

This research has been partially supported by the Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, University of Split through project "Computer Intelligence in Recognition and Support of Human Activities".

AUTHORS' CONTRIBUTIONS

J.M. and S.K. conceived the research idea as well as experimental design. They carried out data acquisition, and participated in data analysis and manuscript preparation. I.S. and V.P. participated in data analysis and interpretation, as well as manuscript preparation, providing critical feedback that helped shape the research and the manuscript.

ACKNOWLEDGMENTS

Authors would like to express gratitude to the company "Statim d.o.o.", for providing/loaning one of the robots (depicted in Figure 6) used in the experimental testing. Authors would also like to thank to all the students and the Faculty staff who took part in the teleoperation study.

REFERENCES

[1] L. Kunze, N. Hawes, T. Duckett, M. Hanheide, T. Krajník, Artificial intelligence for long-term robot autonomy: a survey, IEEE Robot. Autom. Lett. 3 (2018), 4023–4030.

[2] J. Guiochet, M. Machin, H. Waeselynck, Safety-critical advanced robots: a survey, Robot. Auton. Syst. 94 (2017), 43–52.

[3] D. Ramachandram, G.W. Taylor, Deep multimodal learning: a survey on recent advances and trends, IEEE Signal Process. Mag. 34 (2017), 96–108.

[4] F. Bonin-Font, A. Ortiz, G. Oliver, Visual navigation for mobile robots: asurvey, J. Intell. Robot. Syst. 53 (2008), 263–296.

[5] R.A. Jarvis, Intelligent robotics: past, present and future, Int. J. Comput. Sci. Appl. 5 (2008), 23–35. http://www.tmrfindia.org/ijcsa/v5i33.pdf

[6] V. Sze, Y. Chen, T. Yang, J. S. Emer, Efficient processing of deep neural networks: a tutorial and survey, Proc. IEEE. 105 (2017), 2295–2329.

[7] J. Wang, Z. Wang, D. Zhang, J. Yan, Combining knowledge with deep convolutional neural networks for short text classification, in Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI-17, AAAI Press, Melbourne, 2017, pp. 2915–2921.

[8] Y. Zhu, R. Mottaghi, E. Kolve, J.J. Lim, A. Gupta, L. Fei-Fei, A. Farhadi, Target-driven visual navigation in indoor scenes using deep reinforcement learning, in 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 2017, pp. 3357–3364.

[9] A. Giusti, J. Guzzi, D.C. Ciresan, F.-L. He, J.P. Rodriguez, F. Fontana, *et al.,* A machine learning approach to visual perception of forest trails for mobile robots', IEEE Robot. Autom. Lett. 1 (2016), 661–667.

[10] L. Ran, Y. Zhang, Q. Zhang, T. Yang, Convolutional neural network-based robot navigation using uncalibrated spherical images, Sensors. 17 (2017), 1341.

[11] C. Chen, A. Seff, A. Kornhauser, J. Xiao, Deepdriving: Learning affordance for direct perception in autonomous driving, in 2015 IEEE International Conference on Computer Vision (ICCV), ICCV '15, IEEE Computer Society, Washington, 2015, pp. 2722–2730.

[12] A. Jain, A. Singh, H. S. Koppula, S. Soh, A. Saxena, Recurrent neural networks for driver activity anticipation via sensory-fusion architecture, in 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, 2016, pp. 3118–3125.

[13] A. Geiger, P. Lenz, C. Stiller, R. Urtasun, Vision meets robotics: the kitti dataset, Int. J. Robot. Res. 32 (2013), 1231–1237.

[14] W. Maddern, G. Pascoe, C. Linegar, P. Newman, 1 year, 1000 km: the Oxford robotcar dataset, Int. J. Robot. Res. 36 (2017), 3–15.

[15] S. Toprak, N. Navarro-Guerrero, S. Wermter, Evaluating integration strategies for visuo-haptic object recognition, Cogn. Comput. 10 (2017), 408–425.

[16] Y.-M. Chou, Y.-M. Chan, J.-H. Lee, C.-Y. Chiu, C.-S. Chen, Merging deep neural networks for mobile devices, in IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Salt Lake City, 2018.

[17] Z. Ahmad, J. Zhang, Combination of multiple neural networks using data fusion techniques for enhanced nonlinear process modelling, Comput. Chem. Eng. 30 (2005), 295–308.

[18] G. Vincenti, G. Trajkovski, Fuzzy mediation in shared control and online learning, Intelligence Integration in Distributed Knowledge Management, in: D. Król, N.T. Nguyen (Eds.), Intelligence Integration in Distributed Knowledge Management, IGI Global, Hershey, 2009, pp. 263–285.

[19] A. Maturo, E. Sciarra, A.G.S. Ventre, Counselling: decision making, consensus, and mediation, Procedia Soc. Behav. Sci. 5 (2010), 1770–1776.

[20] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, C. Cadena, From perception to decision: a data-driven approach to end-to-end motion planning for autonomous ground robots, in 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 2017, pp. 1527–1533.

[21] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A.Y. Ng, ROS: an open-source robot operating system, in ICRA Workshop on Open Source Software, Kobe, 2009, vol. 3, p. 5.

[22] S. Levine, C. Finn, T. Darrell, P. Abbeel, End-to-end training of deep visuomotor policies, J. Mach. Learn. Res. 17 (2016), 1334–1373. http://jmlr.org/papers/v17/15-522.html

[23] Y. LeCun, U. Muller, J. Ben, E. Cosatto, B. Flepp, Off-road obstacle avoidance through end-to-end learning, in Proceedings of the 18th International Conference on Neural Information Processing Systems, NIPS'05, MIT Press, Cambridge, 2005, pp. 739–746.

[24] M. Bojarski, D.D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, *et al.,* End to end learning for self-driving cars, arXiv CoRR, vol. abs/1604.07316, 2016. https://arxiv.org/abs/1604.07316

[25] S. Shalev-Shwartz, S. Shammah, A. Shashua, Safe, multi-agent, reinforcement learning for autonomous driving, in Proceeding of the NIPS Workshop on Learning, Inference and Control of Multi-Agent System, Barcelona, 2016.

[26] R. Hadsell, P. Sermanet, J. Ben, A. Erkan, M. Scoffier, K. Kavukcuoglu, U. Muller, Y. LeCun, Learning long-range vision for autonomous off-road driving, J. Field Robot. 26 (2009), 120–144.

[27] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, J. Malik, Cognitive mapping and planning for visual navigation, in 2017 IEEE Conferenceon ComputerVisionand Pattern Recognition(CVPR), Honolulu, 2017, pp. 7272–7281.

[28] D. Gandhi, L. Pinto, A. Gupta, Learning to fly by crashing, in 2017IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, 2017, pp. 3948–3955.

[29] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A.J. Ballard, A. Banino, *et al.,* Learning to navigate in complex environments, arXiv CoRR, vol. abs/1611.03673, 2016. https://arxiv.org/abs/1611.03673

[30] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, *et al.,* Using simulation and domain adaptation to improve efficiency of deep robotic grasping, in 2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, 2018, pp. 4243–4250.

[31] F. Sadeghi, S. Levine, Cad2rl: real single-image flight without a single real image, in Proceeding of Robotics: Science and System, Cambridge, 2017.

[32] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, S. Birchfield, Deep object pose estimation for semantic robotic grasping of household objects, in: A. Billard, A. Dragan, J. Peters, J. Morimoto (Eds.), Proceeding of The 2nd Conference on Robotics Learning, vol. 87 of Proceedings of Machine Learning Research, PMLR, Zürich, 2018, pp. 306–316.

[33] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, Human-level control through deep reinforcement learning, Nature. 518 (2015), 529–533.

[34] T. Zhang, G. Kahn, S. Levine, P. Abbeel, Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search, in 2016 International Conference on Robotics and Automation (ICRA), Stockholm, 2016, pp. 528–535.

[35] J. Lunenburg, R. van de Molengraft, M. Steinbuch, A representation method based on the probability of collision for safe robot navigation in domestic environments, Auton. Robots. 42 (2018), 601–614.

[36] C.-H. Chen, C.-C. Wang, Y.T. Wang, P.T. Wang, Fuzzy logic controller design for intelligent robots, Math. Probl. Eng. 2017, (2017), 1–12.

[37] L.K. Hansen, P. Salamon, Neural network ensembles, IEEE Trans. Pattern Anal. Mach. Intell. 12 (1990), 993–1001.

[38] L. Tai, S. Li, M. Liu, Autonomous exploration of mobile robots through deep neural networks, Int. J. Adv. Robot. Syst. 14 (2017), 1729881417703571.

[39] S.-B. Cho, J.H. Kim, Multiple network fusion using fuzzy logic, IEEE Trans. Neural Netw. 6 (1995), 497–501.

[40] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, G.E. Hinton, Adaptive mixtures of local experts, Neural Comput. 3 (1991), 79–87.